



UNIVERSIDAD CÉSAR VALLEJO

**FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA MECÁNICA
ELÉCTRICA**

Modelo híbrido basado en stacking para mejorar la predicción de la
temperatura del motor eléctrico raghavendra

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero Mecánico Electricista

AUTORES:

Enco Guerrero, Jhony Alberto (orcid.org/0000-0003-1692-5027)

Lorenzo Perez, Rocio Pilar (orcid.org/0000-0002-7103-9741)

ASESORA:

Mg. Serrepe Ranno, Miriam Marcela (orcid.org/0000-0001-9342-1717)

LÍNEA DE INVESTIGACIÓN:

Modelamiento y Simulación de Sistemas Electromecánicos

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo sostenible y adaptación al cambio climático

TRUJILLO – PERÚ

2023

Dedicatoria

Dedico esta tesis a mis padres y familia por el apoyo brindado durante esta etapa de mi vida profesional, guiándome para alcanzar mis objetivos.

A mis docente y amistades que han Contribuido para que este trabajo se pueda desarrollar de la menor manera en esta vida profesional.

Enco Guerrero, Jhony Alberto

Dedico esta tesis mis padres y familia por el apoyo brindado durante esta etapa de mi vida profesional, por ayudarme a lograr mis objetivos.

A mis docente y amistades que han sido parte para que este trabajo se pueda desarrollar de la mejor manera en esta vida profesional.

Lorenzo Pérez, Rocio Pilar

Agradecimiento

Agradezco a Dios, mis padres, mi pareja y familia por el apoyo durante esta etapa de mi vida profesional por ayudarme a lograr mis objetivos.

A mis compañeros de trabajo por brindarme el apoyo y poder cumplir con mis horarios de clases, a mis docentes, asesor y amistades que han contribuido para que este trabajo se pueda desarrollar de la menor manera.

Enco Guerrero, Jhony Alberto

Agradezco a Dios, mis padres y familia por el apoyo durante esta etapa de mi vida profesional por ayudarme a lograr mis objetivos.

A mis compañeros de trabajo por brindarme el apoyo y poder cumplir con mis horarios de clases, a mis docentes, asesor y amistades que han contribuido para que este trabajo se pueda desarrollar de la menor manera.

Lorenzo Pérez, Rocio Pilar

Declaratoria de autenticidad del asesor



UNIVERSIDAD CÉSAR VALLEJO

**FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA MECÁNICA ELÉCTRICA**

Declaratoria de Autenticidad del Asesor

Yo, SERREPE RANNO MIRIAM MARCELA, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA MECÁNICA ELÉCTRICA de la UNIVERSIDAD CÉSAR VALLEJO SAC - TRUJILLO, asesor de Tesis titulada: "Modelo Híbrido Basado en Stacking para Mejorar la Predicción de la Temperatura del Motor Eléctrico Raghavendra", cuyos autores son LORENZO PEREZ ROCIO PILAR, ENCO GUERRERO JHONY ALBERTO, constato que la investigación tiene un índice de similitud de 12.00%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

TRUJILLO, 12 de Diciembre del 2023

Apellidos y Nombres del Asesor:	Firma
SERREPE RANNO MIRIAM MARCELA DNI: 06437594 ORCID: 0000-0001-9342-1717	Firmado electrónicamente por: SRANNOMM el 14- 12-2023 09:43:31

Código documento Trilce: TRI - 0694215



Declaratoria de originalidad de los autores



UNIVERSIDAD CÉSAR VALLEJO

**FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA MECÁNICA ELÉCTRICA**

Declaratoria de Originalidad de los Autores

Nosotros, LORENZO PEREZ ROCIO PILAR, ENCO GUERRERO JHONY ALBERTO estudiantes de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA MECÁNICA ELÉCTRICA de la UNIVERSIDAD CÉSAR VALLEJO SAC - TRUJILLO, declaramos bajo juramento que todos los datos e información que acompañan la Tesis titulada: "Modelo Híbrido Basado en Stacking para Mejorar la Predicción de la Temperatura del Motor Eléctrico Raghavendra", es de nuestra autoría, por lo tanto, declaramos que la Tesis:

1. No ha sido plagiada ni total, ni parcialmente.
2. Hemos mencionado todas las fuentes empleadas, identificando correctamente toda cita textual o de paráfrasis proveniente de otras fuentes.
3. No ha sido publicada, ni presentada anteriormente para la obtención de otro grado académico o título profesional.
4. Los datos presentados en los resultados no han sido falseados, ni duplicados, ni copiados.

En tal sentido asumimos la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de la información aportada, por lo cual nos sometemos a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

Nombres y Apellidos	Firma
JHONY ALBERTO ENCO GUERRERO DNI: 71517379 ORCID: 0000-0003-1692-5027	Firmado electrónicamente por: JAENCOE el 12-12-2023 18:38:09
ROCIO PILAR LORENZO PEREZ DNI: 72488158 ORCID: 0000-0002-7103-9741	Firmado electrónicamente por: RLORENZOP el 12-12-2023 18:41:14

Código documento Trilce: TRI - 0694216

Índice de contenidos

Dedicatoria	ii
Agradecimiento	iii
Declaratoria de autenticidad del asesor	iv
Declaratoria de originalidad de los autores	v
Índice de contenidos	vi
Índice de tablas	vii
Índice de figuras	xi
Resumen	xvii
Abstract	xviii
I. INTRODUCCIÓN	1
II. MARCO TEÓRICO	5
III. METODOLOGÍA	18
3.1. Tipo y diseño de investigación	18
3.2. Variables y Operacionalización	19
3.3. Población, muestra y muestreo	20
3.4. Técnicas e instrumentos de recolección de datos	20
3.5. Procedimientos	21
3.6. Método de análisis de datos	22
3.7. Aspectos éticos	22
IV. RESULTADOS	24
V. DISCUSIÓN	35
<i>Tabla 14.</i> Resumen de valores obtenidos por algoritmos y Stacking.	35
VI. CONCLUSIONES	40
VII. RECOMENDACIONES	41
REFERENCIAS	42
ANEXOS	50

Índice de tablas

Tabla 1. Cuadro comparativo de las métricas de ML usadas en estudios previos para la predicción de la temperatura de los motores eléctricos.....	8
Tabla 2. Cuadro comparativo de las técnicas usadas en estudios previos para la predicción de la temperatura de los motores eléctricos.	9
Tabla 3. Términos de las métricas de evaluación	16
Tabla 4. Métrica de evaluación temperatura del motor – R squared Stacking 1 ..	27
Tabla 5. Métrica de evaluación temperatura del motor – R squared Stacking 2 ..	27
Tabla 6. Métrica de evaluación temperatura del motor — Mean absolute error Stacking 1.....	28
Tabla 7. Métrica de evaluación temperatura del motor — Mean absolute error Stacking 2.....	29
Tabla 8. Métrica de evaluación temperatura del motor — Mean squared error Stacking 1.....	30
Tabla 9. Métrica de evaluación temperatura del motor — Mean squared error Stacking 2.....	30
Tabla 10. Métrica de evaluación temperatura del motor — Root mean squared error Stacking 1.....	31
Tabla 11. Métrica de evaluación temperatura del motor — Root mean squared error Stacking 2.....	32
Tabla 12. Métrica de evaluación temperatura del motor — Mean absolute percentage error Stacking 1.	33
Tabla 13. Métrica de evaluación temperatura del motor — Mean absolute percentage error Stacking 2.	33
Tabla 14. Resumen de valores obtenidos por algoritmos y Stacking.....	35
Tabla 14. Operacionalización de variables.....	50
Tabla 15. Matriz de consistencia	32
Tabla 16. Métrica de evaluación temperatura del motor – R squared Árbol de decisión	86
Tabla 17. Métrica de evaluación temperatura del motor — Mean absolute error Árbol de decisión.....	86
Tabla 18. Métrica de evaluación temperatura del motor — Mean squared error Árbol de decisión	87

Tabla 19. Métrica de evaluación temperatura del motor — Root mean squared error Árbol de decisión.....	87
Tabla 20. Métrica de evaluación temperatura del motor — Mean absolute percentage error Árbol de decisión	87
Tabla 21. Métrica de evaluación temperatura del motor – R squared Regresión lineal.....	94
Tabla 22. Métrica de evaluación temperatura del motor — Mean absolute error Regresión lineal.....	94
Tabla 23. Métrica de evaluación temperatura del motor — Mean squared error Regresión lineal.....	94
Tabla 24. Métrica de evaluación temperatura del motor — Root mean squared error Regresión lineal	95
Tabla 25. Métrica de evaluación temperatura del motor — Mean absolute percentage error Regresión lineal	95
Tabla 26. Métrica de evaluación temperatura del motor – R squared ElasticNet	101
Tabla 27. Métrica de evaluación temperatura del motor — Mean absolute error ElasticNet	101
Tabla 28. Métrica de evaluación temperatura del motor — Mean squared error ElasticNet	101
Tabla 29. Métrica de evaluación temperatura del motor — Root mean squared error ElasticNet.....	102
Tabla 30. Métrica de evaluación temperatura del motor — Mean absolute percentage error ElasticNet.....	102
Tabla 31. Métrica de evaluación temperatura del motor – R squared KNN	108
Tabla 32. Métrica de evaluación temperatura del motor — Mean absolute error KNN.....	108
Tabla 33. Métrica de evaluación temperatura del motor — Mean squared error KNN.....	108
Tabla 34. Métrica de evaluación temperatura del motor — Root mean squared error KNN	109
Tabla 35. Métrica de evaluación temperatura del motor — Mean absolute percentage error KNN	109
Tabla 36. Métrica de evaluación temperatura del motor – R squared Lasso	114

Tabla 37. Métrica de evaluación temperatura del motor — Mean absolute error Lasso.....	115
Tabla 38. Métrica de evaluación temperatura del motor — Mean squared error Lasso.....	115
Tabla 39. Métrica de evaluación temperatura del motor — Root mean squared error Lasso	115
Tabla 40. Métrica de evaluación temperatura del motor — Mean absolute percentage error Lasso	116
Tabla 41. Métrica de evaluación temperatura del motor – R squared Random Forest Regressor.....	122
Tabla 42. Métrica de evaluación temperatura del motor — Mean absolute error Random Forest Regressor.	122
Tabla 43. Métrica de evaluación temperatura del motor — Mean squared error Random Forest Regressor.	122
Tabla 44. Métrica de evaluación temperatura del motor — Root mean squared error Random Forest Regressor.....	123
Tabla 45. Métrica de evaluación temperatura del motor — Mean absolute percentage error Random Forest Regressor.....	123
Tabla 46. Métrica de evaluación temperatura del motor – R squared Ridge.....	129
Tabla 47. Métrica de evaluación temperatura del motor — Mean absolute error Ridge	129
Tabla 48. Métrica de evaluación temperatura del motor — Mean squared error Ridge.....	129
Tabla 49. Métrica de evaluación temperatura del motor — Root mean squared error Ridge	130
Tabla 50. Métrica de evaluación temperatura del motor — Mean absolute percentage error Ridge	130
Tabla 51. Métrica de evaluación temperatura del motor – R squared XG Boost	136
Tabla 52. Métrica de evaluación temperatura del motor — Mean absolute error XG Boost.....	136
Tabla 53. Métrica de evaluación temperatura del motor — Mean squared error XG Boost	136

Tabla 54. Métrica de evaluación temperatura del motor — Root mean squared error XG Boost	137
Tabla 55. Métrica de evaluación temperatura del motor — Mean absolute percentage error XG Boost.....	137
Tabla 56. Datos_motor Raghavendra:	138

Índice de figuras

Figura 1. Diseño de investigación	18
Figura 2. Diagrama secuencial del modelo propuesto.	24
Figura 3. Resultados temperatura del motor según el indicador R squared- Temperatura del motor. Fuente: Elaboración propia	28
Figura 4. Resultados temperatura del motor según el indicador Mean absolute error- Temperatura del motor. Fuente: Elaboración propia	29
Figura 5. Resultados temperatura del motor según el Mean squared error - Temperatura del motor. Fuente: Elaboración propia	31
Figura 6. Resultados temperatura del motor según el Root mean squared error - Temperatura del motor. Fuente: Elaboración propia	32
Figura 7. Resultados temperatura del motor según el Mean absolute percentage error - Temperatura del motor. Fuente: Elaboración propia	34
Figura 08. Conexión con la base de datos	81
Figura 09.. Visualización de los atributos de la temperatura del motor	81
Figura 10. Visualización de la cantidad de filas y columns.....	81
Figura 11. Presentación de la cantidad de columns que tienen valores nulos ...	82
Figura 12. Verificación del valor perdido	82
Figura 13. Visualización de datos limpios para ser procesado por el algoritmo ...	82
Figura 14. Exploración de las variables.....	83
Figura 15. Selección de variables	83
Figura 16. Librerías para ejecutar el modelo	83
Figura 17. Correlación de variables.....	84
Figura 18. Visualización de Pair plots	84
Figura 19. Data particionada	85
Figura 20. Modelo Árbol de decisión Regressor.....	85
Figura 21. Modelo para Y1-temperatura del motor.....	85
Figura 22. Resultados de las métricas del modelo Árbol de decisión (Train)	85
Figura 23. Resultados de las métricas del modelo Árbol de decisión (Test)	86
Figura 24. Conexión con la base de datos	89
Figura 25. Visualización de los atributos de la temperatura del motor	89
Figura 26. Visualización de la cantidad de filas y columns.....	89
Figura 27. Presentación de la cantidad de columns que tienen valores nulos ...	90

Figura 28. Verificación del valor perdido	90
Figura 29. Visualización de datos limpios para ser procesado por el algoritmo ...	90
Figura 30. Exploración de las variables.....	91
Figura 31. Selección de variables	91
Figura 32. Librerías para ejecutar el modelo	91
Figura 33. Correlación de variables.....	92
Figura 34. Visualización de Pair plots	92
Figura 35. Data particionada	92
Figura 36. Modelo Regresión lineal.....	93
Figura 37. Modelo para Y1-temperatura del motor.....	93
Figura 38. Resultados de las métricas del modelo Regresión lineal (Train).....	93
Figura 39. Resultados de las métricas del modelo Regresión lineal (Test)	93
Figura 40. Conexión con la base de datos	96
Figura 41. Visualización de los atributos de la temperatura del motor	96
Figura 42. Visualización de la cantidad de filas y columnas.....	96
Figura 43. Presentación de la cantidad de columnas que tienen valores nulos ...	97
Figura 44. Verificación del valor perdido	97
Figura 45. Visualización de datos limpios para ser procesado por el algoritmo ...	97
Figura 46. Exploración de las variables.....	98
Figura 47. Selección de variables	98
Figura 48. Librerías para ejecutar el modelo	98
Figura 49. Correlación de variables.....	99
Figura 50. Visualización de Pair plots	99
Figura 51. Data particionada	99
Figura 52. Modelo ElasticNet	100
Figura 53. Modelo para Y1-temperatura del motor.....	100
Figura 54. Resultados de las métricas del modelo ElasticNet (Train)	100
Figura 55. Resultados de las métricas del modelo ElasticNet (Test)	100
Figura 56. Conexión con la base de datos	103
Figura 57. Visualización de los atributos de la temperatura del motor	103
Figura 58. Visualización de la cantidad de filas y columnas.....	103
Figura 59. Presentación de la cantidad de columnas que tienen valores nulos .	104
Figura 60. Verificación del valor perdido	104

Figura 61. Visualización de datos limpios para ser procesado por el algoritmo .	104
Figura 62. Exploración de las variables.....	105
Figura 63. Selección de variables	105
Figura 64. Librerías para ejecutar el modelo	105
Figura 65. Correlación de variables.....	106
Figura 66. Visualización de Pair plots	106
Figura 67. Data particionada	106
Figura 68. Modelo KNN Regressor	107
Figura 69. Modelo para Y1-temperatura del motor.....	107
Figura 70. Resultados de las métricas del modelo KNN (Train).....	107
Figura 71. Resultados de las métricas del modelo KNN (Test).....	107
Figura 72. Conexión con la base de datos	110
Figura 73. Visualización de los atributos de la temperatura del motor	110
Figura 74. Visualización de la cantidad de filas y columnas.....	110
Figura 75. Presentación de la cantidad de columnas que tienen valores nulos .	111
Figura 76. Verificación del valor perdido	111
Figura 77. Visualización de datos limpios para ser procesado por el algoritmo .	111
Figura 78. Exploración de las variables.....	112
Figura 79. Selección de variables	112
Figura 80. Librerías para ejecutar el modelo	112
Figura 81. Correlación de variables.....	112
Figura 82. Visualización de Pair plots	113
Figura 83. Data particionada	113
Figura 84. Modelo Lasso	113
Figura 85. Modelo para Y1-temperatura del motor.....	113
Figura 86. Resultados de las métricas del modelo Lasso (Train).....	114
Figura 87. <i>Resultados de las métricas del modelo Lasso (Test)</i>	114
Figura 88. Conexión con la base de datos	117
Figura 89. Visualización de los atributos de la temperatura del motor	117
Figura 90. Visualización de la cantidad de filas y columnas.....	117
Figura 91. Presentación de la cantidad de columnas que tienen valores nulos .	118
Figura 92. Verificación del valor perdido	118
Figura 93. Visualización de datos limpios para ser procesado por el algoritmo .	118

Figura 94. Exploración de las variables.....	119
Figura 95. Selección de variables	119
Figura 96. Librerías para ejecutar el modelo.....	119
Figura 97. Correlación de variables.....	120
Figura 98. Visualización de Pair plots	120
Figura 99. Data particionada	120
Figura 100. Modelo Random Forest.....	121
Figura 101. Modelo para Y1-temperatura del motor.....	121
Figura 102. Resultados de las métricas de Random Forest (Train)	121
Figura 103. <i>Resultados de las métricas de Random Forest (Test)</i>	121
Figura 104. Conexión con la base de datos	124
Figura 105. Visualización de los atributos de la temperatura del motor	124
Figura 106. Visualización de la cantidad de filas y columnas.....	124
Figura 107. Presentación de la cantidad de columnas que tienen valores nulos	125
Figura 108. Verificación del valor perdido	125
Figura 109. Visualización de datos limpios para ser procesado por el algoritmo	125
Figura 110. Exploración de las variables.....	126
Figura 111. Selección de variables	126
Figura 112. Librerías para ejecutar el modelo.....	126
Figura 113. Correlación de variables.....	127
Figura 114. Visualización de Pair plots	127
Figura 115. Data particionada	127
Figura 116. Modelo Ridge	128
Figura 117. Modelo para Y1-temperatura del motor.....	128
Figura 118. Resultados de las métricas del modelo Ridge (Train)	128
Figura 119. Resultados de las métricas del modelo Ridge (Test)	128
Figura 120. Conexión con la base de datos	131
Figura 121. Visualización de los atributos de la temperatura del motor	131
Figura 122. Visualización de la cantidad de filas y columnas.....	131
Figura 123. Presentación de la cantidad de columnas que tienen valores nulos	132
Figura 124. Verificación del valor perdido	132
Figura 125. Visualización de datos limpios para ser procesado por el algoritmo	132
Figura 126. Exploración de las variables.....	133

Figura 127. Selección de variables	133
Figura 128. Librerías para ejecutar el modelo	133
Figura 129. Correlación de variables.....	134
Figura 130. Visualización de Pair plots	134
Figura 131. Data particionada	134
Figura 132. Modelo XG Boost	135
Figura 133. Modelo para Y1-temperatura del motor.....	135
Figura 134. Resultados de las métricas del modelo XG Boost (Train)	135
Figura 135. Resultados de las métricas del modelo XG Boost (Train)	135
Figura 135. Conexión con la base de datos – Stacking 1	138
Figura 136. Visualización de los atributos de la temperatura del motor	139
Figura 137. Visualización de los atributos de la temperatura del motor	139
Figura 138. Visualización de la cantidad de filas y columnas.....	139
Figura 139. Verificación de valores perdidos.....	140
Figura 140. Visualización de datos limpios para ser procesado por el algoritmo	140
Figura 141. Exploración de las variables.....	141
Figura 142. Selección de variables	142
Figura 143. Librerías para ejecutar el modelo	142
Figura 144. Correlación de variables.....	143
Figura 145. Visualización de Pair plots	143
Figura 146. Data particionada	144
Figura 147. Librería de modelos.....	144
Figura 148. Modelo Stacking 1	145
Figura 149. Resultados de las métricas del modelo Stacking 1 (Train).....	145
Figura 150. Resultados de las métricas del modelo Stacking1 (Test).....	146
Figura 151. Conexión <i>con la base de datos</i> – Stacking 2	147
Figura 152. Visualización de los atributos de la temperatura del motor	147
Figura 153. Visualización de los atributos de la temperatura del motor	148
Figura 154. Visualización de la cantidad de filas y columnas.....	148
Figura 155. Verificación de valores perdidos.....	149
Figura 156. Visualización de datos limpios para ser procesado por el algoritmo	149
Figura 157. Exploración de las variables.....	150
Figura 158. Selección de variables	151

Figura 159. Librerías para ejecutar el modelo	151
Figura 160. Correlación de variables.....	152
Figura 161. Visualización de Pair plots	152
Figura 162. Data particionada	153
Figura 163. Librería de modelos.....	153
Figura 164. <i>Modelo Stacking 2</i>	154
Figura 165. Resultados de las métricas del modelo Stacking 2 (Train).....	155
Figura 166. <i>Resultados de las métricas del modelo Stacking2 (Test)</i>	155

Resumen

La presente investigación tuvo como objetivo aplicar un modelo híbrido basado en *Stacking* para predecir la temperatura del motor eléctrico Raghavendra, siendo de tipo aplicada, pre-experimental y de diseño experimental; la población estuvo conformada por 1,048,576 registros de temperatura de los motores eléctricos Raghavendra. La técnica de recolección de datos fue el análisis documental y el instrumento ficha de registro. Los resultados muestran que *Stacking 1* obtuvo los mejores valores de *R Squared* con 99.87%, *Mean absolute error* con 0.15, *Mean squared error* con 0.53, *Root mean squared error* con 0.73 y *Mean absolute percentage error* con 0.00282%. Concluyendo que *Stacking 1* fue el mejor algoritmo para predecir la temperatura del motor eléctrico Raghavendra.

Palabras clave: *Stacking*, *Machine learning*, temperatura, motor eléctrico, *Python*

Abstract

The aim of this research was to apply a hybrid model based on Stacking to predict the temperature of the Raghavendra electric motor, being of applied, pre-experimental and experimental design type; the population consisted of 1,048,576 temperature records of Raghavendra electric motors. The data collection technique was documentary analysis and the record card instrument. The results show that Stacking 1 obtained the best R2 Squared values with 99.87%, Mean absolute error with 0.15, Mean squared error with 0.53, Root mean squared error with 0.73 and Mean absolute percentage error with 0.00282%. Concluding that Stacking 1 was the best algorithm for predicting the temperature of the Raghavendra electric motor.

Keywords: Stacking, Machine learning, temperature, electric motor, Python

I. INTRODUCCIÓN

En este capítulo, se llevó a cabo un análisis exhaustivo del panorama actual en lo que respecta a los motores, específicamente a la temperatura de estos a nivel internacional. Este análisis se fundamentó en investigaciones previas realizadas por The World Energy Outlook y Acorn Industrial Services con sus respectivos datos estadísticos que sirvieron de base para el presente estudio. Asimismo, se presentaron los objetivos general y específicos de la investigación, junto con las hipótesis formuladas, y se explicó la justificación teórica, social y económica del estudio.

Los motores eléctricos se emplean ampliamente en una variedad de aplicaciones como vehículos eléctricos y sistemas de turbina, donde realizar la medición de la temperatura de los componentes internos de un motor eléctrico como el motor síncrono de imán permanente (PMSM), es necesario para garantizar su funcionamiento seguro (Hosseini, Shahbandegan y Akilan, 2022).

Según la investigación realizada por The World Energy Outlook (2021), se evidencia que, en el año 2019, los motores eléctricos representaron aproximadamente el 70% del consumo eléctrico en la industria de ingeniería eléctrica y electrónica (sistemas de motores eléctricos e iluminación del proceso de calor-electroquímica, siendo este de 6700 Twh), además que, en el sector terciario, los motores eléctricos desempeñaron un papel significativo en el consumo de energía, representando más del 40%.

Asimismo, Acorn Industrial Services (2021), señala que el sobrecalentamiento se identifica como el factor causante de aproximadamente el 55% de los fallos en el aislamiento, esto se debe a las elevadas temperaturas en el entorno operativo como en el caso de motores eléctricos, donde cada aumento de 10 grados centígrados en la temperatura de un motor provoca una reducción del 50% en su vida útil.

De acuerdo a un estudio realizado por The Association of Electrical and Mechanical Trades (2022), las principales entidades de normalización han llegado a la conclusión que el 30% de los problemas que afectan al motor se originan debido

a defectos en el aislamiento y el 60% de estas incidencias son ocasionadas por un excesivo calentamiento del motor.

Por lo tanto, este estudio se justificó teóricamente, ya que se fundamenta en la aplicación de técnicas de machine learning (ML), las cuales brinda ventajas a las empresas industriales como mantenimiento predictivo, optimización del rendimiento y detección de anomalías en tiempo real de los motores eléctricos, de manera oportuna. Además, se justificó en el aspecto social, porque la predicción de la temperatura de los motores es un problema que afecta a la gran mayoría de las empresas del sector (automóviles, maquinaria industrial y sistemas de climatización) e implica un consumo significativo de energía eléctrica, por lo que una predicción precisa de la temperatura de los motores es esencial para evitar el sobrecalentamiento, a través del desarrollo soluciones más efectivas para mantener la temperatura de los motores en niveles óptimos, y beneficiaría a las empresas en términos de ahorro de costos y eficiencia operativa, así como tiene un impacto positivo en la sociedad al reducir el desperdicio de energía, emisiones de carbono y promover un entorno más sostenible y ecológico. Finalmente, se justificó en el aspecto económico, ya que al hacer uso de técnicas de machine learning se va a poder predecir de manera temprana las temperaturas de los motores, lo que se reduciría costos de mantenimiento, fallos de motores, reparaciones costosas y reemplazo de equipos, esto debido al sobrecalentamiento de los motores.

La investigación se enfocó en la temperatura de los motores eléctricos, ya que en este sector industrial se ha identificado un desafío crítico, esto debido a que la eficiencia de un motor eléctrico está directamente relacionada con su temperatura de operación, donde un aumento de temperatura puede resultar en pérdida de eficiencia y disminución del rendimiento del motor (Krause y Krause, 2022), asimismo las altas temperaturas pueden acortar significativamente la vida útil de los componentes del motor, como los devanados del estator y del rotor, cabe resaltar que un sobrecalentamiento constante puede causar degradación del material y fallas prematuras (Toliyat y Kliman, 2018). Es así que, la temperatura del motor es un factor esencial para el rendimiento y la vida útil de estos. Sin embargo, la mayoría de las fábricas ha enfrentado dificultades para predecir con precisión la

temperatura de los motores eléctricos en funcionamiento, lo que ha llevado a problemas de sobrecalentamiento y fallos prematuros en los equipos, esto ocasionado por sobrecarga de trabajo (el motor trabaja por encima de su capacidad nominal), fricción excesiva de las piezas como rodamientos y ejes, corriente eléctrica elevada, problemas de refrigeración y bobinado, mala ventilación o humedad, lo que termina teniendo consecuencias negativas en el rendimiento del motor (menos eficiencia, desgaste de la vida útil del motor y fallos), riesgos para la seguridad en el entorno de trabajo (como situaciones peligrosas, ej.: incendios o daños a componentes cercanos), así como altos costos de reparación y reemplazo. Sobre la base de realidad problemática presentada se planteó el siguiente problema general: ¿Como un modelo híbrido basado en *Stacking* permitirá mejorar la predicción de la temperatura del motor eléctrico Raghavendra? A la vez, los problemas específicos de la investigación fueron los siguientes ¿Cómo un modelo híbrido basado en *Stacking* permitirá mejorar la exactitud del *R2 Squared* en la predicción de la temperatura del motor eléctrico Raghavendra?, ¿Cómo un modelo híbrido basado en *Stacking* permitirá mejorar la exactitud del *Mean absolute error* en la predicción de la temperatura del motor eléctrico Raghavendra?, ¿Cómo un modelo híbrido basado en *Stacking* permitirá mejorar la exactitud del *Mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra?, ¿Cómo un modelo híbrido basado en *Stacking* permitirá mejorar la exactitud del *Root mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra?, ¿Cómo un modelo híbrido basado en *Stacking* permitirá mejorar la exactitud del *Mean absolute percentage error* en la predicción de la temperatura del motor eléctrico Raghavendra?

En base a lo dicho con anterioridad, el presente estudio tuvo como objetivo general: Desarrollar un modelo híbrido basado en *Stacking* para mejorar la predicción de la temperatura del motor eléctrico Raghavendra.

A la vez, tuvo los siguientes objetivos específicos: determinar en qué porcentaje un modelo híbrido basado en *Stacking* permite mejorar la exactitud del *R2 squared* en la predicción de la temperatura del motor eléctrico Raghavendra; determinar en qué porcentaje un modelo híbrido basado en *Stacking* permite mejorar la exactitud del *Mean absolute error* en la predicción de la temperatura del motor eléctrico Raghavendra; determinar en qué porcentaje un modelo híbrido

basado en *Stacking* permite mejorar la exactitud del *Mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra; determinar en qué porcentaje un modelo híbrido basado en *Stacking* permite mejorar la exactitud del *Root mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra y; determinar en qué porcentaje un modelo híbrido basado en *Stacking* permite mejorar la exactitud del *Mean absolute percentage error* en la predicción de la temperatura del motor eléctrico Raghavendra.

La hipótesis general de la investigación fue: Un modelo híbrido basado en *Stacking* permite mejorar la predicción de la temperatura del motor Raghavendra. Las hipótesis específicas fueron: Un modelo híbrido basado en *Stacking* permite mejorar la exactitud del R^2 squared en la predicción de la temperatura del motor eléctrico Raghavendra, un modelo híbrido basado en *Stacking* permite mejorar la exactitud del Mean absolute error la temperatura del motor eléctrico Raghavendra, un modelo híbrido basado en *Stacking* permite mejorar la exactitud del Mean squared error en la predicción de la temperatura del motor eléctrico Raghavendra. un modelo híbrido basado en *Stacking* permite mejorar la exactitud del Root mean squared error en la predicción de la temperatura del motor eléctrico Raghavendra, un modelo híbrido basado en *Stacking* permite mejorar la exactitud del Mean absolute percentage error en la predicción de la temperatura del motor eléctrico Raghavendra.

II. MARCO TEÓRICO

En este capítulo, se describen los antecedentes, las teorías relacionadas de la variable independiente, dependiente, dimensiones e indicadores, que han sido obtenidos de las diferentes bases de datos indexadas como: *IEEE Xplore*, *ProQuest*, *ScienceDirect*, *Springer* y *Web of Science* de los últimos 5 años.

Thosar *et al.*, (2020), realizaron un estudio acerca de la temperatura del motor, con el objetivo de realizar predicciones utilizando modelos lineales sencillos que pueden ejecutarse en un hardware de baja potencia, para lo cual se utilizó como conjunto de datos, las distintas sesiones de mediciones, conteniendo como datos de entrada tensión, temperatura del ambiente, velocidad del motor, etc., y datos de salida a los componentes de temperatura, a la vez que se hizo uso del algoritmo Regresión Lineal con descenso de gradiente y ecuaciones normales, donde Regresión Lineal con descenso de gradiente alcanzó los mejores valores de *Mean squared error* (MSE) de las dimensiones, siendo estos: pm (Temperatura del imán permanente en °C) con 0.1423, stator_yoke (yugo del estator) con 0.0600, stator_winding (bobinado del estator) con 0.1170 y stator_tooth (diente del estator) con 0.1299, concluyendo que utilizando técnicas de machine learning como regresión con ecuaciones normales es factible para lograr resultados sólidos, lo que mejora el rendimiento, esto debido a su simplicidad y requisitos de pre-procesamiento notables, al tener una precisión y tiempo de ejecución más eficiente.

Como también, Devi *et al.*, (2021), realizaron un estudio acerca de la temperatura de un motor, con el objetivo de evaluar el rendimiento de las técnicas de machine learning para predecir la temperatura de motor, utilizando el lenguaje de programación *Python* y una población de 9,94,8071 datos de temperatura del motor con 12 atributos recolectados del *UCI Machine learning*. Asimismo, el conjunto de datos lo dividió en 80:20 para el conjunto de entrenamiento y prueba, se utilizó el algoritmo *Convolutional neural network (CNN)* con las siguientes capas de activación: *Sigmoid*, *Softmax*, *Softsign*, *Relu*, etc., donde el algoritmo CNN con la capa Relu alcanzó el mejor valor de MSE de $3.87735e-05$, *Mean absolute error* (MAE) de 0.00441429 y R squared (R^2) de 0.999054, concluyendo que las técnicas de Machine learning tienen grandes ventajas al aplicarlo para predecir la temperatura del motor.

En el estudio desarrollado por Hughes *et al.*,(2023), acerca de la temperatura del motor, cuyo objetivo fue comparar técnicas de *Machine Learning (ML)* para desarrollar modelos de evaluación de la temperatura del motor, cuyo conjunto de datos estuvo compuesta por 185 horas de grabaciones que abarcan más de 1,3 millones de muestras multidimensionales, asimismo se usó la metodología *Cross validation* con 10 repeticiones, el lenguaje de programación *Python*, y del algoritmo Ordinary least squares (OLS), el mismo que alcanzó un valor de MSE de 2.68, concluyendo que las técnicas de machine learning son eficientes, al reducir los errores de predicción y la sensibilidad de la calidad de los datos, a la vez de obtener puntuaciones y tiempo de entrenamiento excepcionalmente corto y capaces de operar 4 órdenes de magnitud más rápido que el tiempo real, generando pronósticos térmicos avanzados.

Al-Gabalawy, *et al.*,(2022), realizaron un estudio acerca de la temperatura del motor síncrono, cuya finalidad fue determinar qué elementos críticos tienen un impacto significativo en la temperatura del motor, haciendo uso de los siguientes algoritmos: Linear regression, Ridge regression, Polynomial regression, Xgboost, LM with interaction terms, Lasso regression, Polynomial regression with no outliers y *Support vector machine (SVM)*, así como el lenguaje de programación *Python* y *cross validation*, obteniendo como resultado que el algoritmo SVM alcanzó los mejores valores de: *Root mean squared error (RMSE)* es igual a 0.5898. Sin embargo, Polynomial regression with no outliers alcanzó el valor más alto de R^2 *Squared* igual a 0.682, concluyendo que el machine learning tiene un impacto significativo al predecir la temperatura del motor.

Czerwinski, Gęca y Kolano (2021), quienes realizaron un estudio acerca de la estimación de la temperatura sin sensores de un motor, cuyo objetivo fue proponer dos modelos para la estimación de la temperatura del devanado del motor BLDC, mediante métodos de aprendizaje automático, para lo cual tomaron en cuenta un conjunto de datos de 160 horas de mediciones de temperatura del motor BLDC obtenida de un banco de pruebas para un perfil de carga, velocidad de giro y condiciones de refrigeración, por lo que se usó la metodología *Cross validation* con 5 repeticiones, haciendo uso de los siguientes algoritmos: *ElasticNet*, *SVM* y *Stochastic Gradient Descent (SGD)*, donde los resultados obtenidos muestran que

el modelo 2, el cual incluye SGD fue el que alcanzó los mejores valores de las métricas establecidas: MSE igual a 0.35, RMSE con 0.59, no obstante SVM logró el mejor *Mean absolute percentage error* (MAPE) igual a 0.77 y *R Squared* con 0.998. Por lo que se concluye que las técnicas de machine learning demostraron ser eficientes para anticipar la temperatura del motor, además que puede detectar componentes de la maquinaria, evitando fallas del motor, daños y cortocircuitos.

De la misma manera, en el estudio realizado por Kirchgässner, Wallscheid y Böcker (2020), acerca de la estimación de la temperatura del motor eléctrico, este tuvo como objetivo comparar técnicas de machine learning para predecir temperaturas latentes en motores síncronos de imanes permanentes, se tomó en cuenta un conjunto de datos comprendido de 140 horas de mediciones de múltiples variables tomadas a una frecuencia de muestreo de 2 Hz de un motor eléctrico síncrono de imán permanente (PMSM) trifásico de 52 kW usado en aplicaciones automotrices, a la vez que se hizo uso de la metodología Cross validation, y de los algoritmos: *temporal convolutional network* (TCN) y *Recurrent neural network* (RNN), el cual TCN alcanzó un puntaje de MSE de 1.52, concluyendo que el enfoque propuesto es útil, ya que puede adaptarse fácilmente a distintos tipos de motores, objetivos relacionados con la temperatura (como componentes de motores), o incluso dominios (sistemas de baterías) sin requerir conocimiento en el área, siempre y cuando exista relación en las señales de entrada y salida.

A la vez que, Guo *et al.*,(2020), en su investigación sobre la temperatura del motor síncrono de imanes permanentes, tuvo como finalidad proponer un modelo informático para la predicción de la temperatura del motor síncrono de imanes permanentes (PMSM), tomando en cuenta como muestra 6,000 piezas del prototipo alemán del fabricante del equipo original, cuya plataforma de medición fue recolectada del departamento *Leistungselektronik und Elektrische Antriebstechnik* (LEA) de la Universidad de Paderborn, para lo cual se utilizó el lenguaje de programación *Python* y Cross validation con 5 repeticiones, usando como metodología el empleo de 6 técnicas de machine learning: *SVM*, *Ridge*, *AdaBoosting*, *DecisionTree*, *Random Forest*, DNN (modelo propuesto), cuyos resultados demuestran que DNN alcanzó los valores más óptimos de MAE de 0.1515, RMSE de 0.2368 y *R Squared* de 0.9439, concluyendo que el modelo

propuesto (DNN) demostró un rendimiento superior en comparación con los otros enfoques en lo que respecta a la predicción de la temperatura del devanado del estator en los motores PMSM, ofreciendo un soporte técnico para alertas de temperatura y funcionamiento seguro de estos motores.

Kirchgässner, Wallscheid y Böcker (2021), efectuaron una investigación de la temperatura del imán permanente en motores síncronos, cuyo objetivo fue evaluar varios modelos de aprendizaje automático en cuanto a su precisión de estimación para la tarea de predecir perfiles latentes de temperatura de imanes de alta dinámica, para esto se trabajó con conjunto de datos abarcó un total de 139 horas de grabaciones equivalente a un millón de registros multidimensionales, recopilado de datos de equipo de prueba de motores de alta calidad y los atributos de entorno, ambiente y temperaturas del refrigerante, donde se hizo uso de los siguientes técnicas de *Machine Learning*: *Support vector regression, randomized trees, OLS, neural networks, Multi Layer-perceptron (MLP)* y *k-Nearest neighbors*. Por lo que, los resultados muestran que MLP alcanzó los valores más óptimos de MAE de 1.32 y OLS alcanzó el mejor valor de MSE de 3.10; sin embargo, ambos algoritmos lograron un R Squared de 0.98. Concluyendo de esta manera que los algoritmos de machine learning (MLP y KNN) ayuda en la predicción de temperatura de los motores, marcando una gran calidad predictiva a nivel moderado.

En la Tabla 1 y Tabla 2, se presentan cuadros comparativos de los estudios previos acerca de las métricas y técnicas de ML en la predicción de la temperatura de los motores eléctricos, con el objetivo de realizar un estudio más profundo y detallado en este tema, que pueden ser útil para la presente tesis.

Tabla 1. Cuadro comparativo de las métricas de ML usadas en estudios previos para la predicción de la temperatura de los motores eléctricos.

AUTOR Y AÑO	MÉTRICAS				
	MSE	RMSE	MAE	MAPE	R Squared
Thosar <i>et al.</i> ,(2020)	X				
Devi <i>et al.</i> ,(2021)	X		X		X
Hughes <i>et al.</i> ,(2023)	X				
Al-Gabalawy, <i>et al.</i> ,(2022)		X			X
Czerwinski, Gęca y Kolano (2021)	X	X		X	X
Kirchgässner, Wallscheid y Böcker (2020)	X				
Guo <i>et al.</i> ,(2020)		X	X		X
Kirchgässner, Wallscheid y Böcker (2021)	X		X		X

Fuente: Elaboración propia.

Tabla 2. Cuadro comparativo de las técnicas usadas en estudios previos para la predicción de la temperatura de los motores eléctricos.

AUTOR Y AÑO	ALGORITMOS UTILIZADOS									
	Thosar <i>et al.</i> ,(2020)	Devi <i>et al.</i> ,(2021)	Hughes <i>et al.</i> ,(2023)	Al-Gabalawy, <i>et al.</i> ,(2022)	Czerwinski, Geça y Kolano (2021)	Kirchgässner, Wallscheid y Böcker (2020)	Guo <i>et al.</i> ,(2020)	Kirchgässner, Wallscheid y Böcker (2021)		
Regresión Lineal con descenso de gradiente	X									
Regresión Lineal con ecuaciones normales	X									
Convolutional neural network con capa de activación Sigmoid		X								
Convolutional neural network con capa de activación Softmax		X								
Convolutional neural network con capa de activación Relu		X								
Ordinary least squares (OLS)			X	X					X	
Linear regression				X						
Ridge regression				X			X			
Polynomial regression				X						
XGBoost				X						
LM with interaction terms				X						
Lasso regression				X						

Polynomial regression with no outliers	X			
Support vector machine (SVM)	X	X		X
ElasticNet		X		
Stochastic Gradient Descent (SGD)		X		
Temporal convolutional network (TCN)			X	
Recurrent neural network (RNN)			X	
AdaBoosting				X
DecisionTree				X
Random Forest				X
DNN				X
Support vector regression				X
Randomized trees				X
Neural networks				X
Multi-Layer-perceptron (MLP)				X
K-Nearest neighbors				X

Fuente: Elaboración propia.

A continuación, se presentan las teorías relacionadas con cada variable de estudio, contribuyendo al respaldo y fundamentación de la investigación:

La primera variable considerada fue un modelo híbrido basado en *Stacking*. Según Chatzimpampas, *et al.*, (2020), señala que *Stacking* es un método de conjunto que realiza una combinación de modelos heterogéneos, compuestos al menos por una capa, y luego utiliza otro meta-modelo, con el fin de hacer un resumen de predicción de estos algoritmos.

En este trabajo de investigación, también se hizo uso de cinco técnicas de *machine learning*, siendo estos: *Decision tree Regressor*, *Linear regression*, y *K-neighbors regressor (KNN-R)*, *Lasso*, *Random Forest Regressor*, *XG Boost regressor*.

Es así que, Veliz (2020), señala que el machine learning es un conjunto de métodos diseñados para crear modelos a partir de datos sin requerir una programación específica del problema, el cual tiene la capacidad de abordar tareas como predicción de nuevas instancias y explicación de resultados. Para Mahesh (2020), el ML es aquel campo científico enfocado en el uso de modelos estadísticos y algoritmos en sistemas informáticos para realizar tareas específicas sin requerir una programación detallada.

Taulli (2019), define al Árbol de decisión, como un algoritmo de clasificación / regresor, tanto para data no lineal utilizado en la manipulación de datos que no son de naturaleza numérica y para data de procedencia probabilística numérica, donde en su estructura, el punto de partida es el nodo raíz, y a partir de este nodo se ramifican diferentes caminos de decisión, conocidos como divisiones, cuyas probabilidades son calculadas para finalmente llegar a una hoja que representa el resultado final de las decisiones tomadas en el árbol. Para Wilmott (2022), los arboles de decisión representan una técnica de aprendizaje supervisado semejante a diagramas de flujo, aplicados tanto para tareas de clasificación como de regresión, el cual inicia con un conjunto de datos que están etiquetados y que poseen diversas características, para proceder a organizarlo jerárquicamente y dividirlos según los atributos relevantes.

Con respecto a Regresión lineal, Zhou (2021), señala que es un método de ML, que tiene como objetivo aprender un modelo lineal que pueda predecir con exactitud las etiquetas de salida de valor real, donde las variables discretas, se pueden convertir en variables de valor real cuando existe una relación ordinal entre los valores. A la vez, Burkov (2019), indica que la regresión lineal es un algoritmo de regresión ampliamente utilizado que desarrolla un modelo compuesto por una combinación lineal de características de datos de entrada, donde se selecciona el hiperplano de tal manera que se ubique lo más cercano posible a todos los datos de entrenamiento.

En relación a ElasticNet, este es un algoritmo de machine learning potente que combina las propiedades de Lasso y Ridge, realizado al agregar la función de pérdida estándar de mínimos cuadrados, usado para abordar desafíos como la multicolinealidad y el sobreajuste, así como problemas comunes en conjuntos de datos con muchas dimensiones (Sosnovshchenko, 2018).

Asimismo, KNN-Regressor, es un algoritmo conceptualmente fácil de entender y de aspecto intuitivo, que permite establecer los límites de decisión en todo el espacio en función de cualquier conjunto de entrenamiento, el cual se basa en 2 factores clave: la disponibilidad de una métrica de similitud eficaz para calcular las distancias entre cualquier par de objetos en el espacio, y en segundo lugar, la cantidad de muestras en el conjunto de entrenamiento, el cual garantiza una cobertura adecuada de todas las áreas del espacio (Jiang, 2021). El algoritmo k-NN puede utilizarse también en escenarios de regresión, una vez identificados los k vecinos de cada punto, en lugar de considerar su clase y establecer un sistema de votación, se considerará el valor que toma la etiqueta para cada uno de ellos y se devolverá como predicción el valor medio de dichos valores.

Por otra parte, Lasso, es un método de análisis de regresión que se basa en la minimización del error cuadrático empírico con un término de regularización que depende de la norma del vector de pesos, como en el caso de la regresión ridge (Mohri, 2018). Según Kuhn (2018), Lasso es un modelo que obtiene coeficientes de regresión que pueden tener un valor absoluto de 0, lo que conlleva a la regularización y a la selección de características de forma simultánea.

Con lo referente a *Random Forest*, este es un algoritmo que combina árboles predictores están correlacionadas, donde en cada paso de división se elige una muestra aleatoria de “m” predictores como candidatos para la división, a partir del conjunto completo de “p” predictores disponibles, sin embargo, cabe resaltar que, en cada etapa de división de un árbol, el algoritmo no considera la mayoría de los predictores disponibles (James y Witten, 2021). Para Watt, Borhani, y Katsaggelos (2020), *Random Forest* es un modelo de regresión que consta de un conjunto de árboles que se definen de manera recursiva, cuyo término “Random” se refiere tanto al hecho de que cada árbol utiliza una porción aleatoria de los datos originales como conjunto de entrenamiento, donde solo se considera un subconjunto al azar de las dimensiones de las características de entrada como opciones para las divisiones en cada nodo de los arboles resultantes.

Ridge-Regressor es una tecnica de machine learning utilizada generalmente cuando las variables independientes están altamente correlacionadas, el cual está técnica conserva todos los coeficientes, a diferencia de la regresión LASSO (Thevaraja, Rahman y Gabirial, 2019).

Con respecto a *XG Boost- regressor*, este es un conjunto de árboles de decisión, el cual el algoritmo construye una expansión aditiva de la función objetivo a través de la minimización de una función de perdida para regular la complejidad de dichos arboles (Bentéjac, Csörgő y Martínez-Muñoz, 2021).

Por último, *Python*, es un lenguaje de programación de alto nivel y versátil usado en una variedad de aplicación, el cual ofrece una amplia gama de estructuras de lenguaje y declaraciones que permiten representar datos y llevar a cabo operaciones en diversos tipos de información y su disponibilidad de miles de paquetes, bibliotecas y módulos de programación, a la vez que utiliza herramientas de machine learning como: NumPy, Matplotlib y TensorFlow (Wang, 2023).

Para Hunt (2019), este una interfaz web de codificación multiplataforma, lo que permite emplearlo en diferentes entornos como sistemas Windows, Linux o Mac de Apple, entre otros, además que es un lenguaje de programación de fácil comprensión, lo que lo convierte en una elección accesible, con un sólido respaldo en términos de bibliotecas gráficas.

De igual manera, entre las dimensiones de la variable modelo híbrido basado en *Stacking* se tomó en cuenta la *KDD (Knowledge Discovery in Databases)*, definido como un enfoque automático de análisis exploratorio y modelado de grandes conjuntos de datos, el cual abarca las siguientes etapas: Comprensión del dominio y objetivos de KDD, selección y adición, Preprocesamiento-Limpieza de datos, Transformación, Minería de datos, Evaluación e interpretación, Discovered Knowledge (Visualización e integración) (Maimon y Rokach, 2010).

A continuación, se detallan los conceptos, el cual engloba la segunda variable: Predicción de la temperatura del motor.

La temperatura del motor, se define como un indicador que mide la relación de la temperatura que abarca todo el motor, calculado mediante la relación entre la temperatura total en la boquilla y la temperatura total en la cara del compresor (Nasa, 2021). Asimismo, la temperatura del motor, es un indicador diseñado para medir la temperatura refrigerante en el motor, donde su función principal es señalar si el refrigerante se encuentra en un estado frío, dentro del rango normal o con sobrecalentamiento (YPFRuta, 2022).

Con respecto a la predicción, este es referido a la conjunción de aptitudes artísticas y científicas utilizadas para analizar patrones en acontecimientos tanto naturales como sociales, donde su propósito radica en estimar lo que podría suceder en circunstancias inciertas, es decir proporciona proyecciones acerca de eventos de los cuales no se puede asegurar su ocurrencia (Hardt y Recht, 2022).

Entonces, La predicción de la temperatura del motor se refiere a un proceso de análisis y estimación anticipada de la temperatura a base a datos históricos, parámetros de funcionamiento y modelos matemáticos, donde se consideran factores como temperatura total en la boquilla, temperatura total en la cara del compresor y el comportamiento refrigerante del motor, cuyo propósito de esta predicción es anticipar posibles sobrecalentamientos dentro del rango normal y prevenir fallos (Nasa, 2021; Hardt y Recht, 2022).

En el estudio, se llevará a cabo la predicción de la temperatura del motor eléctrico Raghavendra, utilizando cinco métricas de evaluación: *R Squared*, *Mean*

absolute error, Mean squared error, Root mean squared error y Mean absolute percentage error.

Los términos utilizados para las métricas de evaluación serán definidos de la siguiente manera:

Tabla 3. Términos de las métricas de evaluación

\hat{y}	Es el valor predicho de la muestra i-ésima
y_i	Es el valor verdadero correspondiente
\bar{y}	Es el promedio muestral
$P(ij)$	Es el valor predicho por el modelo individual i para el registro j (de n registros)
T_j	Es el valor objetivo para el registro j
$P(ij) = T_j$ $E_i = 0$	Por lo que, el índice E_i va de 0 a infinito, y 0 corresponde al valor ideal
\hat{r}_n	Es el índice de predicción
r_n	Es la calificación real en el conjunto de datos de prueba
N	Es el número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción
a	Son los datos reales
b	Son los datos resultantes
N	Es la cantidad de datos

Fuente: Elaboración propia.

Asimismo, a continuación, se proporciona definiciones detalladas de cada una de las métricas:

R-Squared (R^2)

Para Valbuena, et al (2019), el R-cuadrado es aquella métrica que muestra la relación entre la suma de los residuos al cuadrado y la suma total de cuadrados, el cual es expresado en la ecuación (1):

$$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (1)$$

Mean absolute error

de acuerdo con Vujović (2021), El Mean absolute error se define como el promedio de los valores absolutos de los errores de predicción individuales de todas las instancias de prueba, donde cada error de predicción corresponde a la diferencia del valor real y predicho, el cual es representada en la ecuación (2):

$$E_i = \frac{1}{n} \sum_{j=1}^n |P_{(ij)} - \sum_{j=1}^n |P_{(ij)} - T_j|| \quad (2)$$

Mean squared error

Según Joshi (2020), Es un estimador que evalúa la proximidad entre una línea de regresión con un conjunto de datos, es decir la distancia euclidiana, por lo que el MSE penaliza más un menor número de errores significativos (Valores atípicos) que un mayor número de errores pequeños, el cual es representado en la ecuación (3):

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2 \quad (3)$$

Root mean squared error

Para Hodson (2022), este representa la raíz cuadrada del error cuadrático medio, donde la introducción de la raíz no cambia las comparaciones de magnitud entre los modelos, expresado con las mismas unidades que los datos originales, por lo que representa el error promedio o “estándar” en caso de errores que siguen una distribución normal, y lo cual es formulado en la ecuación (4):

$$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}} \quad (4)$$

Mean absolute percentage error

Para Prayudani, *et al.*, (2019), es aquel indicador que evalúa el nivel de error en una predicción al contrastarla con los valores reales de una serie, y este se calcula al dividir los errores absolutos de cada periodo entre los valores reales de observación de este mismo periodo, expresada en la ecuación (5):

$$MAPE = \frac{\sum_{t=1}^n \left| \frac{a-b}{a} \right|}{n} * 100\% \quad (5)$$

III. METODOLOGÍA

3.1. Tipo y diseño de investigación

Tipo de Investigación: El estudio fue aplicada con alcance explicativo, ya que se enfoca en proporcionar una comprensión profunda de la situación, siendo su principal objetivo resolver los problemas identificados en el ámbito específico o empresa en particular (Hernández y Mendoza, 2018). En el presente estudio se buscó dar solución del problema observado en el sector industrial, con lo que respecta a la predicción de la temperatura del motor.

Diseño de Investigación: Se hizo uso del diseño experimental de tipo pre-experimental, y transversal. Según Ramos (2021), señala que el diseño pre-experimental es aquel que se caracteriza por llevar a cabo una intervención exclusivamente a un grupo específico en base a la hipótesis para observar su impacto. Por otra parte, para Arias (2021), un estudio transversal, es aquel tipo de estudio donde la información se da en un solo lapso de tiempo.

Es así que el presente estudio se enfocó en la temperatura del motor, el cual implicó la manipulación de la variable independiente (Modelo híbrido basado en *Stacking*) para observar cómo afecta a la variable dependiente (predicción de la temperatura del motor), y pre-experimental, dado que implicó una sola medición sin un grupo de control equivalente.



Figura 1. Diseño de investigación

Dónde:

G= Temperatura del motor Raghavendra

X= modelo híbrido basado en *Stacking*

O= Métricas de evaluación

3.2. Variables y Operacionalización

Variable 1: Modelo híbrido basado en *Stacking*

Definición conceptual

Stacking es un “método de conjunto que realiza una combinación de modelos heterogéneos, compuestos al menos por una capa, y luego utiliza otro meta-modelo, con el fin de hacer un resumen de predicción de estos algoritmos” (Chatzimpampas, et al., 2020).

Definición operacional

Para desarrollar el modelo híbrido basado en Stacking se tomó en cuenta los pasos de la metodología KDD, siendo estos: Comprensión del dominio y Objetivos de KDD, Selección y adición, Preprocesamiento-Limpieza de datos, Transformación, Minería de datos, Evaluación e interpretación, Discovered Knowledge (Visualización e integración) (Maimon y Rokach, 2010).

Variable 2: Predicción de la temperatura del motor

Definición conceptual

La predicción de la temperatura del motor se refiere a un proceso de análisis y estimación anticipada de la temperatura a base a datos históricos, parámetros de funcionamiento y modelos matemáticos, donde se consideran factores como temperatura total en la boquilla, temperatura total en la cara del compresor y el comportamiento refrigerante del motor, cuyo propósito de esta predicción es anticipar posibles sobrecalentamientos dentro del rango normal y prevenir fallos (Nasa, 2021; Hardt y Recht, 2022).

Definición operacional

Para medir los resultados del aprendizaje se emplearon fichas que incluyen las métricas de evaluación a estudiar, siendo estas: R^2 , MAE, MSE, RMSE y MAPE, las mismas que se obtuvieron mediante el uso de herramientas de machine learning.

La operacionalización de las variables consiste en un conjunto de técnicas y métodos que especifica cómo se va a medir cada variable en la investigación. (Arias, 2021). La operacionalización de variables se presenta en la Tabla 1 (Anexo N°1).

3.3. Población, muestra y muestreo

Población: Es aquel conjunto o totalidad referente a los elementos del cual se busca identificar sus características generales y específicas dependiendo de los objetivos del estudio (Cabezas, Andrade y Torres, 2018). La investigación estuvo compuesta por 1,048,576 registros de temperatura de los motores eléctricos Raghavendra, que fueron recolectados de la base de datos de kaggle.

Los criterios de selección tomados en cuenta para el estudio fueron los siguientes:

- **Criterios de inclusión:**
 - Motores eléctricos Raghavendra con variaciones en la temperatura
- **Criterios de exclusión:**
 - Motores eléctricos con buen funcionamiento
 - Motores eléctricos adquiridos recientemente

Muestra: La muestra es una porción representativa extraída de la población, siguiendo un plan de acción previamente definido (muestreo), es decir está compuesta por elementos que comparten características similares con la población (Ñaupas *et al.*, 2018). Para la muestra se tomó en cuenta una parte representativa del total de registros de temperatura de los motores eléctricos Raghavendras descrita en la población, esto porque los algoritmos generan resultados más óptimos cuando la cantidad de datos es mayor.

Unidad de análisis: Un motor con variaciones de temperatura.

3.4. Técnicas e instrumentos de recolección de datos

Técnicas de recolección de datos

Se empleó la siguiente técnica: Análisis documental. Para Medina *et al.*, (2023) el análisis documental representa un método que implica la revisión y evaluación sistemática de documentos escritos, como informes, registros y publicaciones, con el propósito de adquirir información y lograr una comprensión más profunda de un fenómeno o problema particular.

Instrumentos de recolección de datos

Para esto, los datos relacionados con la temperatura del motor se obtuvieron de la base de datos de Kaggle, disponible en: <https://www.kaggle.com/code/raghavendra18/raghavendra-electric-motor-dataset/log>, utilizando el instrumento ficha de registro que incluyó los indicadores de las variables de estudio.

Validación y confiabilidad

La validez, es definida por Hernández, Fernández y Baptista (2014), como “un instrumento de evaluación realizado considerando todos los tipos de evidencia disponibles, donde cuanto mayor sea la cantidad de pruebas de validez de contenido, criterio y constructo que respalden un instrumento, este se acercará mejor a sus variables que se intenta medir”. En la presente investigación, se determinó a través de la evaluación de expertos, el cual fue el docente de la experiencia curricular el encargado de evaluar la metodología utilizada en la creación del instrumento, a la vez que un experto con doctorado en ingeniería de sistemas revisó minuciosamente las preguntas y su relación con las variables y dimensiones.

3.5. Procedimientos

Para llevar a cabo este estudio, se siguieron una serie de pasos. En primer lugar, se identificó la problemática y se buscaron trabajos previos que se detallan en los antecedentes para respaldar y fortalecer el marco teórico. Luego, se realizó una revisión de la metodología, recopilando información de la población a través de la base de datos de kaggle con respecto a la temperatura del motor eléctrico Raghavendra. Posteriormente, se aplicó el modelo híbrido basado en Stacking, para lo cual el Stacking 1 estuvo

compuesto por el nivel 0: *Decision tree Regressor*, *Linear regression*, *ElasticNet* y *K-neighbors regressor* y nivel 1: *Random Forest Regressor*, así como el Stacking 2 estuvo compuesto por el nivel 0: *Lasso*, *Linear regression Ridge*, *XG Boost regressor* y nivel 1: *Random Forest Regressor*. Luego, se procesó los datos obtenidos en el lenguaje de programación *Python*. Finalmente, se vaciaron los datos y se procedió al análisis de los resultados, lo que permitió evaluar la precisión de los modelos de la temperatura del motor eléctrico Raghavendra y obtener conclusiones relevantes para el estudio.

3.6. Método de análisis de datos

Para la investigación, se adquirió la información de la base de datos de Kaggle, para luego depurar los datos, esto con el objetivo de eliminar datos redundantes, para lo cual no fue necesario la transformación de las variables, ya que los valores eran cuantitativos. De manera similar, se aplicó el método predictivo usando *Python*, el cual proporcionó datos relacionados con las métricas: *R squared*, *Mean absolute error*, *Mean squared error*, *Root mean squared error* y *Mean absolute percentage error*.

Finalmente, en este estudio, se emplearon herramientas como la matriz de confusión proporcionada por *Python* para evaluar el rendimiento de los algoritmos y el coeficiente Kappa de Cohen para confirmar el nivel de acuerdo y validar las hipótesis.

3.7. Aspectos éticos

Esta investigación estuvo regida en base a las normas establecidas por la Universidad César Vallejo, de la misma manera se respetó la autoría de fuentes tomadas en cuenta, siendo estas las revistas indexadas y repositorios reconocidos, igualmente aspectos indispensables del código de ética de la universidad. Los artículos 15° y 16° pertenecen a la política anti plagio y de derechos de autor alojados en la resolución RCUN°0470-2022-UCV. Asimismo, los artículos 37°, 42°, 44° del Código de Ética del Colegio de Ingenieros del Perú, en relación a divulgar u omitir a los autores o coautores del estudio.

Por último, el investigador se comprometió a respetar la fiabilidad de los datos recopilados, por lo que no realizó cambios que puedan alterar la prueba de hipótesis que se propone, de manera que el estudio demostró la calidad del mismo y ser usado como base teórica para investigaciones futuras.

IV. RESULTADOS

Para la obtención de los resultados del estudio nos basamos en nuestra propuesta que no ha sido realizado por ningún investigador en este campo de estudio la cual se visualiza en la Figura 2, en donde podemos observar la siguiente metodología KDD y nuestra propuesta con algoritmos de regresión:

- a) Obtención de datos: Datos de motor eléctrico Raghavendra.
- b) Limpieza de los datos: Eliminación de datos nulos, vacíos, blancos permitiendo una data legible.
- c) Preprocesamiento de los datos
- d) División de los datos: para el estudio se utilizó 80% para entrenamiento y 20% para validación (Devi *et al.*, (2021))
- e) Entrenamiento de los datos con los algoritmos de Decision Tree Regressor, ElasticNet, KNN - R, Lasso, RandomForestRegressor, Regresión Lineal, Ridge, Xgboost Regressor.
- f) Entrenamiento de nuestra propuesta 1 en la cual nuestro primer modelo consta de dos niveles en el primer nivel se encuentran los algoritmos Decision Tree Regressor, Regresión Lineal, ElasticNet, KNN - R y el segundo nivel el algoritmo RandomForest Regressor, en donde el primer nivel recibe 11 variables de entrada cada uno y luego cada algoritmo realiza una predicción para que luego estos valores que son 4 variables nuevas ingresen al segundo nivel RandomForest Regressor para mejorar la precisión, la cual todo el proceso constituye en un modelo híbrido basado en Stacking.
- g) Entrenamiento de nuestra propuesta 2 en la cual nuestro primer modelo consta de dos niveles en el primer nivel se encuentran los algoritmos Regresión Lineal, Lasso, Ridge, Xgboost Regressor y el segundo nivel el algoritmo Random Forest Regressor, en donde el primer nivel recibe 11 variables de entrada cada uno y luego cada algoritmo realice una predicción para que luego estos valores que son 4 variables nuevas ingresen al segundo nivel Random Forest Regressor para mejorar la precisión, la cual todo el proceso constituye en un modelo híbrido basado en Stacking.

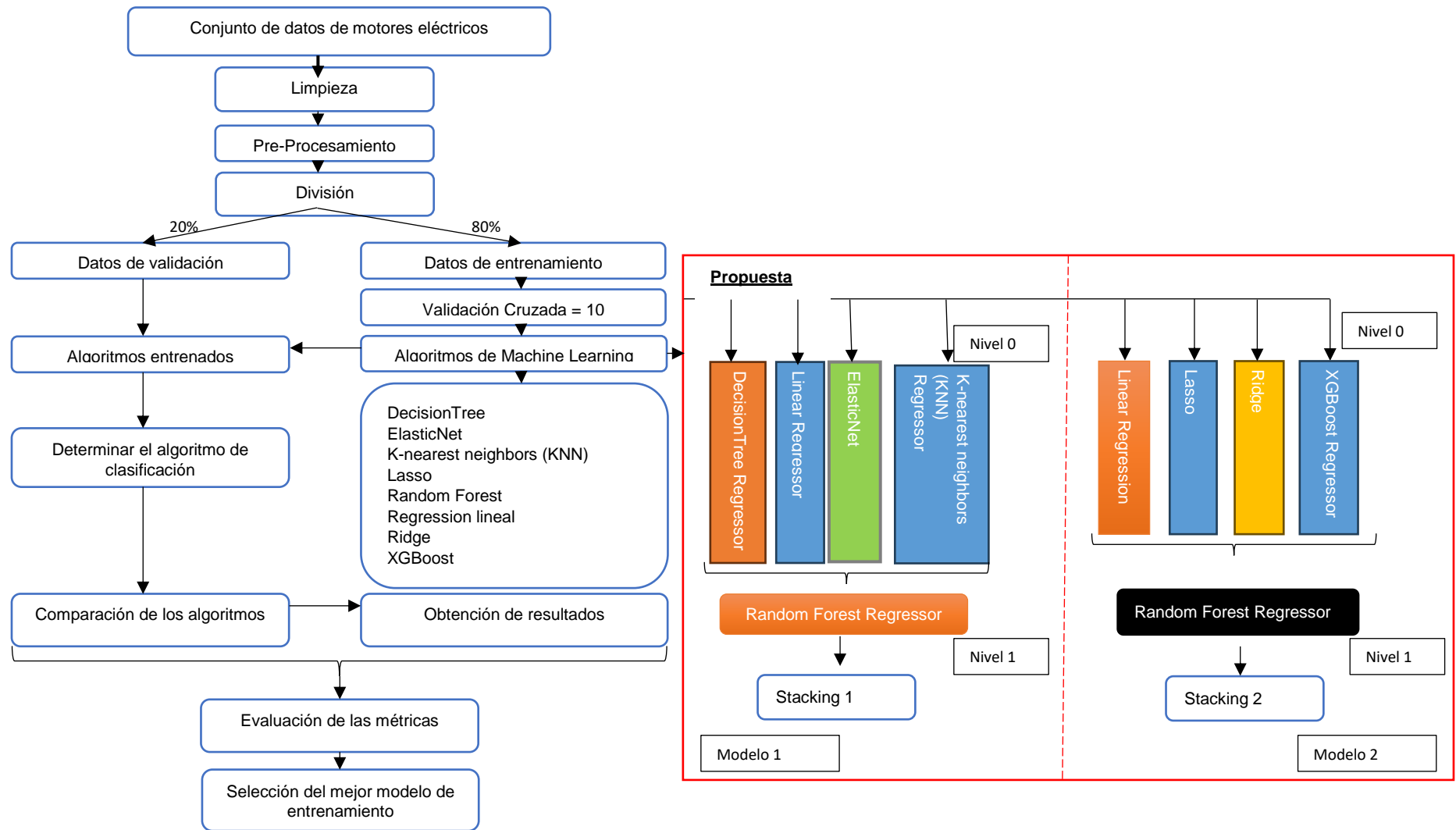


Figura 2. Diagrama secuencial del modelo propuesto.

Pseudo código de la propuesta “Un Modelo Híbrido Basado en Stacking para Mejorar la Predicción de la Temperatura del Motor Eléctrico Ravagheda”

1. Recolectar los datos de motores eléctricos
2. Realizar la limpieza de los datos
 - Eliminar datos faltantes
 - Manejar valores atípicos si es necesario
3. Realizar el pre-procesamiento de los datos
 - Normalizar o estandarizar los datos
 - Codificar variables categóricas si es necesario
4. Dividir la data en 80% para entrenar y 20% para validar
 - Dividir aleatoriamente los datos en conjuntos de entrenamiento y validación
5. Si la división es igual a entrenar, entonces:
 - Asignar validación cruzada = 10
 - Realizar experimentos con los modelos de:
 - Stacking 1
 - Stacking 2
 - DecisionTree
 - ElasticNet
 - K-nearest neighbors (KNN)
 - Lasso
 - Random Forest
 - Regression lineal
 - Ridge
 - XGBoost

6. Si la división es igual a validar, entonces:

- Entrenar los algoritmos:
 - Stacking 1
 - Stacking 2
 - DecisionTree
 - ElasticNet
 - K-nearest neighbors (KNN)
 - Lasso
 - Random Forest
 - Regression lineal
 - Ridge
 - XGBoost

7. Determinar la predicción de los algoritmos

- Obtener predicciones para los conjuntos de validación

8. Comparar las predicciones de los algoritmos

- Calcular métricas de rendimiento (por ejemplo, RMSE, MAE) para cada modelo

9. Obtener resultados

- Analizar y comparar los resultados de los modelos
- Seleccionar el modelo con el mejor rendimiento para futuras predicciones

OBJETIVO ESPECÍFICO 1: Determinar en qué porcentaje un modelo híbrido basado en Stacking permite mejorar la exactitud del *R squared* en la predicción de la temperatura del motor eléctrico Raghavendra.

Para la obtención de los resultados del objetivo se implementó nuestra propuesta que se encuentra detallado en el ANEXO N° 14 teniendo en cuenta nuestra metodología propuesta.

Stacking 1 (Random Forest Regressor) - Temperatura del motor

Tabla 4. Métrica de evaluación temperatura del motor – R squared Stacking 1

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	99.87%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *R squared* de 99.87% haciendo uso del *Stacking 1 (Random Forest Regressor)*.

Stacking 2 (Random Forest Regressor) - Temperatura del motor

Tabla 5. Métrica de evaluación temperatura del motor – R squared Stacking 2

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	98.68%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *R squared* de 98.68% haciendo uso del *Stacking 2 (Random Forest Regressor)*.

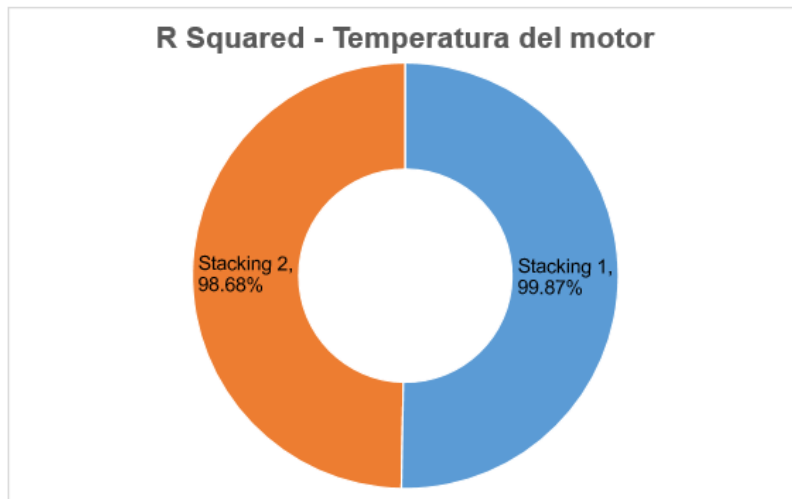


Figura 3. Resultados temperatura del motor según el indicador R squared- Temperatura del motor. Fuente: Elaboración propia

Interpretación: En la Figura 3 se evidencia que el modelo con mejor resultado en cuanto al indicador R squared que evalúa el porcentaje de temperatura del motor correctamente es “Stacking 1” con un 99.87%, asimismo sigue “Stacking 2” con un valor del 98.68%

OBJETIVO ESPECÍFICO 2: Determinar en qué porcentaje un modelo híbrido basado en Stacking permite mejorar la exactitud del *Mean absolute error* en la predicción de la temperatura del motor eléctrico Raghavendra.

Para la obtención de los resultados del objetivo se implementó nuestra propuesta que se encuentra detallado en el ANEXO N° 14 teniendo en cuenta nuestra metodología propuesta.

Stacking 1 (Random Forest Regressor) - Temperatura del motor

Tabla 6. Métrica de evaluación temperatura del motor — Mean absolute error Stacking 1

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	0.15

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean absolute error* de 0.15 haciendo uso del *Stacking 1 (Random Forest Regressor)*.

Stacking 2 (Random Forest Regressor) - Temperatura del motor

Tabla 7. Métrica de evaluación temperatura del motor — Mean absolute error Stacking 2

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	1.52

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean absolute error* de 1.52 haciendo uso del *Stacking 2 (Random Forest Regressor)*.

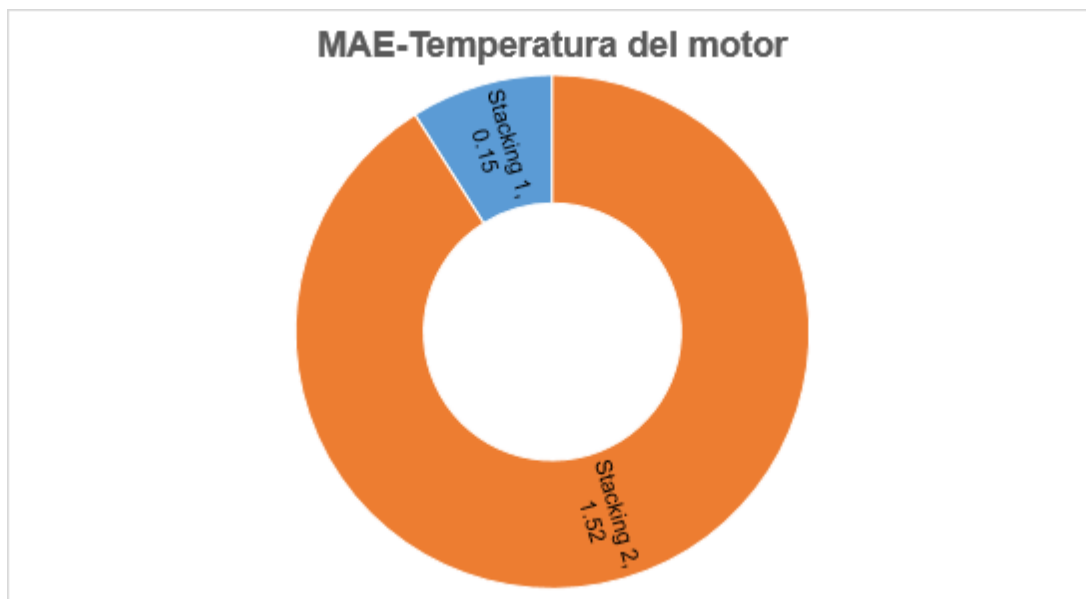


Figura 4. Resultados temperatura del motor según el indicador Mean absolute error- Temperatura del motor. Fuente: Elaboración propia

Interpretación: En la Figura 4 se evidencia que el modelo con mejor resultado en cuanto al Mean absolute error que evalúa el porcentaje de temperatura del motor correctamente es “*Stacking 1*” con 0.15, asimismo sigue “*Stacking 2*” con un valor de 1.52.

OBJETIVO ESPECÍFICO 3: Determinar en qué porcentaje un modelo híbrido basado en Stacking permite mejorar la exactitud del *Mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra.

Para la obtención de los resultados del objetivo se implementó nuestra propuesta que se encuentra detallado en el ANEXO N°14 teniendo en cuenta nuestra metodología propuesta.

Stacking 1 (Random Forest Regressor) - Temperatura del motor

Tabla 8. Métrica de evaluación temperatura del motor — Mean squared error Stacking 1

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	0.53

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean squared error* de 0.53 haciendo uso del *Stacking 1 (Random Forest Regressor)*.

Stacking 2 (Random Forest Regressor)- Temperatura del motor

Tabla 9. Métrica de evaluación temperatura del motor — Mean squared error Stacking 2

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	5.27

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean squared error* de 5.27 haciendo uso del *Stacking 2 (Random Forest Regressor)*.

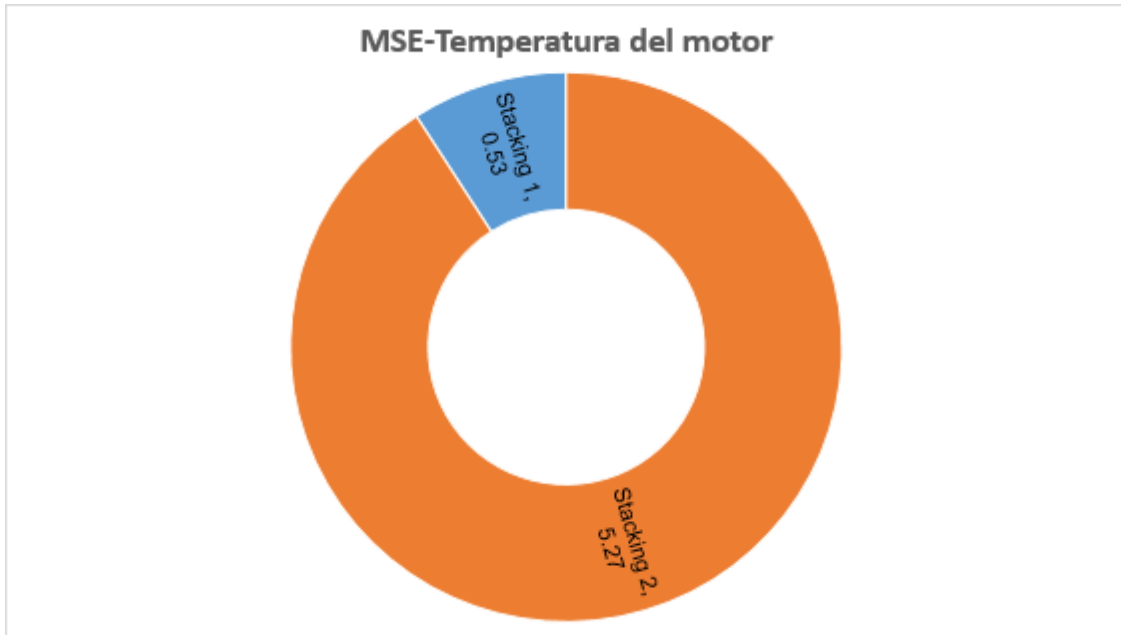


Figura 5. Resultados temperatura del motor según el Mean squared error - Temperatura del motor. Fuente: Elaboración propia

Interpretación: En la Figura 5 se evidencia que el modelo con mejor resultado en cuanto al Mean squared error que evalúa el porcentaje de temperatura del motor correctamente es “Stacking 1” con 0.53, asimismo sigue “Stacking 2” con un valor de 5.27.

OBJETIVO ESPECÍFICO 4: Determinar en qué porcentaje un modelo híbrido basado en Stacking permite mejorar la exactitud del *Root mean squared error* en la predicción de la temperatura del motor eléctrico Raghavendra.

Para la obtención de los resultados del objetivo se implementó nuestra propuesta que se encuentra detallado en el ANEXO N° 14 teniendo en cuenta nuestra metodología propuesta.

Stacking 1 (Random Forest Regressor)- Temperatura del motor

Tabla 10. Métrica de evaluación temperatura del motor — Root mean squared error Stacking 1.

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	0.73

Fuente: Elaboración propia.

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Root mean squared error* de 0.73 haciendo uso del *Stacking 1 (Random Forest Regressor)*.

Stacking 2 (Random Forest Regressor) - Temperatura del motor

Tabla 11. Métrica de evaluación temperatura del motor — Root mean squared error Stacking 2

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	2.29

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Root mean squared error* de 2.29 haciendo uso del *Stacking 2 (Random Forest Regressor)*.

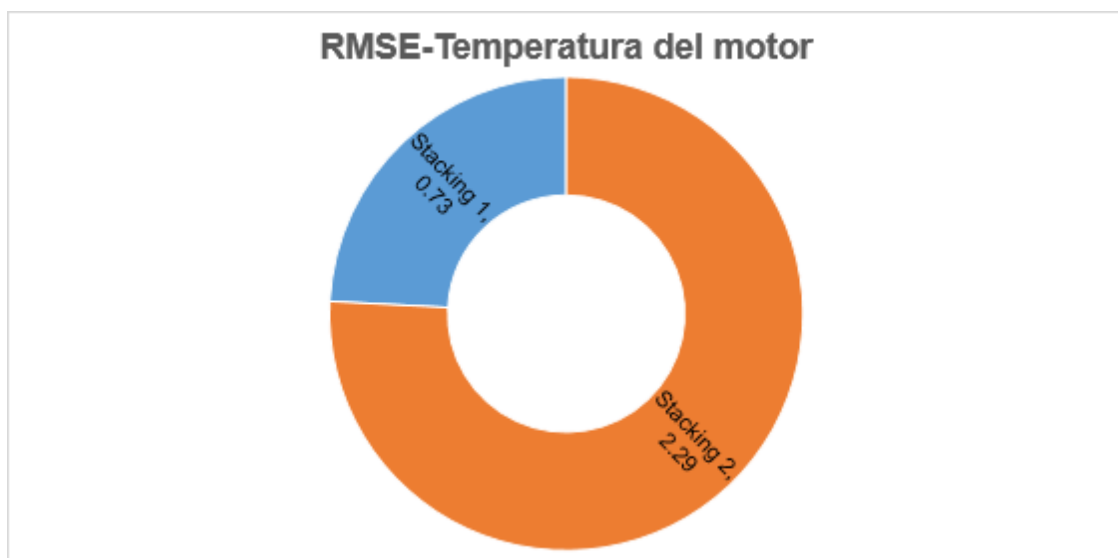


Figura 6. Resultados temperatura del motor según el Root mean squared error - Temperatura del motor. Fuente: Elaboración propia

Interpretación: En la Figura 6 se evidencia que el modelo con mejor resultado en cuanto al Root mean squared error que evalúa el porcentaje de temperatura del motor correctamente es “*Stacking 1*” con 0.73, asimismo sigue “*Stacking 2*” con un valor de 2.29.

OBJETIVO ESPECÍFICO 5: Determinar en qué porcentaje un modelo híbrido basado en Stacking permite mejorar la exactitud del *Mean absolute percentage error* en la predicción de la temperatura del motor eléctrico Raghavendra.

Para la obtención de los resultados del objetivo se implementó nuestra propuesta que se encuentra detallado en el ANEXO N° 14 teniendo en cuenta nuestra metodología propuesta.

Stacking 1 (Random Forest Regressor)- Temperatura del motor

Tabla 12. Métrica de evaluación temperatura del motor — Mean absolute percentage error Stacking 1.

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	0.00282%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean absolute percentage error* de 0.00282% haciendo uso del *Stacking 1 (Random Forest Regressor)*.

Stacking 2 (Random Forest Regressor)- Temperatura del motor

Tabla 13. Métrica de evaluación temperatura del motor — Mean absolute percentage error Stacking 2.

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	0.03%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de *Mean absolute percentage error* de 0.02786% haciendo uso del *Stacking 2 (Random Forest Regressor)*.

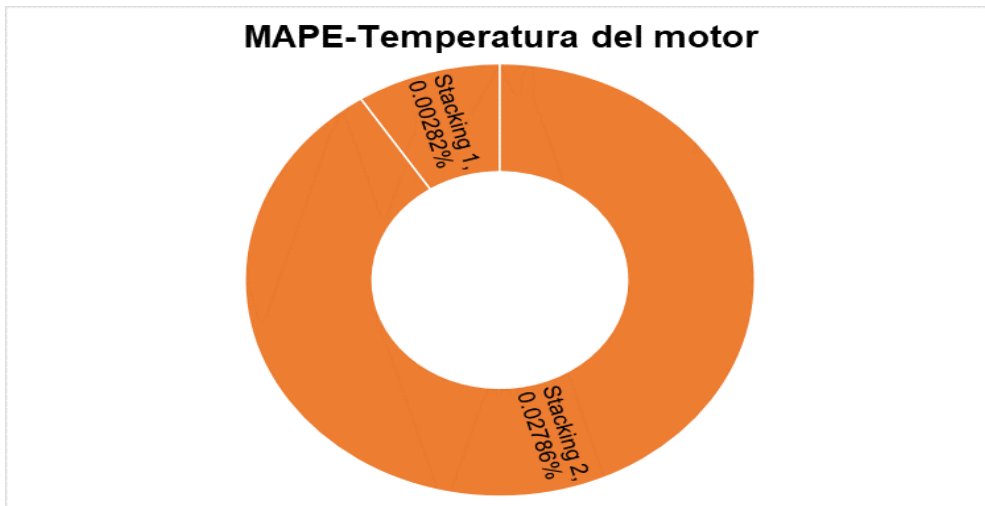


Figura 7. Resultados temperatura del motor según el Mean absolute percentage error - Temperatura del motor. Fuente: Elaboración propia

Interpretación: En la Figura 7 se evidencia que el modelo con mejor resultado en cuanto al Mean absolute percentage error que evalúa el porcentaje de temperatura del motor correctamente es "Stacking 1" con 0.00282%, asimismo sigue "Stacking 2" con un valor de 0.02786%.

V. DISCUSIÓN

Tabla 14. Resumen de valores obtenidos por algoritmos y Stacking.

TECNICAS ML	METRICAS					
	R^2	MAE	MSE	RMSE	MAPE	
	R^2	Mean absolute error	Mean squared error	Root mean squared error	Mean absolute percentage error	
Decision Tree Regressor	0.9993848024	0.0985602469	0.2469884159	0.4969792913	0.0019253455	
MODELO 1	Linear Regressor	0.8540481799	5.2522606365	50.9805783191	7.1400685094	0.1039808110
	ElasticNet	0.8355663377	5.3603003705	54.8151877096	7.4037279603	0.1065152776
	K-nearest neighbors (KNN) Regressor	0.9733413770	1.1765129797	10.4551060716	3.2334356452	0.0206207813
	Stacking 1	0.9986874304	0.1514259515	0.5265245326	0.7256201021	0.0028200483
	Linear Regressor	0.8540481799	5.2522606365	50.9805783191	7.1400685094	0.1039808110
MODELO 2	Lasso	0.8396783275	5.3228965080	53.7736449165	7.3330515419	0.1056951047
	Ridge	0.8540481544	5.2522602614	50.9805784243	7.1400685168	0.1039808037
	XGBoost Regressor	0.99246030720	1.18974601391	2.98163433304	1.72674095713	0.02225452655
	Stacking 2	0.9867996080	1.5206088153	5.2668806400	2.2949685488	0.0278586400

Fuente: Elaboración propia.

En esta sección, se presentan los resultados obtenidos en el estudio, el cual nos basamos en nuestra propuesta que no ha sido realizado por ningún investigador en este campo de estudio, para lo cual luego se realizó la comparación con los estudios del estado del arte, el cual incluyó las métricas de “*R Squared*”, “*Mean absolute error (MAE)*”, “*Mean squared error (MSE)*”, “*Root mean squared error (RMSE)*” y “*Mean absolute percentage error (MAPE)*”, cuyas similitudes y discrepancias con las investigaciones analizadas previamente y descritas en el marco teórico se presentan a continuación:

De acuerdo con el primer objetivo específico, el modelo híbrido basado en *Stacking* que brinda el mejor *R Squared* para predecir la temperatura del motor eléctrico Raghavendra es *Stacking 1* con 99.87%. Este es respaldado con la investigación de Devi *et al.*, (2021), el cual realizaron un estudio acerca de la temperatura de un motor, con el objetivo de evaluar el rendimiento de las técnicas de machine learning para predecir la temperatura de motor, donde se observa que CNN usado con la capa Relu alcanza el mejor valor de *R squared* de 99.91%. Concluyendo que las técnicas de Machine learning tienen grandes ventajas al aplicarlo para predecir la temperatura del motor.

De la misma manera, es corroborado por la investigación de Czerwinski, Gęca y Kolano (2021), quienes hicieron un estudio acerca de la estimación de la temperatura sin sensores de un motor, con el objetivo de proponer dos modelos para la estimación de la temperatura del devanado del motor BLDC mediante métodos de aprendizaje automático, cuyo algoritmo que logró el mejor resultado de *R Squared* de 99.80% fue SVM, el cual concluyeron que las técnicas de machine learning demostraron ser eficientes para anticipar la temperatura del motor, además que puede detectar componentes de la maquinaria, evitando fallas del motor, daños y cortocircuitos.

En base a esto, Valbuena, et al (2019), señala que el R-cuadrado es aquella métrica que muestra la relación entre la suma de los residuos al cuadrado y la suma total de cuadrados.

De igual manera, en el segundo objetivo específico, de acuerdo a los resultados obtenidos, el modelo híbrido basado en Stacking que presenta el mejor *Mean Absolute Error (MAE)*, es decir el promedio de todos los errores absolutos para predecir la temperatura del motor eléctrico Raghavendra, es *Stacking 1* de 0.15. Estos hallazgos se confirman con la investigación de Guo *et al.*,(2020), donde los autores realizaron una investigación sobre la temperatura del motor síncrono de imanes permanentes, con la finalidad de proponer un modelo informático para la predicción de la temperatura del motor síncrono de imanes permanentes (PMSM), donde DNN fue la técnica que alcanzó el mejor valor de MAE de 0.15, concluyendo que el modelo propuesto (DNN) demostró un rendimiento superior en comparación con los otros enfoques, lo que ofrece un soporte técnico para alertas de temperatura y funcionamiento seguro de estos motores.

A la vez, también es confirmado por el estudio de Kirchgässner, Wallscheid y Böcker (2021), donde los autores realizaron una investigación de la temperatura del imán permanente en motores síncronos, cuyo objetivo fue evaluar varios modelos de aprendizaje automático en cuanto a su precisión de estimación para la tarea de predecir perfiles latentes de temperatura de imanes de alta dinámica, por lo que MLP alcanzó el valor más óptimo de MAE de 1.32, cuya conclusión fue que

las técnicas de machine learning aporta en el pronóstico de la temperatura de los motores.

Es así que, Vujović (2021), indica que el MAE se define como el promedio de los valores absolutos de los errores de predicción individuales de todas las instancias de prueba, donde cada error de predicción corresponde a la diferencia del valor real y predicho.

En relación con el tercer objetivo, el modelo híbrido basado en Stacking que ofrece el mejor Mean Squared Error (MSE) para predecir la temperatura del motor eléctrico Raghavendra, es *Stacking 1* de 0.53. Estos hallazgos son respaldados por la investigación de Thosar *et al.*,(2020), donde dichos autores realizaron un estudio acerca de la temperatura del motor, con el objetivo de realizar predicciones utilizando modelos lineales sencillos que pueden ejecutarse en un hardware de baja potencia, entre los resultados se muestra que Regresión lineal con descenso de gradiente obtuvo el mejor valor de MSE de 0.14, concluyendo que al usar ecuaciones junto con modelos de aprendizaje automático mejora el rendimiento de estos con resultados notables.

Por otra parte, es refutado por el estudio de Kirchgässner, Wallscheid y Böcker (2020), en su estudio acerca de la estimación de la temperatura del motor eléctrico, este tuvo como objetivo comparar algoritmos de machine learning para predecir temperaturas latentes en motores síncronos de imanes permanentes, dentro de sus hallazgos se observó que Temporal Convolutional Network (TCN) logró un MSE de 1.52, donde los autores concluyeron que el modelo que propusieron es útil, esto debido a que se adapta fácilmente a todo tipo de motor como también dominios, sin la necesidad de tener conocimiento en el área.

De forma similar, también es contrastado por el estudio de Hughes *et al.*,(2023), cuyo estudio acerca de la temperatura del motor tuvo como objetivo comparar algoritmos de *Machine Learning (ML)* para desarrollar modelos de evaluación de la temperatura del motor, donde se evidenció que OLS alcanzó un MSE de 2.68, concluyendo que las técnicas de aprendizaje automático son importantes para la reducción de errores al momento de predecir la calidad de la información, al ser capaz de funcionar hasta 4 órdenes de magnitud más rápido, es

así que partiendo de esto, Joshi (2020), señala que esta métrica es un estimador que evalúa la proximidad entre una línea de regresión con un conjunto de datos, es decir la distancia euclidiana, por lo que el MSE penaliza más un menor número de errores significativos (Valores atípicos) que un mayor número de errores pequeños.

En cuanto al cuarto objetivo específico, el modelo híbrido basado en Stacking que ofrece el mejor *Root Mean Squared Error (RMSE)* para predecir la temperatura del motor eléctrico Raghavendra, es *Stacking 1* de 0.73. Este resultado se refuta con el trabajo de investigación de Al-Gabalawy, *et al.*, (2022), cuyos autores realizaron un estudio acerca de la temperatura del motor síncrono, cuya finalidad fue determinar qué elementos críticos tienen un impacto significativo en la temperatura del motor, en os hallazgos se pudo observar que SVM logró un RMSE de 0.59, el cual concluyeron que el aprendizaje automático tiene una influencia significativa al momento de predecir la temperatura del motor.

Por lo que, Hodson (2022), indica que el RMSE representa la raíz cuadrada del error cuadrático medio, donde la introducción de la raíz no cambia las comparaciones de magnitud entre los modelos, expresado con las mismas unidades que los datos originales, por lo que representa el error promedio o “estándar” en caso de errores que siguen una distribución normal.

Por último, en el quinto objetivo específico, el modelo híbrido basado en *Stacking* que evalúa completamente el *Mean Absolute Percentage Error (MAPE)* para predecir la temperatura del motor eléctrico Raghavendra, es *Stacking 1* de 0.00282%. Estos hallazgos se refutan con la investigación de Czerwinski, Geça y Kolano (2021), donde el autor realizó un estudio acerca de la estimación de la temperatura sin sensores de un motor, cuyo objetivo fue proponer dos modelos para la estimación de la temperatura del devanado del motor BLDC mediante métodos de aprendizaje automático, el cual muestra que el algoritmo SVM alcanzó el mejor valor de MAPE, siendo este de 0.77%, concluyendo que las técnicas de machine learning demostraron ser eficientes para anticipar la temperatura del motor, además que puede detectar componentes de la maquinaria, evitando fallas del motor, daños y cortocircuitos.

Por otra parte, Prayudani, *et al.*, (2019), señala que el MAPE es aquel indicador que evalúa el nivel de error en una predicción al contrastarla con los valores reales de una serie, y este se calcula al dividir los errores absolutos entre los valores reales de observación de cada periodo.

A través de la metodología utilizada, se lograron obtener datos significativos en relación con la información recopilada para la investigación, dado que se trata de una investigación de tipo aplicada con alcance explicativo, se buscó dar solución del problema observado en el sector industrial, con lo que respecta a la predicción de la temperatura del motor. La elección de un estudio pre experimental y transversal resultó beneficioso, ya que al enfocarnos en la temperatura del motor implicó la manipulación de la variable independiente (Modelo híbrido basado en *Stacking*) para observar cómo afecta a la variable dependiente (predicción de la temperatura del motor eléctrico Raghavendra), lo cual los datos se obtuvieron de la base de datos de Kagge, facilitando así la recolección de datos mediante fichas de registros con los indicadores de las variables de estudio para su posterior análisis a través del lenguaje de programación *Python*.

Esta investigación espera alcanzar una alta relevancia social y científica por la ampliación de conocimientos que aporta el modelo híbrido basado en *Stacking* en la mejora de la predicción de la temperatura del motor eléctrico Raghavendra, fomentando la utilización de enfoques de este tipo para generar beneficios tanto en el ámbito en que se desarrolla el estudio, como beneficios sociales y económicos.

VI. CONCLUSIONES

A continuación, se presentan las conclusiones del estudio:

1. Después de realizar el experimento, se pudo determinar que el modelo híbrido basado en Stacking que ofrece el mejor *R Squared* para predecir la temperatura del motor eléctrico Raghavendra, es Stacking 1=99.87%.
2. El modelo híbrido basado en Stacking que ofrece el mejor *Mean Absolute Error* (MAE) para predecir la temperatura del motor eléctrico Raghavendra, es Stacking 1= 0.15.
3. Según los resultados obtenidos, se pudo establecer que el modelo híbrido basado en Stacking que presenta el mejor *Mean Squared Error* (MSE), es decir, el promedio de errores elevados al cuadrado para predecir la temperatura del motor eléctrico Raghavendra, es Stacking 1= 0.53.
4. Con respecto al cuarto objetivo específico, se pudo establecer que el modelo híbrido basado en Stacking que proporciona la mejor *Root Mean Squared Error* (RMSE)-es decir, la desviación estándar de los valores residuales para predecir la temperatura del motor eléctrico Raghavendra, es Stacking 1= 0.73.
5. En relación con el quinto objetivo específico, se pudo determinar que el modelo híbrido basado en Stacking que ofrece el mejor *Mean Absolute Percentage Error* (MAPE) para predecir la temperatura del motor eléctrico Raghavendra, es Stacking 1= 0.00%.
6. Por último, se puede afirmar que Stacking 1 permite predecir con porcentajes elevados el R Squared, MAE, MSE, RMSE y MAPE la temperatura del motor eléctrico Raghavendra con eficacia.

VII. RECOMENDACIONES

Se propone la elaboración de un modelo predictivo con una metodología diferente al empleado en esta investigación, como son Semma y Crisp, con la finalidad de comparar y determinar cuál de estas es la mejor en el desarrollo de la predicción de la temperatura del motor eléctrico y minimizar las complicaciones referentes al proceso.

Se recomienda realizar estudios más exhaustivos sobre la temperatura del motor eléctrico, esto porque en el país no existe ningún estudio, y en otros países existe una carencia de investigaciones con respecto a este tema.

Se sugiere explorar otros tipos de algoritmos y métodos que faciliten el análisis de datos de entrada, permitiendo la predicción de los datos de salida dentro de un rango aceptable en relación con la predicción de la temperatura del motor eléctrico. Entre los modelos a considerarse se encuentra cascading, Decision Tree J48 O C4.5, Adaboost, Gradient Boosting, etc.

En futuras investigaciones, se sugiere extender el alcance del estudio realizado considerando una división de datos de 60% para entrenamiento y un 40% para la validación, considerando la implementación de un sistema inteligente que, al recibir datos históricos y realizar predicciones, pueda seguir identificando nuevos patrones y de este modo, perfeccionar su precisión con el tiempo.

REFERENCIAS

- Arias, José Y Covinos, Mitsuo. Diseño y Metodología de la investigación [en línea]. 1.a ed. Arequipa: Enfoques Consulting EIRL, 2021 [fecha de consulta: 25 de noviembre de 2023]. Disponible en: <http://repositorio.concytec.gob.pe/handle/20.500.12390/2260>
ISBN: 978-612-48444-2-3.
- Arias, José. Guía para elaborar la operacionalización de variables. Espacio I+ D, Innovación más desarrollo [en línea]. Octubre 2021, n.º 28. [Fecha de consulta: 25 de noviembre de 2023]. Disponible en <https://espacioimasd.unach.mx/index.php/Inicio/article/view/274>
ISSN: 2007-6703.
- Arias, José Y Covinos, Mitsuo. Diseño y Metodología de la investigación [en línea]. 1.a ed. Arequipa: Enfoques Consulting EIRL, 2021 [fecha de consulta: 25 de noviembre de 2023]. Disponible en: <http://repositorio.concytec.gob.pe/handle/20.500.12390/2260>
ISBN: 978-612-48444-2-3.
- Acorn Industrial Services (July 28, 2021), Electric Motor Problems And Solutions. England :). [Accessed on November 25, 2023]. Retrieved from <https://www.acorn-ind.co.uk/insight/electric-motor-problems-and-solutions/#:~:text=Overheating%20is%20generally%20caused%20by,life%20is%20reduced%20by%20half>.
- Angelos Chatzimpampas [et al]. Stackgenvis: Alignment of data, algorithms, and models for stacking ensemble learning using performance metrics by IEEE Transactions on Visualization and Computer Graphics [online]. October 2020, n.o.2.[Date of consultation: December 02, 2023]. Available in <https://ieeexplore.ieee.org/abstract/document/9222343>
ISSN: 1941-0506.
- Bentéjac, Candice, Csörgő, Anna And Martínez, Gonzalo. A comparative analysis of gradient boosting algorithms. Artificial Intelligence Review [online]. August 2021, n.o 1. [Date of consultation: November 25, 2023]. Available in <https://link.springer.com/article/10.1007/s10462-020-09896-5>.

Burkov, Andriy. The hundred-page machine learning book [online]. 1.a ed. Canada: Andriy Burkov, 2019 [Date of consultation: November 25, 2023]. Available in: <https://www.countrybookshelf.com/book/9781999579517> ISBN: 978-1999579500.

Cabezas, Edison, Andrade, Diego Y Torres, Johana. Introducción a la metodología de la investigación científica [en línea]. 1.a ed. Ecuador: Comisión Editorial de la Universidad de las Fuerzas Armadas ESPE, 2028 [fecha de consulta: 25 de octubre de 2022]. Disponible en: <https://repositorio.espe.edu.ec/bitstream/21000/15424/1/Introduccion%20a%20la%20Metodologia%20de%20la%20investigacion%20cientifica.pdf> ISBN: 978-9942-765-44-4.

Czerwinski, Dariusz, Gęca, Jakub And Kolano, Krzysztof. Machine learning for sensorless temperature estimation of a BLDC motor. Sensors [online]. July 2021, n.o 14. [Date of consultation: November 25, 2023]. Available in <https://www.mdpi.com/1424-8220/21/14/4655> ISSN: 1424-8220.

Devi Shyamala [et al]. Convolutional Neural Network Enactment Layer based Ambient Temperature Prediction of Electric Motor, Turkish Journal of Computer and Mathematics Education (TURCOMAT) [online]. 2021, n.o12. [Date of consultation: November 25, 2023]. Available in <https://turcomat.org/index.php/turkbilmat/article/view/3611>.

Excessive Heat In Electric Motors: A Common Root Cause Of Insulation Failure. England: The Association of Electrical and Mechanical Trades (January 31, 2022). [Date of consultation: November 25, 2023]. Retrieved from <https://www.theaemt.com/resource/excessive-heat-in-electric-motors-a-common-root-cause-of-insulation-failure.html#:~:text=Many%20articles%20and%20studies%20written,reasons%20a%20motor%20will%20overheat.>

Hardt, Moritz And Recht, Benjamin. Patterns, predictions, and actions: a story about machine learning. United States: Princeton University Press, 2022. 309 pp. ISBN: 9780691233734.

- Hai Guo [et al]. Predicting temperature of permanent magnet synchronous motor based on deep neural network. *Energies* [online]. September 2020, n.o18.[Date of consultation: November 25, 2023]. Available in <https://www.mdpi.com/1996-1073/13/18/4782>
ISSN: 1996-1073.
- Hernández, Roberto, Fernández, Carlos Y Baptista, Pilar. Metodología de la investigación. 6.a ed. México D.F.: McGraw-Hill, 2014. 634 pp.
ISBN: 978-1-4562-2396-0.
- Hernández, Roberto Y Mendoza, Christian. Metodología de la investigación. Las rutas cuantitativa, cualitativa y mixta [en línea]. 1.a ed. México: Mc Graw Hill Education, 2018 [fecha de consulta: 29 de octubre de 2022]. Disponible en:http://www.biblioteca.cij.gob.mx/Archivos/Materiales_de_consulta/Drogas_de_Abuso/Articulos/SampieriLasRutas.pdf
ISBN: 978-1-4562-6096-5.
- Hodson, Timothy. Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not. *Geoscientific Model Development* [Online]. July 2022, n.o.14. [Date of consultation: November 25, 2023]. Available at. <https://gmd.copernicus.org/articles/15/5481/2022/gmd-15-5481-2022-discussion.html>
- Hosseini, Siavash, Shahbandegan, Amirmohammad And Akilan, Thangarajah. Deep Neural Network Modeling for Accurate Electric Motor Temperature Prediction. En 2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE). IEEE, 2022. p. 170-175.
- Hunt, John. Advanced guide to *Python* 3 programming [online]. 1.a ed. UK: Springer International Publishing, 2019 [Date of consultation: November 25, 2023]. Available in: <https://link.springer.com/book/10.1007/978-3-030-25943-3>
ISBN: 978-3-030-25942-6.
- Humberto Ñaupas [et al.]. Metodología de la investigación cuantitativa-cualitativa y redacción de la tesis por 5a ed. Colombia: Ediciones de la U, 2018. 560 pp.
ISBN: 978-958-762-876-0.

- Jiang, Hui. Machine learning fundamentals: a concise introduction [online]. 1.a ed. USA: Cambridge University Press, 2021 [Date of consultation: November 25, 2023]. Available in: <https://www.amazon.com/Machine-Learning-Fundamentals-Concise-Introduction/dp/1108837042> ISBN: 978-1-108-83704-0.
- Joshi, Ameet. Machine learning and artificial intelligence. Switzerland: Springer Nature, 2020. 262 pp. ISBN: 978-3-030-26621-9
- Kirchgässner, Wilhelm, Wallscheid, Oliver and Böcker, Joachim. Estimating electric motor temperatures with deep residual machine learning. IEEE Transactions on Power Electronics [online]. December 2020, n.o 7. [Date of consultation: November 25, 2023]. Available in <https://ieeexplore.ieee.org/document/9296842> ISSN: 1941-0107.
- Kirchgässner, Wilhelm, Wallscheid, Oliver And Böcker, Joachim. Data-driven permanent magnet temperature estimation in synchronous motors with supervised machine learning: A benchmark. IEEE Transactions on Energy Conversion [online]. January 2021, n.o 3. [Date of consultation: November 25, 2023]. Available in <https://ieeexplore.ieee.org/document/9328339> ISSN: 1558-0059.
- Krause, Paul and Krause, Thomas. Introduction to Modern Analysis of Electric Machines and Drives. United States: John Wiley & Sons, 2022. 272 pp. ISBN: 978-1-119-90815-9
- Mahesh, Batta. Machine learning algorithms-a review. International Journal of Science and Research (IJSR) [online]. January 2020, n.º 1. [Date of consultation: November 25, 2023]. Available in <https://www.ijsr.net/archive/v9i1/ART20203995.pdf> ISSN: 2319-7064.
- Maimon, Oded and Rokach, Lior. Data Mining and Knowledge Discovery Handbook. 2.nd ed. London: Springer New York, 2010. 37 pp. ISBN: 978-0-387-09822-7

- Max Kuhn [et al.]. Applied predictive modeling New York: Springer, 2018. 595 pp. ISBN: 978-1-4614-6848-6.
- Miguel Medina [et al.]. Metodología de la investigación: técnicas e instrumentos de investigación: Instituto Universitario de Innovación Ciencia y Tecnología Inudi Perú S.A.C, 2023. 160 pp. ISBN: 978-612-5069-70-2
- Mohri, Mehryar, Rostamizadeh, Afshin and Talwalkar, Ameet. Foundations of machine learning. 2.nd ed. England: MIT press, 2018. 505 pp. ISBN: 9780262039406.
- Mostafa Al-Gabalawy [et al]. Temperature prediction for electric vehicles of permanent magnet synchronous motor using robust machine learning tools Journal of Ambient Intelligence and Humanized Computing [online]. May 2022, n.o1.[Date of consultation: November 25, 2023]. Available in <https://link.springer.com/article/10.1007/s12652-022-03888-9> .
- Nasa. Temperature variation. May 13, 2021. Available in: <https://www.grc.nasa.gov/www/k-12/airplane/etr.html>.
- Ramos, Carlos. Diseños de investigación experimental. CienciAmérica [en línea]. Enero-junio 2021, n.º 3. [Fecha de consulta: 25 de noviembre de 2023]. <https://dialnet.unirioja.es/servlet/articulo?codigo=7890336> ISSN: 1390-9592.
- Ruben Valbuena [et al]. Evaluating observed versus predicted forest biomass: R-squared, index of agreement or maximal information coefficient? European Journal of Remote Sensing [online]. May 2019, n.o1.[Date of consultation: November 25, 2023]. Available in <https://www.tandfonline.com/doi/full/10.1080/22797254.2019.1605624> ISSN: 2279-7254.
- Ryan Hughes [et al], Real-time temperature prediction of electric machines using machine learning with physically informed features by. Energy and AI [online]. October 2023, n.o1.[Date of consultation: November 25, 2023]. Available in <https://www.sciencedirect.com/science/article/pii/S2666546823000605> ISSN: 2666-5468

- Santi Prayudani [et al]. Analysis accuracy of forecasting measurement technique on random k-nearest neighbor (rknn) using MAPE and MSE, Journal of Physics: Conference Series [online]. 2019, n.o1. [Date of consultation: November 25, 2023]. Available in <https://iopscience.iop.org/article/10.1088/1742-6596/1361/1/012089/meta>
- Sosnovshchenko, Alexander. Machine Learning with Swift: Artificial Intelligence for IOS [online]. 1.st ed. United Kingdom: Packt Publishing Ltd., 2018 [Date of consultation: December 23, 2023]. Available in <http://repositorio.concytec.gob.pe/handle/20.500.12390/2260>
ISBN: 978-1787121515
- Taulli, Tom. Artificial Intelligence Basics: A Non-Technical Introduction [online]. 1.a ed. USA: Appres, 2019 [Date of consultation: November 25, 2023]. Available in: <https://link.springer.com/book/10.1007/978-1-4842-5028-0>
ISBN: 978-1-4842-5027-3.
- The World Energy Outlook 2021 [online]. France: IEA Publications. [Date of consultation: November 25, 2023]. Available in <https://www.iea.org/reports/world-energy-outlook-2021>
- Thevaraja, Mayoora, Rahman, Azizur And Gabirial, Mathew. Recent developments in data science: Comparing linear, ridge and lasso regressions techniques using wine data. In Proceedings of the International Conference on Digital Image & Signal Processing. Oxford United Kingdom, 2019. p. 1-6.
- Thosar, Poorva, et al., Prediction of Motor Temperature using Linear Regression. En 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE). IEEE, 2020. p. 7-12.
- Toliyat, Hamid And Kliman, Gerald. Handbook of electric motors. 2.nd ed. United States: CRC press, 2018. 850 pp. ISBN: 9781315220826
- Veliz, carlos. Aprendizaje automático. Introducción al aprendizaje profundo. 2.a ed. Lima: Fondo Editorial de la Pontificia Universidad Católica del Perú, 2020. 344 pp. ISBN: 978-612-317-882-6.

- Vujović, Željko. Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications* [Online]. 2021, n.o 12. [Consultation date: May 15, 2023]. Available in <https://thesai.org/Publications/ViewPaper?Volume=12&Issue=6&Code=IJACSA&SerialNo=70>
ISSN: 2156-5570.
- Wang, Harris. *Introduction to Computer Programming with Python* [online]. 1.a ed. Canada: Athabasca University, 2023 [Date of consultation: November 25, 2023]. Available in: <https://www.aupress.ca/books/oer-202301-introduction-to-computer-programming-with-Python/>
ISBN: 9781998944088 .
- Watt, Jeremy, Borhani, Reza And Katsaggelos, Aggelos. *Machine learning refined: Foundations, algorithms, and applications*. 2.nd ed. UK: Cambridge University Press, 2020. 593 pp. ISBN: 978-1-108-48072-7.
- Wilmott, Paul. *Machine learning: an applied mathematics introduction*. United States: Panda Ohana Publishing, 2019. 246 pp. ISBN: 978-1-9160816-0-4.
- Ypfruta (marzo, 2022)., Lo que tenés que saber sobre el indicador de temperatura del motor [fecha de consulta: 25 de noviembre de 2023]. Recuperado de <https://ruta.ypf.com/lo-que-tenes-que-saber-sobre-el-indicador-de-temperatura-del-motor.html#:~:text=El%20indicador%20de%20temperatura%20del%20motor%2C%20como%20su%20nombre%20lo,ver%20reflejado%20en%20el%20tablero.>
- Zhou, Zhi. *Machine Learning*. Singapore: Springer Nature, 2021. 460 pp. ISBN: 978-981-15-1966-6.

ANEXOS

ANEXO N° 1

Tabla 15. Operacionalización de variables

Variables de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Escala de medición
Independiente: Modelo híbrido basado en <i>Stacking</i>	Es un método de conjunto que realiza una combinación de modelos heterogéneos, compuestos al menos por una capa, y luego utiliza otro meta-modelo, con el fin de hacer un resumen de predicción de estos algoritmos (Chatzimpampas, et al., 2020).	Para desarrollar el modelo híbrido basado en Stacking se tomó en cuenta los pasos de la metodología KDD, siendo estos: Comprensión del dominio y Objetivos de KDD, Selección y adición, Preprocesamiento- Limpieza de datos, Transformación, Minería de datos, Evaluación e interpretación, Discovered Knowledge (Visualización e integración) (Maimon y Rokach, 2010).	Comprensión del dominio y Objetivos	-Comprensión de conceptos básicos de machine learning Malo (1), Regular(2), Bueno(3)	Ordinal
				-Conocimientos de algoritmos de machine learning Malo (1), Regular(2), Bueno(3)	Ordinal
				-Aplicación de machine learning en un dominio específico Malo (1), Regular(2), Bueno(3)	Ordinal
			Selección y adición	-Cantidad de variables seleccionadas(Uds)	Intervalo
			Preprocesamiento- Limpieza de datos	- Porcentaje de cantidad de datos completos (%) -Cantidad de valores atípicos (Uds)	Razón Intervalo
			Transformación	-Cantidad de variables cambiadas (Uds)	Intervalo
			Modelado	-Comparación de precisión de modelos (%)	Razón
			Evaluación e interpretación	-Nivel de precisión del modelo Malo (1), Regular(2), Bueno(3)	Ordinal
			Discovered Knowledge (Visualización e integración)	-Nivel de visualización del modelo Malo (1), Regular(2), Bueno(3)	Ordinal

Dependiente: Predicción de la temperatura del motor	La predicción de la temperatura del motor se refiere a un proceso de análisis y estimación anticipada de la temperatura a base a datos históricos, parámetros de funcionamiento y modelos matemáticos, donde se consideran factores como temperatura total en la boquilla, temperatura total en la cara del compresor y el comportamiento refrigerante del motor, cuyo propósito de esta predicción es anticipar posibles sobrecalentamientos dentro del rango normal y prevenir fallos (Nasa, 2021; Hardt y Recht, 2022).	Para medir los resultados del aprendizaje se emplearon fichas que incluyen las métricas de evaluación a estudiar, siendo estas: R2 Squared, MAE, MSE, RMSE y MAPE, las mismas que se obtuvieron mediante el uso de herramientas de machine learning.	R Squared (Valbuena, <i>et al.</i> , 2019)	\hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente	Razón
			Mean absolute error (MAE) (Vujović, 2021)	$P(ij)$ = valor predicho por el modelo individual i para el registro j (de n registros) T_j = valor objetivo para el registro j. $P(ij) = T_j$ y $E_i = 0$. Por lo que, el índice E_i va de 0 a infinito, y 0 corresponde al valor ideal	
			Mean squared error (MSE) (Joshi, 2020)	\hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente	
			Root mean squared error (RMSE) (Hodson, 2022)	\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.	
		Mean absolute percentage error (MAPE) Prayudani <i>et al.</i> , 2019)	Dónde a = datos reales, b = datos resultantes, n = la cantidad de datos		

Fuente: Elaboración propia.

ANEXO N° 2

Tabla 16. Matriz de consistencia

Variable	Problema	Hipótesis	Objetivos	Dimensiones	Indicadores
Modelo híbrido basado en Stacking	General		General		
	¿Como un modelo híbrido basado en Stacking permitirá mejorar la predicción de la temperatura del motor eléctrico Raghavendra?	Un modelo híbrido basado en <i>Stacking</i> permite mejorar la predicción de la temperatura del motor eléctrico Raghavendra	Desarrollar un modelo híbrido basado en <i>Stacking</i> para mejorar la predicción de la temperatura del motor eléctrico Raghavendra	Comprensión del dominio y Objetivos	-Comprensión de conceptos básicos de machine learning Malo (1), Regular(2), Bueno(3)
	Específicos	Específicos	Específicos		
	¿Cómo un modelo híbrido basado en <i>Stacking</i> permitirá mejorar la exactitud del <i>R Squared</i> en la predicción de la temperatura del motor eléctrico Raghavendra?	Raghavendra Un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del Mean absolute error la temperatura del motor eléctrico Raghavendra.	Determinar en qué porcentaje un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del <i>R squared</i> en la predicción de la temperatura del motor eléctrico Raghavendra.		-Conocimientos de algoritmos de machine learning Malo (1), Regular(2), Bueno(3) -Aplicación de machine learning en un dominio específico Malo (1), Regular(2), Bueno(3)
				Selección y adición	-Cantidad de variables seleccionadas(Uds)
				Preprocesamiento -Limpieza de datos	- Porcentaje de cantidad de datos completos (%) -Cantidad de valores atípicos (Uds)
				Transformación	-Cantidad de variables cambiadas (Uds)
				Modelado	-Comparación de precisión de modelos (%)
				Evaluación e interpretación	-Nivel de precisión del modelo Malo (1), Regular(2), Bueno(3)
				Discovered Knowledge (Visualización e integración)	-Nivel de visualización del modelo Malo (1), Regular(2), Bueno(3)
	¿Cómo un modelo híbrido basado en <i>Stacking</i> permitirá mejorar la exactitud	Un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del Root mean squared error en la	Determinar en qué porcentaje un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del <i>Mean</i>	R2 Squared (Valbuena, et al.,2019)	\hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente

Predicción de la temperatura del motor	del <i>Mean squared error</i> en la predicción de la temperatura del motor eléctrico Raghavendra?	predicción de la temperatura del motor eléctrico Raghavendra.	<i>squared error</i> en la predicción de la temperatura del motor eléctrico Raghavendra.	Mean absolute error (MAE) (Vujović, 2021)	$P(ij)$ = valor predicho por el modelo individual i para el registro j (de n registros) T_j = valor objetivo para el registro j . $P(ij) = T_j$ y $E_i = 0$. Por lo que, el índice E_i va de 0 a infinito, y 0 corresponde al valor ideal
	¿Cómo un modelo híbrido basado en <i>Stacking</i> permitirá mejorar la exactitud del <i>Root mean squared error</i> en la predicción de la temperatura del motor eléctrico Raghavendra?	Un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del <i>Mean absolute percentage error</i> en la predicción de la temperatura del motor eléctrico Raghavendra.	Determinar en qué porcentaje un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del <i>Root mean squared error</i> en la predicción de la temperatura del motor eléctrico Raghavendra.	Mean squared error (MSE) (Joshi, 2020)	\hat{y}_i = valor predicho de la muestra i -ésima ; y_i = es el valor verdadero correspondiente
	¿Cómo un modelo híbrido basado en <i>Stacking</i> permitirá mejorar la exactitud del <i>Mean absolute percentage error</i> en la predicción de la temperatura del motor eléctrico Raghavendra?	Determinar en qué porcentaje un modelo híbrido basado en <i>Stacking</i> permite mejorar la exactitud del <i>Mean absolute percentage error</i> en la predicción de la temperatura del motor eléctrico Raghavendra.	Root mean squared error (RMSE) (Hodson, 2022)	\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.	
				Mean absolute percentage error (MAPE) Prayudani <i>et al.</i> , 2019)	Dónde a = datos reales, b = datos resultantes, n = la cantidad de datos

Fuente: Elaboración propia.

**ANEXO N° 3: INSTRUMENTO PARA LA VARIABLE DEPENDIENTE:
PREDICCIÓN DE LA TEMPERATURA DEL MOTOR**

INSTRUMENTO PARA EL ALGORITMO ÁRBOL DE DECISIÓN - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Árbol de decisión

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R2 SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Árbol de decisión

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R2 SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 4

INSTRUMENTO PARA EL ALGORITMO REGRESIÓN LINEAL- FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Regresión lineal

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R2 SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Regresión lineal

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 5

INSTRUMENTO PARA EL ALGORITMO KNN - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	KNN

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	KNN

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 6

INSTRUMENTO PARA EL ALGORITMO LASSO - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Lasso

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Lasso

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 7

INSTRUMENTO PARA EL ALGORITMO RANDOM FOREST - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Random Forest

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Random Forest

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 8

INSTRUMENTO PARA EL ALGORITMO XG BOOST - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	XG Boost

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	XG Boost

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 9

INSTRUMENTO PARA EL ALGORITMO ELASTICNET - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	ElasticNet

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	ElasticNet

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 10

INSTRUMENTO PARA EL ALGORITMO RIDGE - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Ridge

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \frac{ a-b }{a}}{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Ridge

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 11

INSTRUMENTO PARA EL ALGORITMO STACKING 1 - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Stacking 1 (Random Forest Regressor)

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Stacking 1 (Random Forest Regressor)

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

ANEXO N° 12

INSTRUMENTO PARA EL ALGORITMO STACKING 2 - FICHA DE REGISTRO

EXPERTO 1:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Stacking 2 (Random Forest Regressor)

Métricas a Evaluar:

Ítem	Indicador	ŷ _i	y _i		Fórmula	Valor del indicador
1	R SQUARED ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	T _j			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) T _j = valor objetivo para el registro j. P(ij) = T _j y E _i = 0. Por lo que, el índice E _i va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P_{(ij)} - T_j $	
		ŷ _i	y _i			
3	Mean squared error (MSE) ŷ _i = valor predicho de la muestra i-ésima ; y _i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		

4	<p>Root mean squared error (RMSE)</p> <p>\hat{r}_n = índice de predicción, r_n = calificación real en el conjunto de datos de prueba, N = número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.</p>				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	
		<i>a</i>	<i>b</i>	<i>n</i>		
5	<p>Mean absolute percentage error (MAPE)</p> <p>a = datos reales, b = datos resultantes, n = la cantidad de datos</p>				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

EXPERTO 2:

FICHA DE REGISTRO	
Tipo de Prueba	Post Test
Base de datos	
Investigador	
Fecha de inicio	
Algoritmo	Stacking 2 (Random Forest Regressor)

Métricas a Evaluar:

Ítem	Indicador	\hat{y}_i	y_i		Fórmula	Valor del indicador
1	R SQUARED \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	
		P(ij)	Tj			
2	Mean absolute error (MAE) P(ij) = valor predicho por el modelo individual i para el registro j (de n registros) Tj = valor objetivo para el registro j. P(ij) = Tj y Ei = 0. Por lo que, el índice Ei va de 0 a infinito, y 0 corresponde al valor ideal				$MAE = E_i = \frac{1}{n} \sum_{j=1}^n P((ij)) - \sum_{j=1}^n P((ij)) - T_j $	
		\hat{y}_i	y_i			
3	Mean squared error (MSE) \hat{y}_i = valor predicho de la muestra i-ésima ; y_i = es el valor verdadero correspondiente				$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	
		\hat{r}_n	r_n	N		
4	Root mean squared error (RMSE) \hat{r}_n = índice de predicción, r_n = calificación real				$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	

	en el conjunto de datos de prueba, N= número de pares de predicción de valoración entre los datos de prueba y el resultado de la predicción.					
		<i>a</i>	<i>b</i>	<i>n</i>		
5	Mean absolute percentage error (MAPE) a = datos reales, b = datos resultantes, n = la cantidad de datos				$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	

Apellido y Nombres
DNI:

**INSTRUMENTO PARA LA VARIABLE INDEPENDIENTE: MODELO HIBRIDO
BASADO EN STACKING**



<https://www.Python.org/>



<https://www.anaconda.com/>

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Comprensión del dominio y Objetivos

Indicador	Ítem	Valor del indicador		
		Malo	Regular	Bueno
Comprensión de conceptos básicos de machine learning	Identificación de los tipos de aprendizaje (supervisado, no supervisado, por refuerzo)			
	Comprensión de la importancia de la división de conjunto de datos en entrenamiento y prueba			
Conocimientos de algoritmos de machine learning	Familiaridad con al menos tres algoritmos de aprendizaje supervisado, como: Regresión lineal, Árbol de decisión y Maquinas de vectores de soporte			
	Conocimiento de al menos dos algoritmos de aprendizaje no supervisado, como: K-means y Análisis de Componentes Principales			
Aplicación de machine learning en un dominio específico	Habilidad para identificar y seleccionar el algoritmo de machine learning adecuado para un problema específico			
	Capacidad para interpretar y comunicar los resultados del modelo de manera efectiva las partes interesadas no técnicas			

Apellido y Nombres

DNI:

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Selección y adición

Indicador	Modelo	Valor del indicador
Cantidad de variables seleccionadas (Uds)	Árbol de decisión	
	Regresión lineal	
	KNN	
	Lasso	
	Random Forest	
	XG Boost	
	ElasticNet	
	Ridge	
	Stacking 1 (Random Forest Regressor)	
	Stacking 2 (Random Forest Regressor)	

Fuente: Elaboración propia

Apellido y Nombres
DNI:

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Preprocesamiento-Limpieza de datos

Indicador	Ítem	Modelo	Valor del indicador
Porcentaje de cantidad de datos completos (%)	Cantidad de datos completos/ cantidad total de datos	Árbol de decisión	
		Regresión lineal	
		KNN	
		Lasso	
		Random Forest	
		XG Boost	
		ElasticNet	
		Ridge	
		Stacking 1 (Random Forest Regressor)	
		Stacking 2 (Random Forest Regressor)	

Indicador	Ítem	Modelo	Valor del indicador
Cantidad de valores atípicos (Uds)	Cantidad de valores atípicos	Árbol de decisión	
		Regresión lineal	
		KNN	
		Lasso	
		Random Forest	
		XG Boost	
		ElasticNet	
		Ridge	
		Stacking 1 (Random Forest Regressor)	
		Stacking 2 (Random Forest Regressor)	

Fuente: Elaboración propia

Apellido y Nombres

DNI:

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Transformación

Indicador	Ítem	Modelo	Valor del indicador
Cantidad de variables cambiadas (Uds)	Cantidad de variables cambiadas	Árbol de decisión	
		Regresión lineal	
		KNN	
		Lasso	
		Random Forest	
		XG Boost	
		ElasticNet	
		Ridge	
		Stacking 1 (Random Forest Regressor)	
		Stacking 2 (Random Forest Regressor)	

Fuente: Elaboración propia

Apellido y Nombres
DNI:

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Modelado

Indicador	Modelo	Valor del indicador
Comparación de precisión de modelos (%)	Árbol de decisión	
	Regresión lineal	
	KNN	
	Lasso	
	Random Forest	
	XG Boost	
	ElasticNet	
	Ridge	
	Stacking 1 (Random Forest Regressor)	
	Stacking 2 (Random Forest Regressor)	

Fuente: Elaboración propia

Apellido y Nombres
DNI:

**INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO
BASADO EN STACKING
- FICHA DE REGISTRO**

EXPERTO 1:

Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Evaluación e interpretación

Indicador	Modelo	Valor del indicador		
		Malo	Regular	Bueno
Nivel de precisión del modelo	Árbol de decisión			
	Regresión lineal			
	KNN			
	Lasso			
	Random Forest			
	XG Boost			
	ElasticNet			
	Ridge			
	Stacking 1 (Random Forest Regressor)			
	Stacking 2 (Random Forest Regressor)			

Apellido y Nombres
DNI:

INSTRUMENTO DE RECOLECCIÓN DE DATOS DEL MODELO HIBRIDO BASADO EN STACKING

- FICHA DE REGISTRO


EXPERTO 1:


Check-List	
Base de datos	
Investigador	
Fecha de inicio	
Dimensión	Discovered Knowledge (Visualización e integración)

Indicador	Modelo	Valor del indicador		
		Malo	Regular	Bueno
Nivel de visualización del modelo	Árbol de decisión			
	Regresión lineal			
	KNN			
	Lasso			
	Random Forest			
	XG Boost			
	ElasticNet			
	Ridge			
	Stacking 1 (Random Forest Regressor)			
	Stacking 2 (Random Forest Regressor)			

 Apellido y Nombres
 DNI:

FICHA DE VALIDACIÓN DE INSTRUMENTOS

Nombre del instrumento	Ficha de registro
Objetivo del instrumento	Evaluación de técnicas atreves de las métricas para los modelos de Stacking
Apellidos y nombres del experto	Sanchez Atuncar, Giancarlo
Documento de identidad	41488834
Tiempo de experiencia profesional	10 años
Grado académico	Ingeniero de sistemas
Nacionalidad	Peruano
Institución de estudio	Universidad Cesar Vallejo
Cargo	Dr. En Sistemas
Firma y sello	 GIANCARLO SANCHEZ ATUNCAR INGENIERO DE SISTEMAS Reg. CIP N° 151767
Fecha de validación	10/11/2023

Nombre del instrumento	Ficha de registro
Objetivo del instrumento	Evaluación de técnicas atreves de las métricas para los modelos de Stacking
Apellidos y nombres del experto	Chávez Rojas Blademir
Documento de identidad	80628737
Tiempo de experiencia profesional	5 años
Grado académico	Ingeniero Mecánico Electricista
Nacionalidad	Peruano
Institución de estudio	Universidad Cesar Vallejo
Cargo	Gerente de proyectos
Firma y sello	 Blademir Chávez Rojas INGENIERO MECÁNICO ELECTRICISTA REG. CIP. 216494
Fecha de validación	13/11/2023

ANEXO N° 13

CÓDIGOS DE ALGORITMOS DE MARCHINE LEARNIG

➤ ÁRBOL DE DECISIONES REGRESSOR (Tauli, 2019)

CONECTAR CON LA BASE DE DATOS

```
In [1]: #librerias para el desarrollo del proyecto
#libreria para graficas #Matplotlib es una libreria de Python especializada en la creacion de graficos en dos dimensiones.

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#librerias pandas para el manejo de los datos
#Pandas es una libreria de Python especializada en el manejo y analisis de estructuras de datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

data = pd.read_csv('../data/datos_motor.csv')

import numpy as np
#invocando a la libreria de clasificacion
from sklearn.tree import DecisionTreeClassifier
```

Figura 8. Conexión con la base de datos

Mostrar Datos

```
In [4]: data
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows × 13 columns

Figura 9. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [5]: data.shape
Out[5]: (1048575, 13)
```

Figura 10. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [6]: data.isnull().sum()
Out[6]: u_q          0
        coolant     0
        stator_winding 0
        u_d         0
        stator_tooth 0
        motor_speed  0
        i_d         0
        i_q         0
        pm          0
        stator_yoke  0
        ambient     0
        torque      0
        profile_id  0
        dtype: int64
```

Figura 11. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [8]: #numero de missing por columnas
        data.isnull().sum()
Out[8]: u_q          0
        coolant     0
        stator_winding 0
        u_d         0
        stator_tooth 0
        motor_speed  0
        i_d         0
        i_q         0
        pm          0
        stator_yoke  0
        ambient     0
        torque      0
        profile_id  0
        dtype: int64

In [9]: #Columnas con missing
        null_columns=data.columns[data.isnull().any()]
        print(null_columns)
Index([], dtype='object')
```

Figura 12. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

```
In [10]: data2=data.drop(columns = ["profile_id"])
In [11]: import missingno as msno
        msno.matrix(data2)
Out[11]: <AxesSubplot:>
```

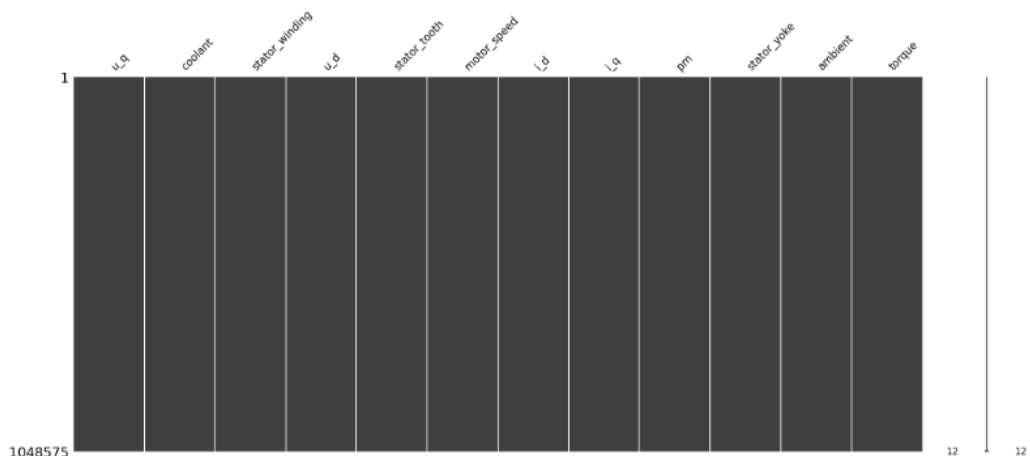


Figura 13. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [14]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data2.hist()
plt.show()
```

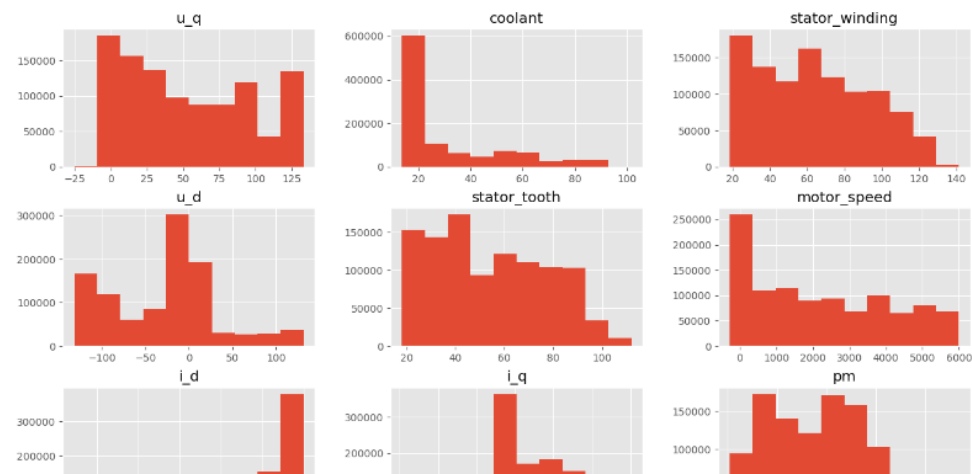


Figura 14. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [15]: X = data2.drop(['pm'],axis=1)
Y1 = data2[['pm']]
```

```
In [16]: X
```

```
Out[16]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	stator_yoke	ambient	torque
0	-0.450882	18.805172	19.089870	-0.350055	18.293219	0.002896	0.004419	0.000328	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	18.314955	19.850672	0.245417
2	-0.440894	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	18.326307	19.850657	0.176615
3	-0.327028	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002048	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	18.326652	19.850639	0.208197
...
1048570	48.158700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	49.485098	25.871140	-167.782798
1048571	49.347272	24.385932	91.492095	116.890796	68.455415	2039.850834	-98.728047	-201.925423	49.486701	25.868811	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205694	49.487069	25.873463	-165.166681
1048573	50.478838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	49.492052	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.299999	68.699325	1874.340648	-83.758749	-199.992317	49.520132	25.882027	-159.677618

1048575 rows x 11 columns

Figura 15. Selección de variables

librerías para la ejecución del modelo

```
In [20]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 16. Librerías para ejecutar el modelo

Correlación de las variables

```
In [21]: #correlación de las variables
import pandas as pds
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```



Figura 17. Correlación de variables

Pair Plots

```
In [57]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

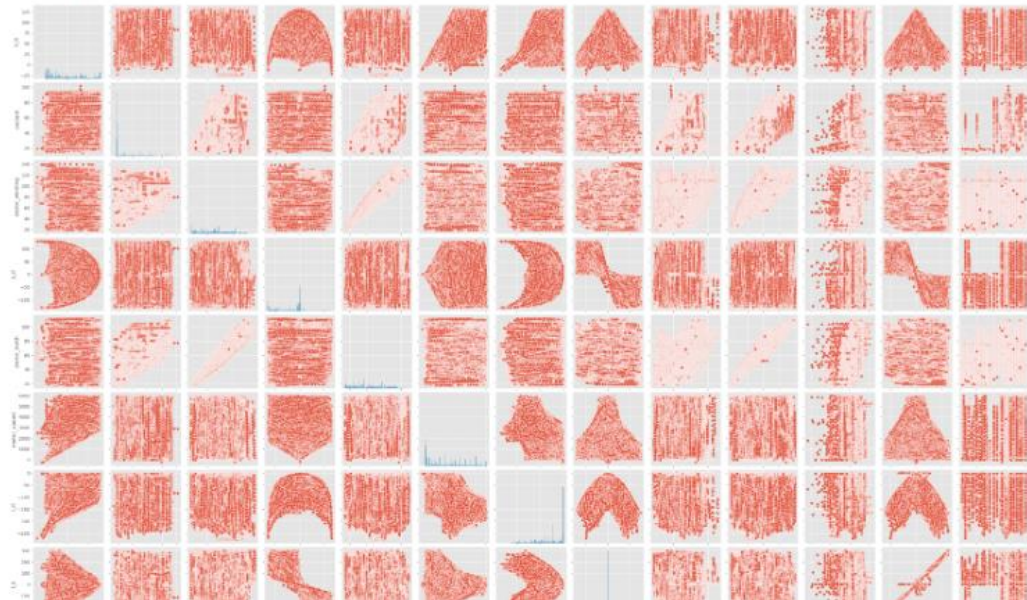


Figura 18. Visualización de Pair plots

Particionamiento de la data

```
In [22]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 11) (838860, 1)
(209715, 11) (209715, 1)
```

Figura 19. Data particionada

Modelo Árbol de Decisión Regresor

```
In [23]: from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 20. Modelo Árbol de decisión Regresor

Modelo para Y1 - temperatura del motor

```
In [24]: model_Y1 = DecisionTreeRegressor()
model_Y1.fit(X_trainset_1, y_trainset_1)

Out[24]: DecisionTreeRegressor()
```

Figura 21. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [26]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [27]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 1.0
Mean absolute error: 3.510948832550725e-11
Mean squared error: 3.0710360271468793e-17
Root mean squared error: 5.541692906636815e-09
Mean absolute percentage error: 5.264556901295749e-13
Mean squared log error: 6.376852709034484e-21
Root mean squared log error: 7.985519838454152e-11
```

Figura 22. Resultados de las métricas del modelo Árbol de decisión (Train)

```

Métricas del Modelo (Testeo)

In [29]: pred_Y1 = model_Y1.predict(X_testset_1)

In [30]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.99874621693692
Mean absolute error: 0.11729901394245323
Mean squared error: 0.5033218740143804
Root mean squared error: 0.7094518123272224
Mean absolute percentage error: 0.0022547578210965016
Mean squared log error: 0.0001734922290137518
Root mean squared log error: 0.01317164488641232

```

Figura 23. Resultados de las métricas del modelo Árbol de decisión (Test)

RESULTADOS DE ALGORITMO ÁRBOL DE DECISION REGRESSOR

R squared – R²

Árbol de decisión Regressor - Temperatura del motor

Tabla 17. Métrica de evaluación temperatura del motor – R squared Árbol de decisión

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	99.87%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 99.87% haciendo uso del **algoritmo Árbol de decisión**.

Mean absolute error

Árbol de decisión Regressor - Temperatura del motor

Tabla 18. Métrica de evaluación temperatura del motor — Mean absolute error Árbol de decisión

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	0.12

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 0.12 haciendo uso del **algoritmo Árbol de decisión**.

Mean squared error

Árbol de decisión Regressor - Temperatura del motor

Tabla 19. Métrica de evaluación temperatura del motor — Mean squared error Árbol de decisión

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	0.50

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 0.50 haciendo uso del algoritmo Árbol de decisión.

Root mean squared error

Árbol de decisión Regressor - Temperatura del motor

Tabla 20. Métrica de evaluación temperatura del motor — Root mean squared error Árbol de decisión

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	0.71

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 0.71 haciendo uso del algoritmo Árbol de decisión.

Mean absolute percentage error

Árbol de decisión Regressor- Temperatura del motor

Tabla 21. Métrica de evaluación temperatura del motor — Mean absolute percentage error Árbol de decisión

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{i=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	0.00%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.00% haciendo uso del algoritmo Árbol de decisión.

➤ LINEAL REGRESSOR (Zhou,2021)

```
CONECTAR CON LA BASE DE DATOS

* [1]: #Librerías para el desarrollo del proyecto
import matplotlib.pyplot as plt #Librería para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerías pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
data = pd.read_csv('...data/datos_motor.csv')
#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
#C:/PROYECTO_NIVEL_DE_SEGURIDAD/data
```

Figura 24. Conexión con la base de datos

Mostrar Datos

```
In [5]: data
Out[5]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows × 13 columns

Figura 25. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [6]: data.shape
Out[6]: (1048575, 13)
```

Figura 26. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [7]: data.isnull().sum()
Out[7]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 27. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [9]: #numero de missing por columnas
        data.isnull().sum()
Out[9]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 28. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

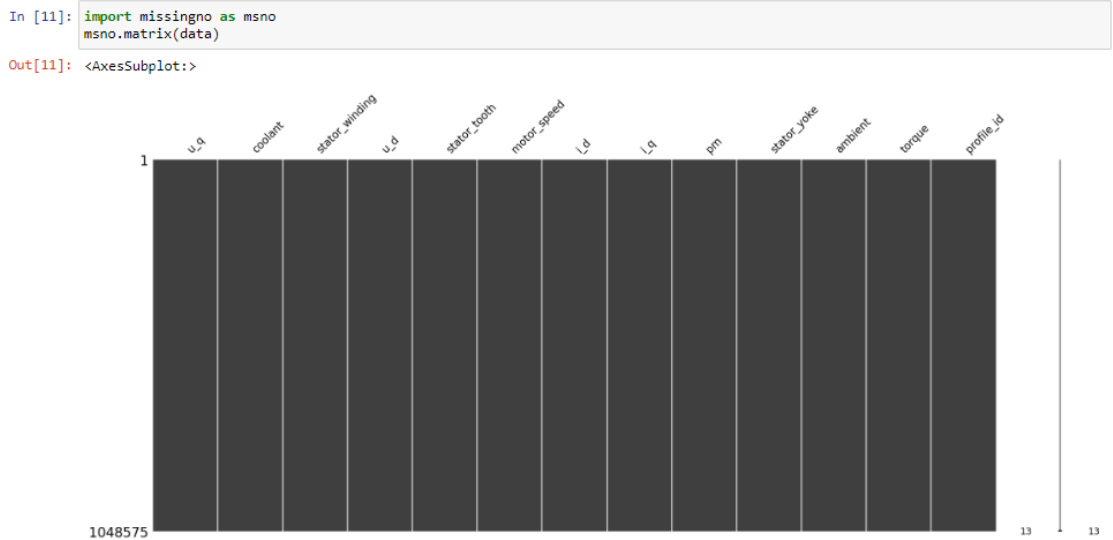


Figura 29. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [16]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

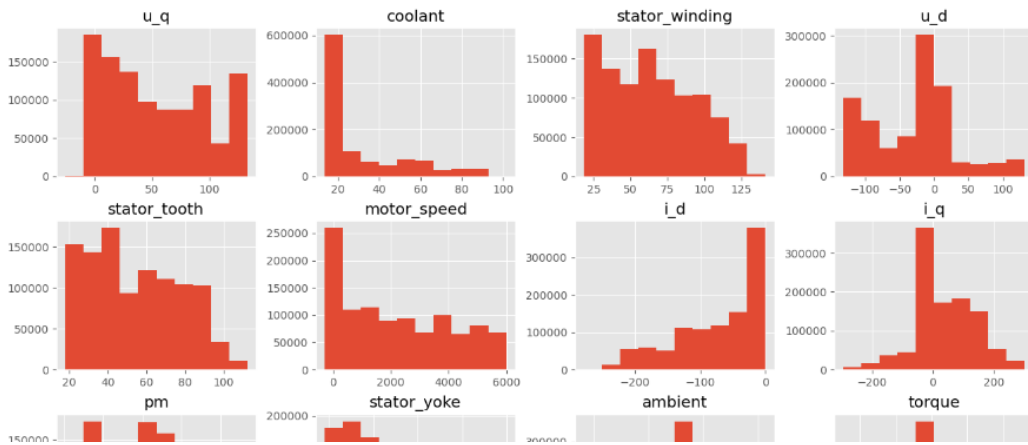


Figura 30. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [17]: X = data.drop(['pm'],axis=1)
Y1 = data[['pm']]
```

```
In [18]: X.shape
```

```
Out[18]: (1048575, 12)
```

```
In [15]: #Y1 = data['heating_load']
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data['cooling_load']
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 31. Selección de variables

librerías para la ejecución del modelo

```
In [19]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 32. Librerías para ejecutar el modelo

Correlación de las variables

```
In [20]: #correlacion de las variables
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

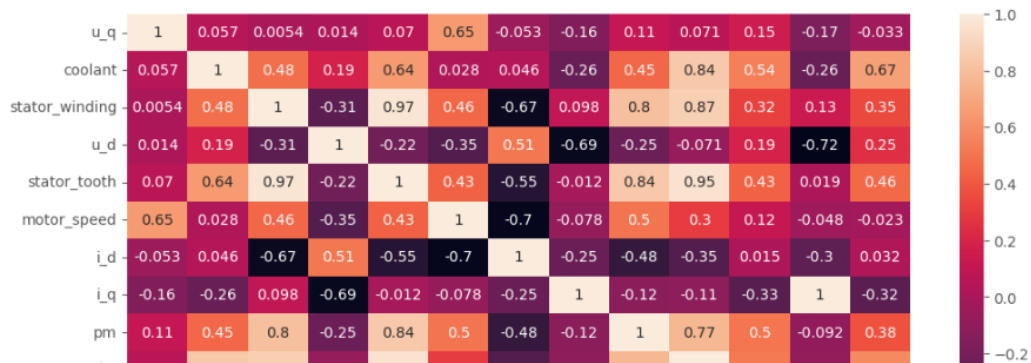


Figura 33. Correlación de variables

Pair Plots

```
In [21]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```



Figura 34. Visualización de Pair plots

Particionamiento de la data

```
In [22]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 35. Data particionada

Regresión Lineal

```
In [23]: from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 36. Modelo Regresión lineal

Modelo para Y1 - temperatura del motor

```
In [24]: model_Y1 = LinearRegression()
model_Y1.fit(X_trainset_1, y_trainset_1)

Out[24]: LinearRegression()
```

Figura 37. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [25]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [26]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared log error:", msle_Y1)
#print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.8550051092015712
Mean absolute error: 5.241812581019085
Mean squared error: 50.67724606531592
Root mean squared error: 7.118795267832607
Mean absolute percentage error: 0.10383835940134893
```

Figura 38. Resultados de las métricas del modelo Regresión lineal (Train)

Métricas del Modelo (Testeo)

```
In [28]: pred_Y1 = model_Y1.predict(X_testset_1)

In [29]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared log error:", msle_Y1)
#print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.8540481799366944
Mean absolute error: 5.252260636505124
Mean squared error: 50.98057831911134
Root mean squared error: 7.140068509413011
Mean absolute percentage error: 0.10398081097010825
```

Figura 39. Resultados de las métricas del modelo Regresión lineal (Test)

RESULTADOS DE ALGORITMO LINEAL REGRESSOR

R squared – R²

Regresión lineal- Temperatura del motor

Tabla 22. Métrica de evaluación temperatura del motor – R squared Regresión lineal

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	85.40%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico, se obtiene un valor de R squared= 85.40% haciendo uso del **algoritmo Regresión lineal.**

Mean absolute error

Regresión lineal- Temperatura del motor

Tabla 23. Métrica de evaluación temperatura del motor — Mean absolute error Regresión lineal

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	5.25

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico, se obtiene un valor de Mean absolute error= 5.25 haciendo uso del **algoritmo Regresión lineal.**

Mean squared error

Regresión lineal - Temperatura del motor

Tabla 24. Métrica de evaluación temperatura del motor — Mean squared error Regresión lineal

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	50.98

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 50.98 haciendo uso del **algoritmo Regresión lineal**.

Root mean squared error

Regresión lineal- Temperatura del motor

Tabla 25. Métrica de evaluación temperatura del motor — Root mean squared error Regresión lineal

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	7.14

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 7.14 haciendo uso del algoritmo Regresión lineal.

Mean absolute percentage error

Regresión lineal- Temperatura del motor

Tabla 26. Métrica de evaluación temperatura del motor — Mean absolute percentage error Regresión lineal

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	0.10%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.10% haciendo uso del algoritmo Regresión lineal.

➤ ELASTICNET (Sosnovshchenko y Baiev, 2028)

```
CONECTAR CON LA BASE DE DATOS

•[1]: #librerías para el desarrollo del proyecto
import matplotlib.pyplot as plt #librería para graficas
from matplotlib.colors import ListedColorMap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#librerías pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #archivos de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
data = pd.read_csv('...data/datos_motor.csv')
#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
#C:\PROYECTO_NIVEL_DE_SEGURIDAD\data
```

Figura 40. Conexión con la base de datos

Mostrar Datos

```
In [5]: data
Out[5]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.916489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 41. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [6]: data.shape
Out[6]: (1048575, 13)
```

Figura 42. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [7]: data.isnull().sum()
Out[7]:
```

u_q	0
coolant	0
stator_winding	0
u_d	0
stator_tooth	0
motor_speed	0
i_d	0
i_q	0
pm	0
stator_yoke	0
ambient	0
torque	0
profile_id	0
dtype: int64	

Figura 43. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [9]: #numero de missing por columnas  
data.isnull().sum()
```

```
Out[9]: u_q          0  
coolant      0  
stator_winding  0  
u_d          0  
stator_tooth  0  
motor_speed  0  
i_d          0  
i_q          0  
pm           0  
stator_yoke  0  
ambient      0  
torque       0  
profile_id   0  
dtype: int64
```

Figura 44. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

```
In [11]: import missingno as msno  
msno.matrix(data)
```

```
Out[11]: <AxesSubplot:>
```

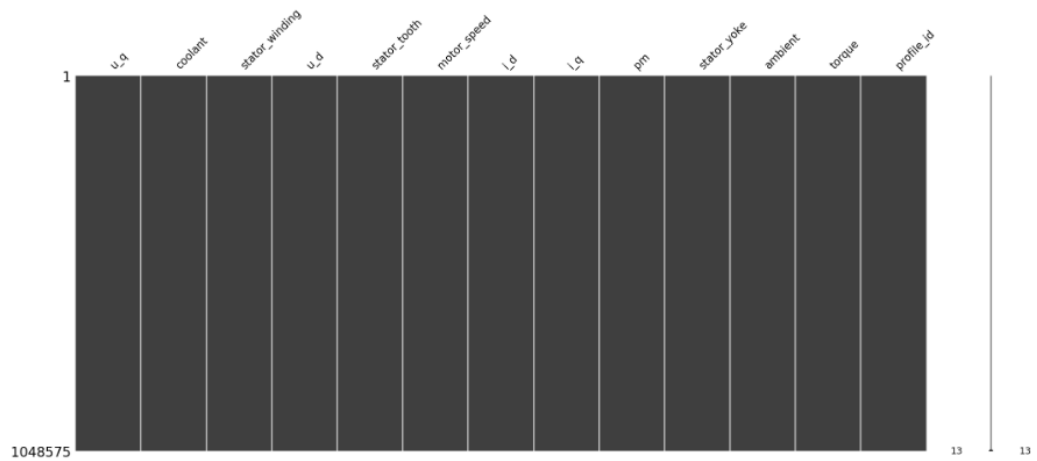


Figura 45. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [16]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

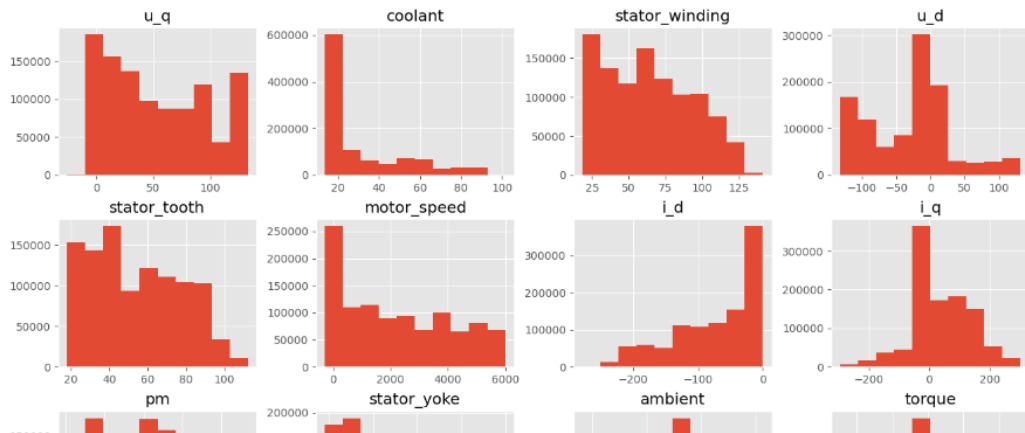


Figura 46. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [17]: X = data.drop(['pm'],axis=1)
Y1 = data[['pm']]
```

```
In [18]: X.shape
```

```
Out[18]: (1048575, 12)
```

```
In [15]: #Y1 = data['heating_Load']
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data['cooling_Load']
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 47. Selección de variables

librerías para la ejecución del modelo

```
In [19]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 48. Librerías para ejecutar el modelo

Correlación de las variables

```
In [20]: #correlacion de las variables
import pandas as pds
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

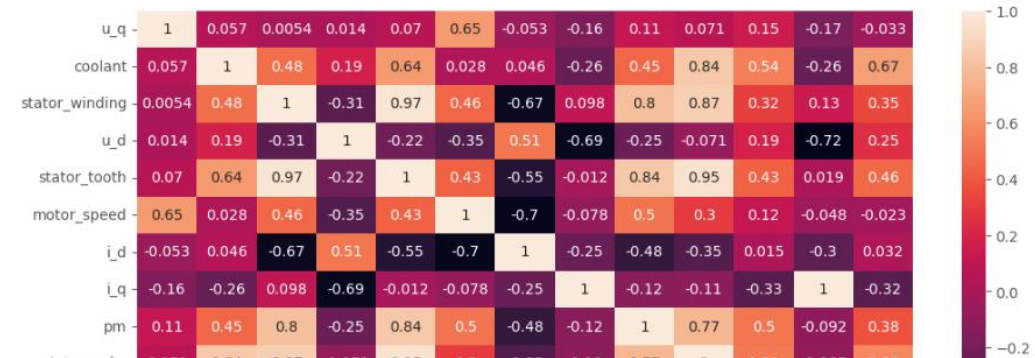


Figura 49. Correlación de variables

Pair Plots

```
In [21]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

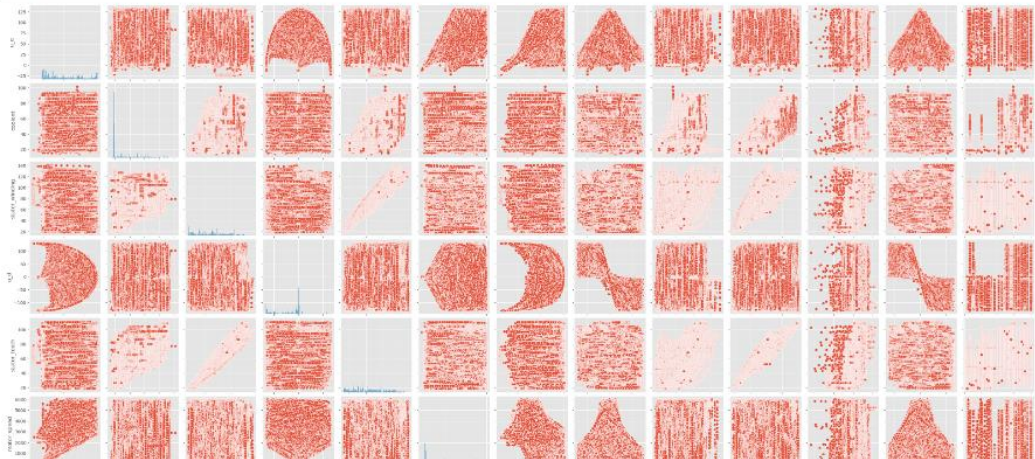


Figura 50. Visualización de Pair plots

Particionamiento de la data

```
In [22]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 51. Data particionada

Modelo ElasticNet

```
In [35]: from sklearn.linear_model import ElasticNet
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 52. Modelo ElasticNet

Modelo para Y1 - temperatura del motor

```
In [39]: model_Y1 = ElasticNet()
model_Y1.fit(X_trainset_1, y_trainset_1)

Out[39]: ElasticNet()
```

Figura 53. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [40]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [41]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.8367603836428598
Mean absolute error: 5.349641369611213
Mean squared error: 54.41462664723118
Root mean squared error: 7.376627050843169
Mean absolute percentage error: 0.10633240666156842
```

Figura 54. Resultados de las métricas del modelo ElasticNet (Train)

Métricas del Modelo (Testeo)

```
In [43]: pred_Y1 = model_Y1.predict(X_testset_1)

In [44]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.8355663376863637
Mean absolute error: 5.36030037047446
Mean squared error: 54.81518770960681
Root mean squared error: 7.403727960264803
Mean absolute percentage error: 0.10651527764248152
```

Figura 55. Resultados de las métricas del modelo ElasticNet (Test)

RESULTADOS DE ALGORITMO ELASTICNET

R squared – R²

ElasticNet- Temperatura del motor

Tabla 27. Métrica de evaluación temperatura del motor – R squared ElasticNet

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	83.56%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 83.56% haciendo uso del **algoritmo ElasticNet**.

Mean absolute error

ElasticNet- Temperatura del motor

Tabla 28. Métrica de evaluación temperatura del motor — Mean absolute error ElasticNet

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	5.36

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 5.36 haciendo uso del algoritmo ElasticNet.

Mean squared error

ElasticNet - Temperatura del motor

Tabla 29. Métrica de evaluación temperatura del motor — Mean squared error ElasticNet

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	54.82

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 54.82 haciendo uso del algoritmo ElasticNet.

Root mean squared error

ElasticNet- Temperatura del motor

Tabla 30. Métrica de evaluación temperatura del motor — Root mean squared error ElasticNet

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	7.40

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 7.40 haciendo uso del algoritmo ElasticNet.

Mean absolute percentage error

ElasticNet- Temperatura del motor

Tabla 31. Métrica de evaluación temperatura del motor — Mean absolute percentage error ElasticNet

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a - b}{a} \right }{n} * 100\%$	0.11%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico, se obtiene un valor de Mean absolute percentage error= 0.11% haciendo uso del algoritmo ElasticNet.

➤ KNN REGRESSOR (Jiang,2021)

```
CONECTAR CON LA BASE DE DATOS

•[1]: #librerias para el desarrollo del proyecto
import matplotlib.pyplot as plt #libreria para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#librerias pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#entlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
data = pd.read_csv('...data/datos_motor.csv')
#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
#C:/PROYECTO_NIVEL_DE_SEGURIDAD/data
```

Figura 56. Conexión con la base de datos

Mostrar Datos

```
In [5]: data
Out[5]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 577. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [6]: data.shape
Out[6]: (1048575, 13)
```

Figura 58. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [7]: data.isnull().sum()
Out[7]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 59. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [9]: #numero de missing por columnas
        data.isnull().sum()
Out[9]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 60. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

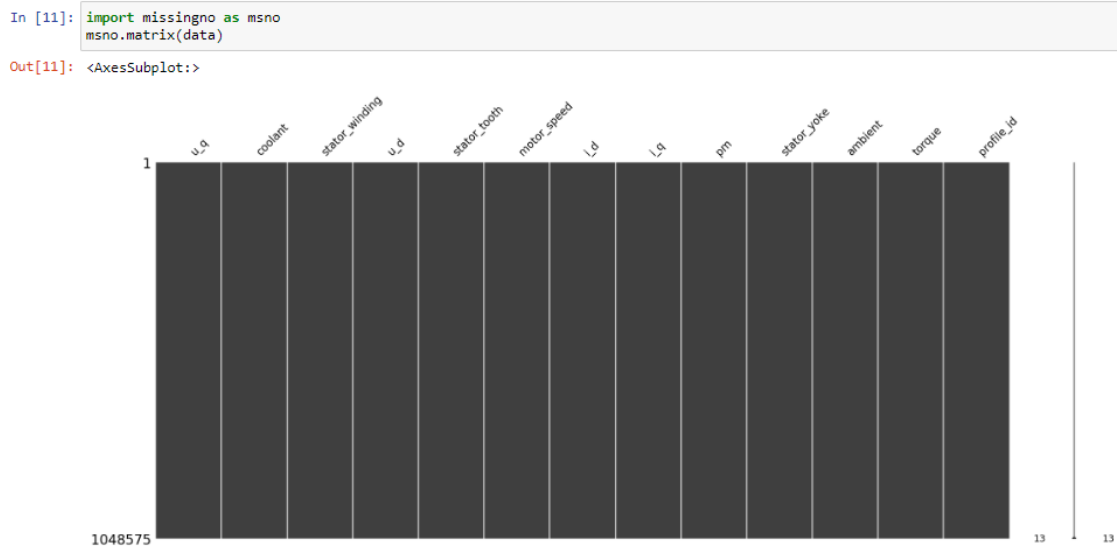


Figura 61. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [16]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

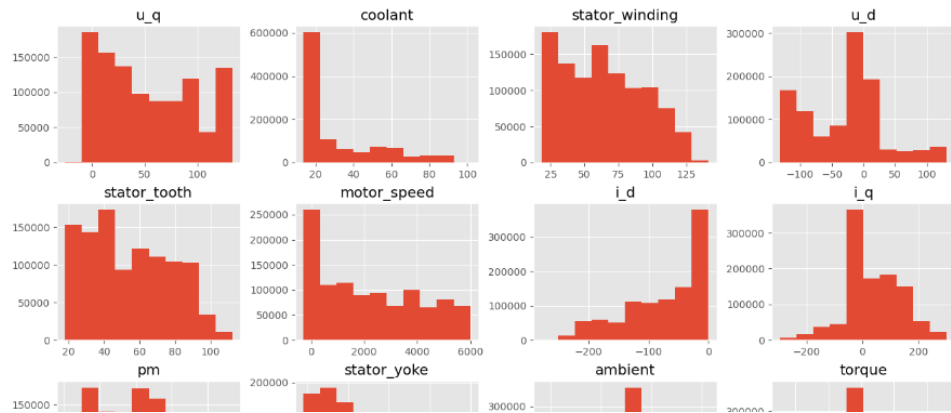


Figura 62. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [17]: X = data.drop(['pm'], axis=1)
Y1 = data[['pm']]
```

```
In [18]: X.shape
```

```
Out[18]: (1048575, 12)
```

```
In [15]: #Y1 = data['heating_Load']
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data['cooling_Load']
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 63. Selección de variables

librerías para la ejecución del modelo

```
In [19]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 64. Librerías para ejecutar el modelo

Correlación de las variables

```
In [20]: #correlacion de las variables
import pandas as pds
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

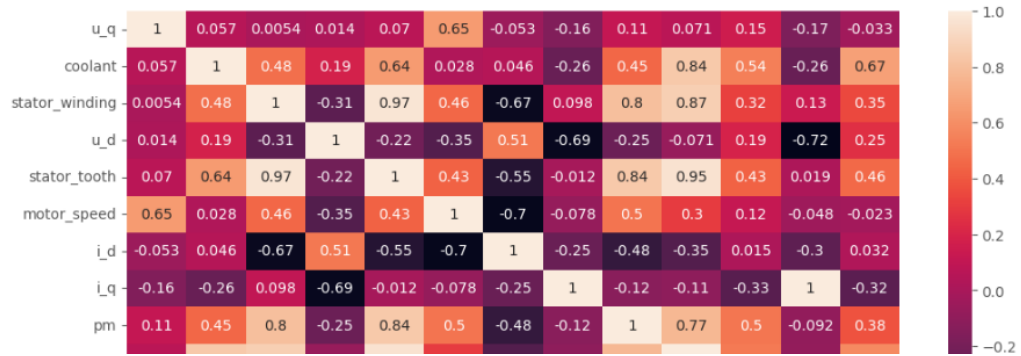


Figura 65. Correlación de variables

Pair Plots

```
In [21]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

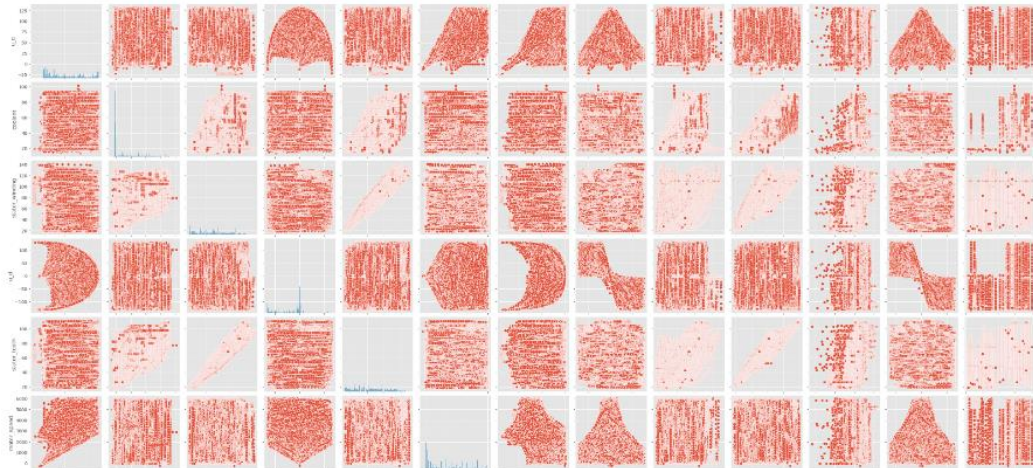


Figura 66. Visualización de Pair plots

Particionamiento de la data

```
In [51]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 67. Data particionada

Modelo KNN Regressor

```
In [61]: from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 68. Modelo KNN Regressor

Modelo para Y1 - temperatura del motor

```
In [62]: model_Y1 = KNeighborsRegressor()
model_Y1.fit(X_trainset_1, y_trainset_1)
```

```
Out[62]: KNeighborsRegressor()
```

Figura 69. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [63]: pred_Y1 = model_Y1.predict(X_trainset_1)
```

```
In [65]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)
```

```
print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))
```

```
R2 score: 0.9832124715586293
Mean absolute error: 0.9097272997918444
Mean squared error: 6.580705389138205
Root mean squared error: 2.5652885506495344
Mean absolute percentage error: 0.015907184950445043
```

Figura 70. Resultados de las métricas del modelo KNN (Train)

Métricas del Modelo (Testeo)

```
In [67]: pred_Y1 = model_Y1.predict(X_testset_1)
```

```
In [68]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)
```

```
print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))
```

```
R2 score: 0.9733413770044375
Mean absolute error: 1.1765129796557583
Mean squared error: 10.455106071572438
Root mean squared error: 3.2334356451880155
Mean absolute percentage error: 0.02062078130104976
```

Figura 71. Resultados de las métricas del modelo KNN (Test)

RESULTADOS DE ALGORITMO KNN_REGRESSOR

R squared – R²

KNN- Temperatura del motor

Tabla 32. Métrica de evaluación temperatura del motor – R squared KNN

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	97.33%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 97.33% haciendo uso del **algoritmo KNN**.

Mean absolute error

KNN- Temperatura del motor

Tabla 33. Métrica de evaluación temperatura del motor — Mean absolute error KNN

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	1.18

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 1.18 haciendo uso del algoritmo KNN.

Mean squared error

KNN - Temperatura del motor

Tabla 34. Métrica de evaluación temperatura del motor — Mean squared error KNN

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	10.46

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 10.46 haciendo uso del algoritmo KNN.

Root mean squared error

KNN- Temperatura del motor

Tabla 35. Métrica de evaluación temperatura del motor — Root mean squared error KNN

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	3.23

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 3.23 haciendo uso del algoritmo KNN.

Mean absolute percentage error

KNN- Temperatura del motor

Tabla 36. Métrica de evaluación temperatura del motor — Mean absolute percentage error KNN

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a - b}{a} \right }{n} * 100\%$	0.02%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.02% haciendo uso del algoritmo KNN.

➤ LASSO (Kuhn,2018)

CONECTAR CON LA BASE DE DATOS

```
#Librerías para el desarrollo del proyecto
import matplotlib.pyplot as plt #librería para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerías pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
```

Figura 72. Conexión con la base de datos

Mostrar Datos

```
In [122]: data
Out[122]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows × 13 columns

Figura 73. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [123]: data.shape
Out[123]: (1048575, 13)
```

Figura 74. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [124]: data.isnull().sum()
Out[124]:
```

u_q	0
coolant	0
stator_winding	0
u_d	0
stator_tooth	0
motor_speed	0
i_d	0
i_q	0
pm	0
stator_yoke	0
ambient	0
torque	0
profile_id	0
dtype:	int64

Figura 75. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [126]: #numero de missing por columnas
data.isnull().sum()
```

```
Out[126]: u_q          0
coolant      0
stator_winding  0
u_d          0
stator_tooth  0
motor_speed  0
i_d          0
i_q          0
pm           0
stator_yoke  0
ambient      0
torque       0
profile_id   0
dtype: int64
```

Figura 76. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

```
In [128]: import missingno as msno
msno.matrix(data)
```

```
Out[128]: <AxesSubplot:>
```

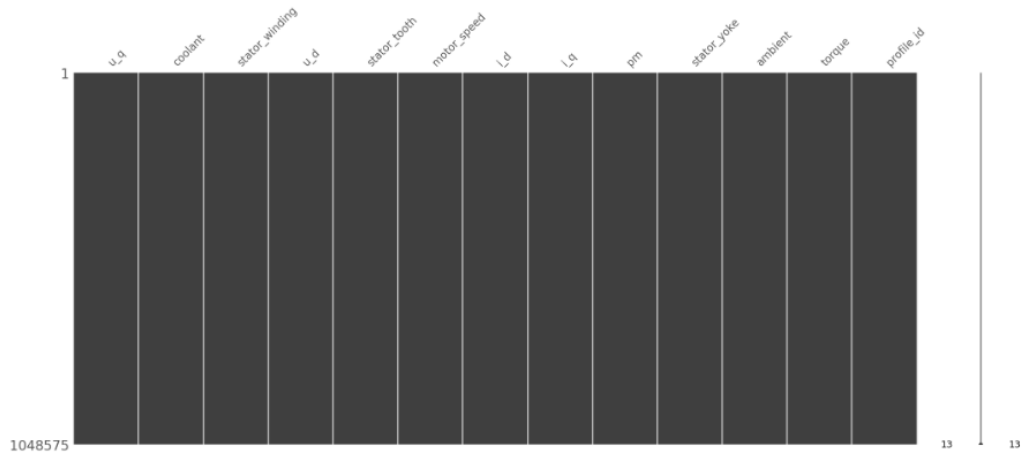


Figura 77. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [130]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

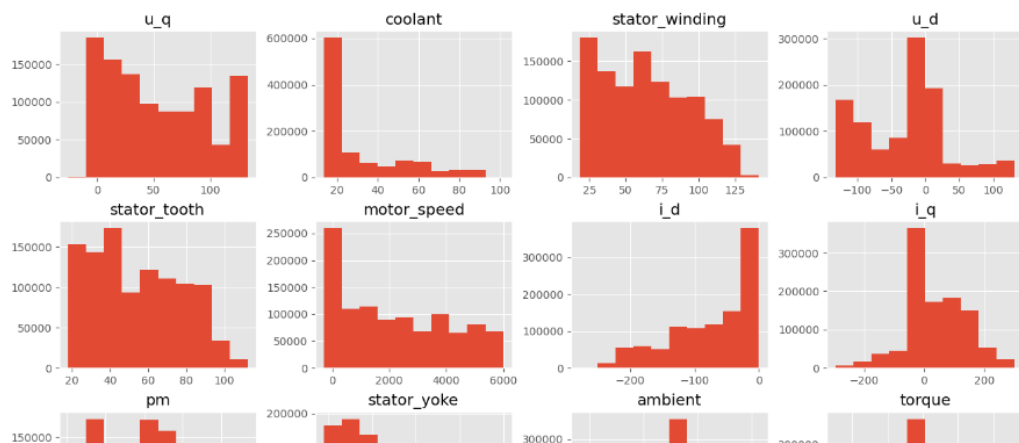


Figura 78. Exploración de las variables

Selección de Variables para el modelo

```
In [75]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#          'glazing_area', 'glazing_area_distribution']].values

In [131]: X = data.drop(['pm'],axis=1)
          Y1 = data[['pm']]

In [132]: X.shape
Out[132]: (1048575, 12)

In [15]: #Y1 = data['heating_Load']
          #Y1.shape
Out[15]: (768,)

In [16]: #Y2 = data['cooling_Load']
          #Y2.shape
Out[16]: (768,)
```

Figura 79. Selección de variables

librerías para la ejecución del modelo

```
In [133]: from sklearn.metrics import roc_curve, auc
          from sklearn import datasets
          from sklearn.multiclass import OneVsRestClassifier
          from sklearn.svm import LinearSVC
          from sklearn.preprocessing import label_binarize
          from sklearn.model_selection import train_test_split
          import matplotlib.pyplot as plt
```

Figura 80. Librerías para ejecutar el modelo

Correlación de las variables

```
In [134]: #correlacion de las variables
          import pandas as pds
          import seaborn as sns
          import matplotlib.pyplot as plt
          data
          corr_df = data.corr(method='pearson')

          plt.figure(figsize=(12,6))
          sns.heatmap(corr_df, annot=True)
          plt.show()
```

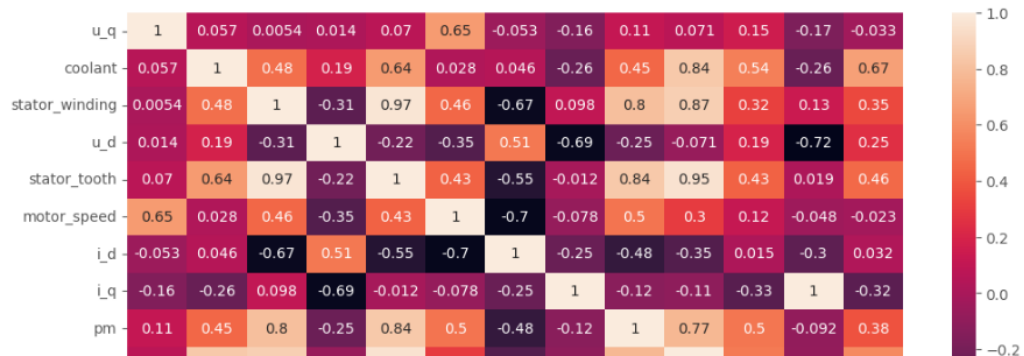


Figura 81. Correlación de variables

Pair Plots

```
In [135]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

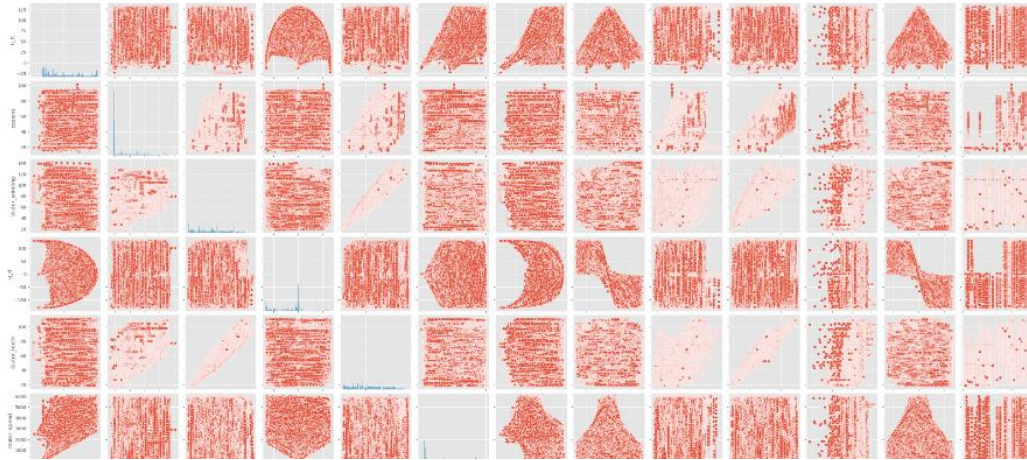


Figura 82. Visualización de Pair plots

Particionamiento de la data

```
In [137]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print (X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 83. Data particionada

Modelo Lasso

```
In [138]: from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 84. Modelo Lasso

Modelo para Y1 - temperatura del motor

```
In [139]: model_Y1 = Lasso()
model_Y1.fit(X_trainset_1, y_trainset_1)
```

Figura 85. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [141]: pred_Y1 = model_Y1.predict(X_trainset_1)
```

```
In [143]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))
```

R2 score: 0.8396783274553107
Mean absolute error: 5.32289650796612
Mean squared error: 53.77364491652296
Root mean squared error: 7.333051541924614
Mean absolute percentage error: 0.10569510470737405

Figura 86. Resultados de las métricas del modelo Lasso (Train)

Métricas del Modelo (Testeo)

```
In [145]: pred_Y1t = model_Y1.predict(X_testset_1)
```

```
In [146]: r2_Y1t = r2_score(pred_Y1t, y_testset_1)
mae_Y1t = mean_absolute_error(y_testset_1, pred_Y1t)
mse_Y1t = mean_squared_error(y_testset_1, pred_Y1t)
mape_Y1t = mean_absolute_percentage_error(y_testset_1, pred_Y1t)
#msle_Y1t = mean_squared_log_error(y_testset_1, pred_Y1t)

print("R2 score:", r2_Y1t)
print("Mean absolute error:", mae_Y1t)
print("Mean squared error:", mse_Y1t)
print("Root mean squared error:", sqrt(mse_Y1t))
print("Mean absolute percentage error:", mape_Y1t)
#print("Mean squared Log error:", msle_Y1t)
#print("Root mean squared Log error:", sqrt(msle_Y1t))
```

R2 score: 0.838539279739134
Mean absolute error: 5.333247393317618
Mean squared error: 54.151946309004835
Root mean squared error: 7.358800602612143
Mean absolute percentage error: 0.10587349000773011

Figura 87. Resultados de las métricas del modelo Lasso (Test)

RESULTADOS DE ALGORITMO LASSO

R squared – R²

Lasso- Temperatura del motor

Tabla 37. Métrica de evaluación temperatura del motor – R squared Lasso

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	83.85%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 83.85% haciendo uso del **algoritmo Lasso**.

Mean absolute error

Lasso- Temperatura del motor

Tabla 38. Métrica de evaluación temperatura del motor — Mean absolute error Lasso

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij)} - \sum_{j=1}^n P_{((ij)} - T_j $	5.33

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 5.33 haciendo uso del algoritmo Lasso.

Mean squared error

Lasso - Temperatura del motor

Tabla 39. Métrica de evaluación temperatura del motor — Mean squared error Lasso

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	54.15

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 54.15 haciendo uso del algoritmo Lasso.

Root mean squared error

Lasso- Temperatura del motor

Tabla 40. Métrica de evaluación temperatura del motor — Root mean squared error Lasso

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	7.36

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 7.36 haciendo uso del algoritmo Lasso.

Mean absolute percentage error

Lasso- Temperatura del motor

Tabla 41. Métrica de evaluación temperatura del motor — Mean absolute percentage error Lasso

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{i=1}^n \left \frac{a-b}{a} \right }{n} * 100\%$	0.11%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.11% haciendo uso del algoritmo Lasso.

➤ RANDOM FOREST REGRESSOR (James yWtten,2021)

CONECTAR CON LA BASE DE DATOS

```
#Librerias para el desarrollo del proyecto
import matplotlib.pyplot as plt #Libreria para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerias pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
```

Figura 88. Conexión con la base de datos

Mostrar Datos

```
In [5]: data
```

```
Out[5]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 89. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [6]: data.shape
```

```
Out[6]: (1048575, 13)
```

Figura 90. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [7]: data.isnull().sum()
Out[7]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 91. Presentación de la cantidad de columnas que tienen valores nulos

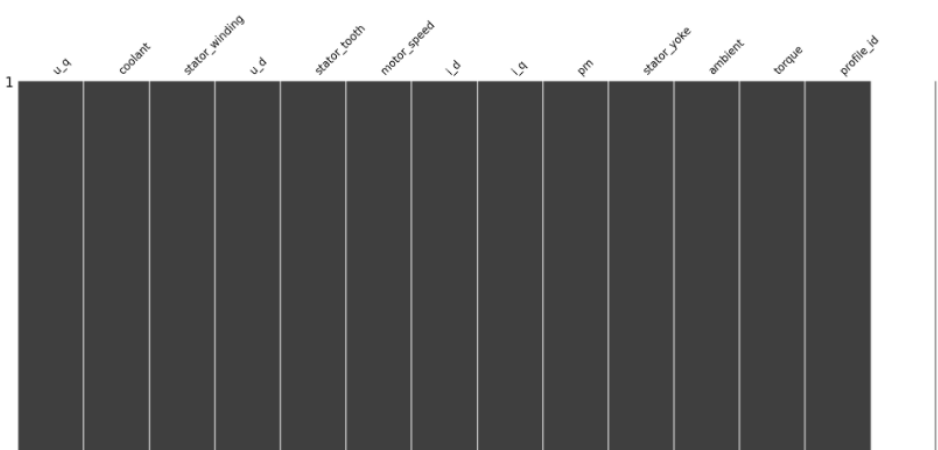
verificando valor perdido

```
In [9]: #numero de missing por columnas
        data.isnull().sum()
Out[9]: u_q          0
        coolant      0
        stator_winding 0
        u_d          0
        stator_tooth 0
        motor_speed  0
        i_d          0
        i_q          0
        pm           0
        stator_yoke  0
        ambient      0
        torque       0
        profile_id   0
        dtype: int64
```

Figura 92. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

```
In [11]: import missingno as msno
         msno.matrix(data)
Out[11]: <AxesSubplot:>
```



The heatmap displays a data matrix with 13 columns and 1048575 rows. The columns are labeled: u_q, coolant, stator_winding, u_d, stator_tooth, motor_speed, i_d, i_q, pm, stator_yoke, ambient, torque, and profile_id. The plot shows a solid black area, indicating that all data points are non-missing.

Figura 93. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [16]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

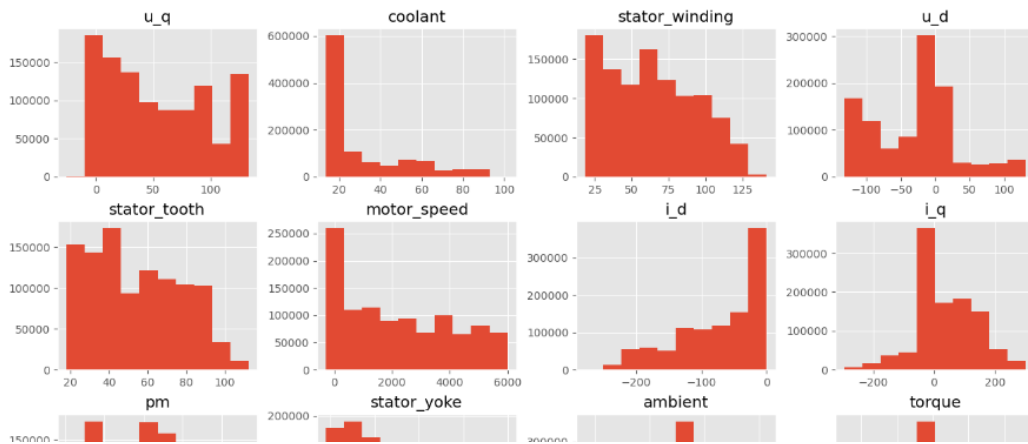


Figura 94. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [17]: X = data.drop(['pm'],axis=1)
Y1 = data[['pm']]
```

```
In [18]: X.shape
```

```
Out[18]: (1048575, 12)
```

```
In [15]: #Y1 = data[['heating_Load']]
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data[['cooling_Load']]
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 95. Selección de variables

librerías para la ejecución del modelo

```
In [94]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 96. Librerías para ejecutar el modelo

Correlación de las variables

```
In [95]: #correlacion de las variables
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

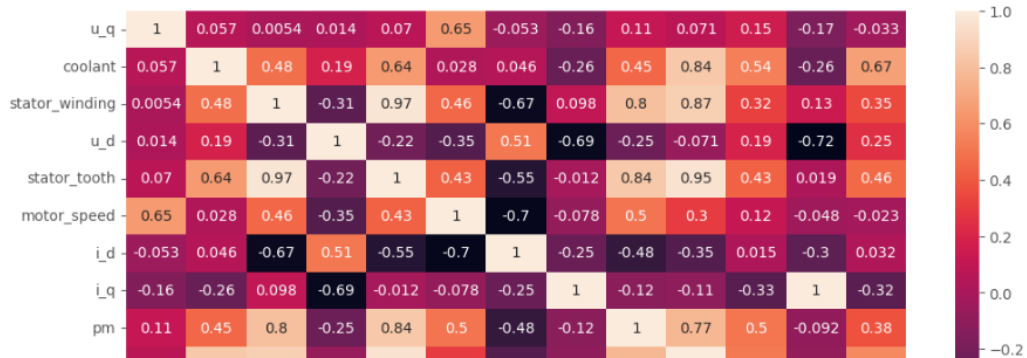


Figura 97. Correlación de variables

Pair Plots

```
In [21]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```



Figura 98. Visualización de Pair plots

Particionamiento de la data

```
In [96]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 99. Data particionada

Modelo Ramdon Forest

```
In [97]: from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 100. Modelo Random Forest

Modelo para Y1 - temperatura del motor

```
In [98]: model_Y1 = RandomForestRegressor()
model_Y1.fit(X_trainset_1, y_trainset_1)
```

Figura 101. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [99]: pred_Y1 = model_Y1.predict(X_trainset_1)
```

```
In [100]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.999969147240091
Mean absolute error: 0.032595808599293195
Mean squared error: 0.01234227416415139
Root mean squared error: 0.1110957882376798
Mean absolute percentage error: 0.0006369622229413963
```

Figura 102. Resultados de las métricas de Random Forest (Train)

Métricas del Modelo (Testeo)

```
In [103]: pred_Y1 = model_Y1.predict(X_testset_1)
```

```
In [104]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.9998099551434135
Mean absolute error: 0.08519870857626165
Mean squared error: 0.0762341515332881
Root mean squared error: 0.276105326883217
Mean absolute percentage error: 0.0016690382178487218
```

Figura 103. Resultados de las métricas de Random Forest (Test)

RESULTADOS DE ALGORITMO RANDOM FORES REGRESSOR

R squared – R²

Random Forest Regressor- Temperatura del motor

Tabla 42. Métrica de evaluación temperatura del motor – R squared Random Forest Regressor.

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	99.98%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 99.98% haciendo uso del **algoritmo Random Forest Regressor**.

Mean absolute error

Random Forest Regressor- Temperatura del motor

Tabla 43. Métrica de evaluación temperatura del motor — Mean absolute error Random Forest Regressor.

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	0.09

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico, se obtiene un valor de Mean absolute error= 0.09 haciendo uso del algoritmo Random Forest Regressor.

Mean squared error

Random Forest Regressor - Temperatura del motor

Tabla 44. Métrica de evaluación temperatura del motor — Mean squared error Random Forest Regressor.

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$	0.08

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 0.08 haciendo uso del algoritmo Random Forest Regressor.

Root mean squared error

Random Forest Regressor- Temperatura del motor

Tabla 45. Métrica de evaluación temperatura del motor — Root mean squared error Random Forest Regressor.

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{f}_n - r_n)^2}{N}}$	0.28

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 0.28 haciendo uso del algoritmo Random Forest Regressor.

Mean absolute percentage error

Random Forest Regressor- Temperatura del motor

Tabla 46. Métrica de evaluación temperatura del motor — Mean absolute percentage error Random Forest Regressor

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a - b}{a} \right }{n} * 100\%$	0.00%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.00% haciendo uso del algoritmo Random Forest Regressor.

➤ RIDGE – REGRESSOR (Thevaraja, Rahman y Gabirial,2019)

CONECTAR CON LA BASE DE DATOS

```
#Librerias para el desarrollo del proyecto
import matplotlib.pyplot as plt #Libreria para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerias pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
```

Figura 104. Conexión con la base de datos

Mostrar Datos

```
In [63]: data
```

```
Out[63]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows × 13 columns

Figura 105. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [64]: data.shape
```

```
Out[64]: (1048575, 13)
```

Figura 106. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [65]: data.isnull().sum()
Out[65]: u_q      0
         coolant  0
         stator_winding  0
         u_d      0
         stator_tooth  0
         motor_speed  0
         i_d      0
         i_q      0
         pm       0
         stator_yoke  0
         ambient  0
         torque   0
         profile_id  0
         dtype: int64
```

Figura 107. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [67]: #numero de missing por columnas
         data.isnull().sum()
Out[67]: u_q      0
         coolant  0
         stator_winding  0
         u_d      0
         stator_tooth  0
         motor_speed  0
         i_d      0
         i_q      0
         pm       0
         stator_yoke  0
         ambient  0
         torque   0
         profile_id  0
         dtype: int64
```

Figura 108. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

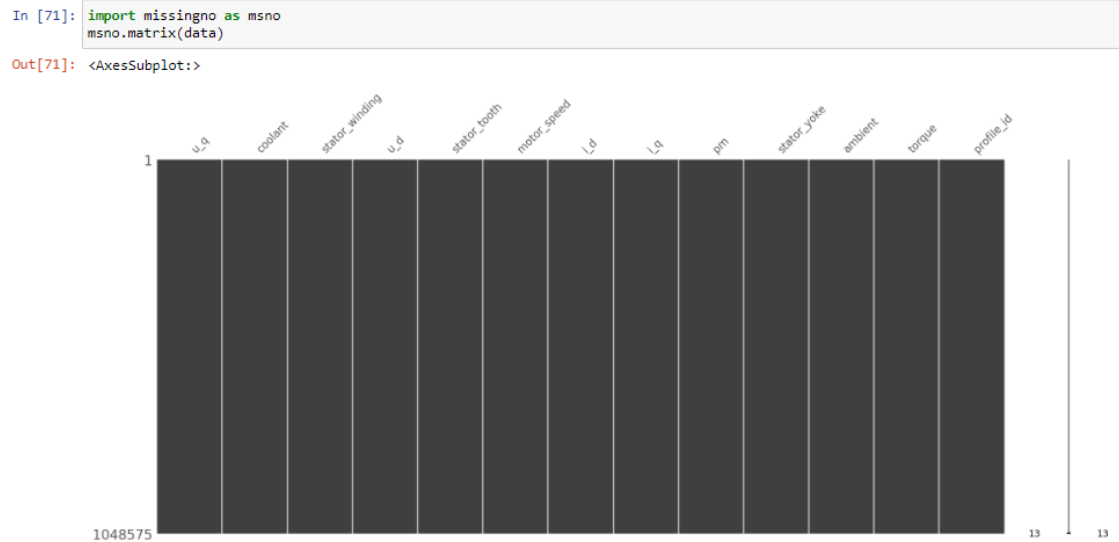


Figura 109. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [74]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

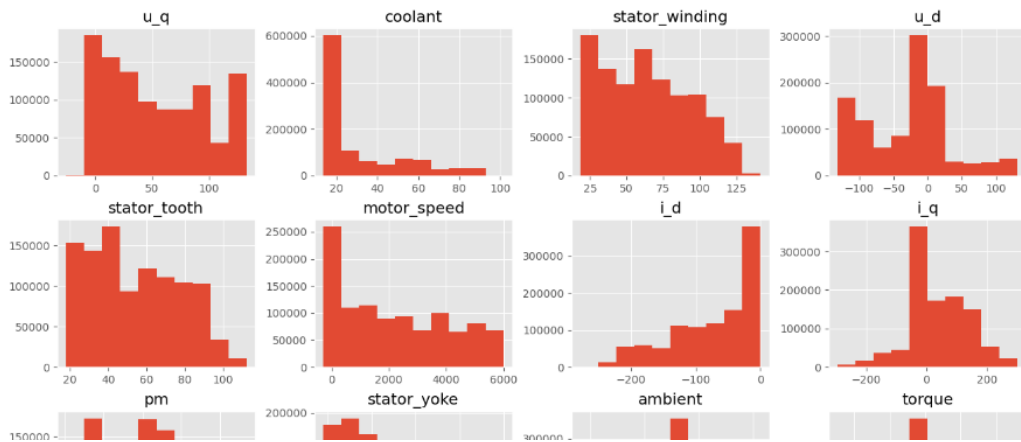


Figura 110. Exploración de las variables

Selección de Variables para el modelo

```
In [75]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#             'glazing_area', 'glazing_area_distribution']].values
```

```
In [76]: X = data.drop(['pm'],axis=1)
Y1 = data[['pm']]
```

```
In [77]: X.shape
```

```
Out[77]: (1048575, 12)
```

```
In [15]: #Y1 = data['heating_Load']
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data['cooling_Load']
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 111. Selección de variables

librerías para la ejecución del modelo

```
In [78]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 112. Librerías para ejecutar el modelo

Correlación de las variables

```
In [79]: #correlacion de las variables
import pandas as pds
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

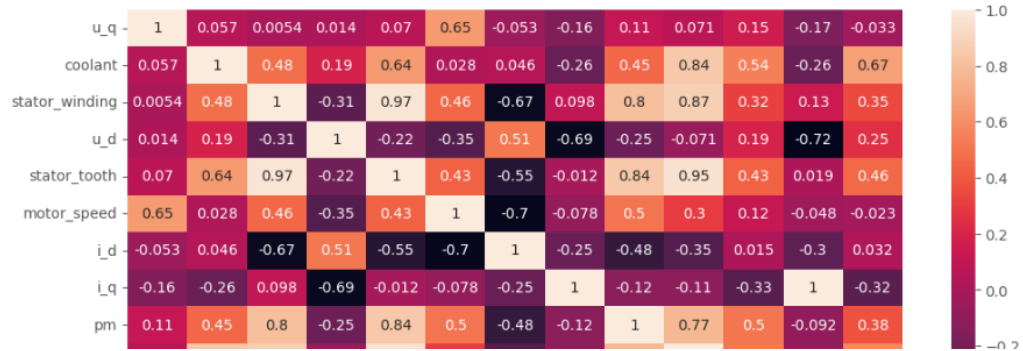


Figura 113. Correlación de variables

Pair Plots

```
In [82]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```



Figura 114. Visualización de Pair plots

Particionamiento de la data

```
In [83]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 115. Data particionada

Modelo Rigde

```
In [85]: from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 116. Modelo Ridge

Modelo para Y1 - temperatura del motor

```
In [89]: model_Y1 = Ridge()
model_Y1.fit(X_trainset_1, y_trainset_1)

Out[89]: Ridge()
```

Figura 117. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [90]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [95]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.8550050838209327
Mean absolute error: 5.2418121968809741
Mean squared error: 50.67724606539247
Root mean squared error: 7.118795267837983
Mean absolute percentage error: 0.10383835181442391
```

Figura 118. Resultados de las métricas del modelo Ridge (Train)

Métricas del Modelo (Testeo)

```
In [113]: pred_Y1t = model_Y1.predict(X_testset_1)

In [111]: r2_Y1 = r2_score(pred_Y1t, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1t)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1t)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1t)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1t)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.8540481543837648
Mean absolute error: 5.252260261415794
Mean squared error: 50.98057842426985
Root mean squared error: 7.1400685167769815
Mean absolute percentage error: 0.1039808036569578
```

Figura 119. Resultados de las métricas del modelo Ridge (Test)

RESULTADOS DE ALGORITMO RIDGE

R squared – R²

Ridge- Temperatura del motor

Tabla 47. Métrica de evaluación temperatura del motor – R squared Ridge

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$	85.40%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico, se obtiene un valor de R squared= 85.40% haciendo uso del **algoritmo Ridge**.

Mean absolute error

Ridge- Temperatura del motor

Tabla 48. Métrica de evaluación temperatura del motor — Mean absolute error Ridge

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	5.25

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 5.25 haciendo uso del algoritmo Ridge.

Mean squared error

Ridge - Temperatura del motor

Tabla 49. Métrica de evaluación temperatura del motor — Mean squared error Ridge

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	50.98

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 50.98 haciendo uso del algoritmo Ridge.

Root mean squared error

Ridge- Temperatura del motor

Tabla 50. Métrica de evaluación temperatura del motor — Root mean squared error Ridge

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{r}_n - r_n)^2}{N}}$	7.14

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 7.14 haciendo uso del algoritmo Ridge.

Mean absolute percentage error

Ridge- Temperatura del motor

Tabla 51. Métrica de evaluación temperatura del motor — Mean absolute percentage error Ridge

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a - b}{a} \right }{n} * 100\%$	0.10%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.10% haciendo uso del algoritmo Ridge.

➤ XG BOOST REGRESSOR (Bentéjac, Csörgő y Martínez-Muñoz, 2021)

```
CONECTAR CON LA BASE DE DATOS

•[1]: #Librerías para el desarrollo del proyecto
import matplotlib.pyplot as plt #Librería para graficas
from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerías pandas para el manejo de los datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

#enlazando los datos en el archivo excel

import os #Ficheros de directorios
#mainpath = "C:/PROYECTO_DE_MOTOR_ELECTRICO/data"
#filename = "measures_v2.csv" #Nombre del archivo
#fullpath = os.path.join(mainpath, filename)

#CARGAR TUS BASES
#data2 = pd.read_csv(fullpath)

#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
data = pd.read_csv('..data/datos_motor.csv')
#C:\PROYECTO_DE_MOTOR_ELECTRICO\data
#C:\PROYECTO_NIVEL_DE_SEGURIDAD\data
```

Figura 120. Conexión con la base de datos

Mostrar Datos

```
In [5]: data
Out[5]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows × 13 columns

Figura 121. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [6]: data.shape
Out[6]: (1048575, 13)
```

Figura 122. Visualización de la cantidad de filas y columnas

Cantidad de columnas que tienen valores nulos

```
In [7]: data.isnull().sum()
```

```
Out[7]: u_q          0  
        coolant      0  
        stator_winding 0  
        u_d          0  
        stator_tooth  0  
        motor_speed   0  
        i_d          0  
        i_q          0  
        pm           0  
        stator_yoke   0  
        ambient       0  
        torque        0  
        profile_id    0  
        dtype: int64
```

Figura 123. Presentación de la cantidad de columnas que tienen valores nulos

verificando valor perdido

```
In [9]: #numero de missing por columnas  
        data.isnull().sum()
```

```
Out[9]: u_q          0  
        coolant      0  
        stator_winding 0  
        u_d          0  
        stator_tooth  0  
        motor_speed   0  
        i_d          0  
        i_q          0  
        pm           0  
        stator_yoke   0  
        ambient       0  
        torque        0  
        profile_id    0  
        dtype: int64
```

Figura 124. Verificación del valor perdido

Datos Limpios para ser procesado por el algoritmo

```
In [11]: import missingno as msno  
         msno.matrix(data)
```

```
Out[11]: <AxesSubplot:>
```

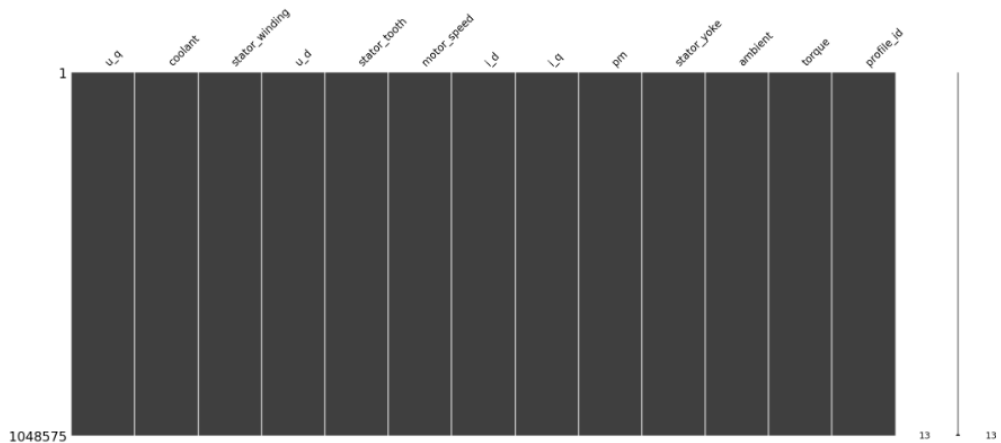


Figura 125. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [16]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data.hist()
plt.show()
```

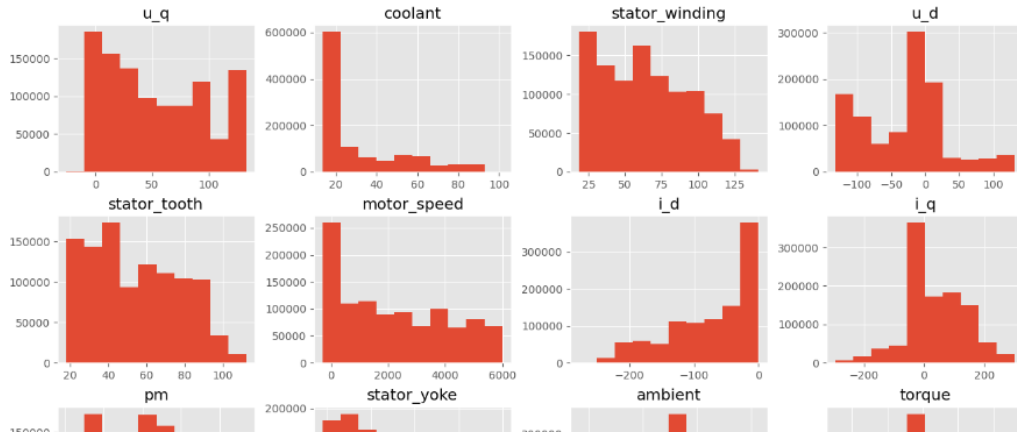


Figura 126. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',
#           'glazing_area', 'glazing_area_distribution']].values
```

```
In [17]: X = data.drop(['pm'],axis=1)
Y1 = data[['pm']]
```

```
In [18]: X.shape
```

```
Out[18]: (1048575, 12)
```

```
In [15]: #Y1 = data['heating_Load']
#Y1.shape
```

```
Out[15]: (768,)
```

```
In [16]: #Y2 = data['cooling_Load']
#Y2.shape
```

```
Out[16]: (768,)
```

Figura 127. Selección de variables

librerías para la ejecución del modelo

```
In [19]: from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.preprocessing import label_binarize
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Figura 128. Librerías para ejecutar el modelo

Correlación de las variables

```
In [20]: #correlacion de las variables
import pandas as pds
import seaborn as sns
import matplotlib.pyplot as plt
data
corr_df = data.corr(method='pearson')

plt.figure(figsize=(12,6))
sns.heatmap(corr_df, annot=True)
plt.show()
```



Figura 129. Correlación de variables

Pair Plots

```
In [21]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

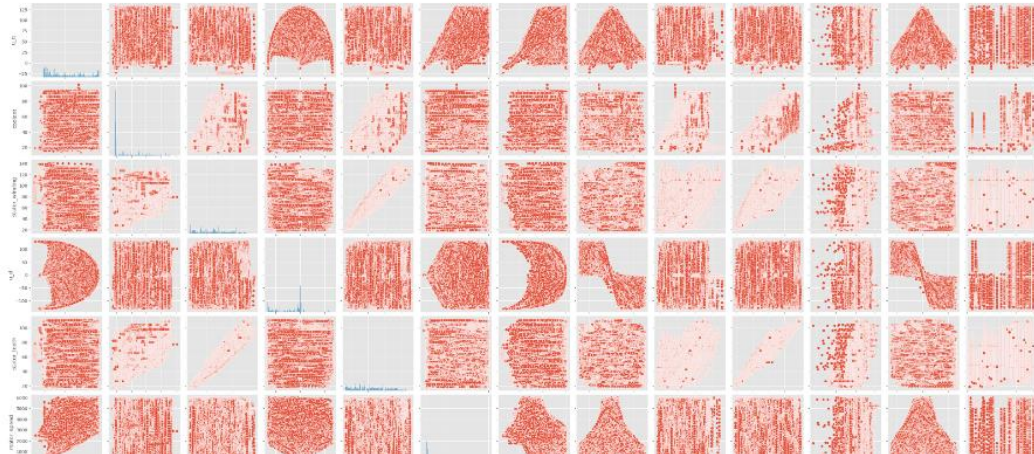


Figura 130. Visualización de Pair plots

Particionamiento de la data

```
In [51]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print(X_testset_1.shape, y_testset_1.shape)

(838860, 12) (838860, 1)
(209715, 12) (209715, 1)
```

Figura 131. Data particionada

Modelo Xgboost

```
In [77]: from xgboost import XGBRegressor
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

Figura 132. Modelo XG Boost

Modelo para Y1 - temperatura del motor

```
In [78]: model_Y1 = XGBRegressor(objective='reg:squarederror')
model_Y1.fit(X_trainset_1, y_trainset_1)

Out[78]: XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
    colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
    early_stopping_rounds=None, enable_categorical=False,
    eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
    importance_type=None, interaction_constraints='',
    learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
    max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
    missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
    num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
    reg_lambda=1, ...)
```

Figura 133. Modelo para Y1-temperatura del motor

Métricas del Modelo (Entrenamiento)

```
In [79]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [80]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.9927390081454855
Mean absolute error: 1.1696074412206834
Mean squared error: 2.8633141548772656
Root mean squared error: 1.6921330192621578
Mean absolute percentage error: 0.02192754771121267
```

Figura 134. Resultados de las métricas del modelo XG Boost (Train)

Métricas del Modelo (Testeo)

```
In [83]: pred_Y1 = model_Y1.predict(X_testset_1)

In [84]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
#msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
#print("Mean squared Log error:", msle_Y1)
#print("Root mean squared Log error:", sqrt(msle_Y1))

R2 score: 0.9924603071973742
Mean absolute error: 1.189746013911228
Mean squared error: 2.9816343330404194
Root mean squared error: 1.7267409571329508
Mean absolute percentage error: 0.02225452654834214
```

Figura 135. Resultados de las métricas del modelo XG Boost (Train)

RESULTADOS DEL ALGORITMO XGBOOST REGRESSOR

R squared – R²

Tabla 52. Métrica de evaluación temperatura del motor – R squared XG Boost

Ítem	Indicador	Medida	Fórmula	R squared
1	R squared	Razón	$R^2 = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$	99.25%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de R squared= 99.25% haciendo uso del **algoritmo XG Boost**.

Mean absolute error

XG Boost- Temperatura del motor

Tabla 53. Métrica de evaluación temperatura del motor — Mean absolute error XG Boost

Ítem	Indicador	Medida	Fórmula	MAE
2	MAE	Razón	$E_i = \frac{1}{n} \sum_{j=1}^n P_{((ij))} - \sum_{j=1}^n P_{((ij))} - T_j $	1.19

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo híbrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute error= 1.19 haciendo uso del algoritmo XG Boost.

Mean squared error

XG Boost - Temperatura del motor

Tabla 54. Métrica de evaluación temperatura del motor — Mean squared error XG Boost

Ítem	Indicador	Medida	Fórmula	MSE
3	MSE	Razón	$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i, \hat{y}_i)^2$	2.98

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean squared error= 2.98 haciendo uso del algoritmo XG Boost

Root mean squared error

XG Boost- Temperatura del motor

Tabla 55. Métrica de evaluación temperatura del motor — Root mean squared error XG Boost

Ítem	Indicador	Medida	Fórmula	RMSE
4	RMSE	Razón	$RMSE = \sqrt{\frac{\sum_{n=1}^N (\hat{f}_n - r_n)^2}{N}}$	1.73

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Root mean squared error = 1.73 haciendo uso del algoritmo XG Boost.

Mean absolute percentage error

XG Boost- Temperatura del motor

Tabla 56. Métrica de evaluación temperatura del motor — Mean absolute percentage error XG Boost

Ítem	Indicador	Medida	Fórmula	MAPE
5	MAPE	Razón	$MAPE = \frac{\sum_{t=1}^n \left \frac{a - b}{a} \right }{n} * 100\%$	0.02%

Fuente: Elaboración propia

Interpretación: Al desarrollar un modelo hibrido para predecir la temperatura del motor eléctrico Raghavendra, se obtiene un valor de Mean absolute percentage error= 0.02% haciendo uso del algoritmo XG Boost.

ANEXO N° 14

DESARROLLO DE LOS MODELOS PROPUESTO

Tabla 57. Datos_motor Raghavendra:

u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke
Voltage (in V)	Coolant temperature (in °C)	Stator winding temperature (in °C) measured with thermocouples	Voltage d-component measurement in dq-coordinate with thermocouples	Stator tooth temperature (in °C) measured with thermocouples	Motor speed (in rpm)	Current d-component measurement in dq-coordinate	Current q-component measurement in dq-coordinate	Permanent magnet temperature (in °C) measured with thermocouples and transmitted wirelessly via	Stator yoke temperature (in °C) measured with thermocouples

Fuente: kaggle-raghavendra electric motor dataset.

MODELO 1 – STACKING 1

CONECTAR CON LA BASE DE DATOS - Stacking1

```
In [2]: #Librerias para el desarrollo del proyecto
#Libreria para graficas #Matplotlib es una Libreria de Python especializada en La creacion de graficos en dos dimensiones.

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerias pandas para el manejo de los datos
# Pandas es una Libreria de Python especializada en el manejo y análisis de estructuras de datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

data = pd.read_csv('data/datos_motor.csv')

import numpy as np
#invocando a la Libreria de clasificacion
from sklearn.tree import DecisionTreeClassifier

In [3]: data
```

Figura 136. Conexión con la base de datos – Stacking 1

```
In [3]: data
```

```
Out[3]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459660	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 137. Visualización de los atributos de la temperatura del motor

Mostrar Datos

```
In [6]: data
```

Out[6]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 138. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [7]: data.shape
```

Out[7]: (1048575, 13)

Cantidad de columnas que tienen valores nulos

```
In [9]: data.isnull().sum()
```

Out[9]:

u_q	0
coolant	0
stator_winding	0
u_d	0
stator_tooth	0
motor_speed	0
i_d	0
i_q	0
pm	0
stator_yoke	0
ambient	0
torque	0
profile_id	0
dtype: int64	

```
In [10]: data.info()
```

Figura 139. Visualización de la cantidad de filas y columnas

verificando valor perdido

```
In [11]: #numero de missing por columnas  
data.isnull().sum()
```

```
Out[11]: u_q          0  
coolant     0  
stator_winding  0  
u_d         0  
stator_tooth  0  
motor_speed  0  
i_d         0  
i_q         0  
pm          0  
stator_yoke  0  
ambient     0  
torque      0  
profile_id  0  
dtype: int64
```

```
In [12]: #Columnas con missing  
null_columns=data.columns[data.isnull().any()]  
  
print(null_columns)  
Index([], dtype='object')
```

Figura 140. Verificación de valores perdidos.

Datos Limpios para ser procesado por el algoritmo

```
In [13]: data2=data.drop(columns = ["profile_id"])
```

```
In [14]: import missingno as msno  
msno.matrix(data2)
```

```
Out[14]: <AxesSubplot:>
```

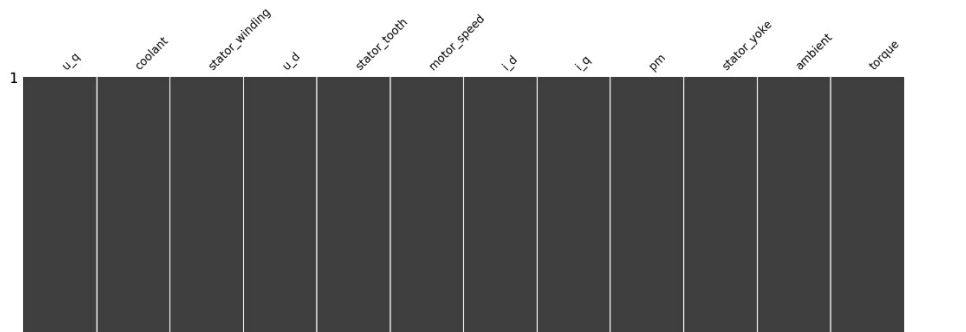


Figura 141. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [17]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data2.hist()
plt.show()
```

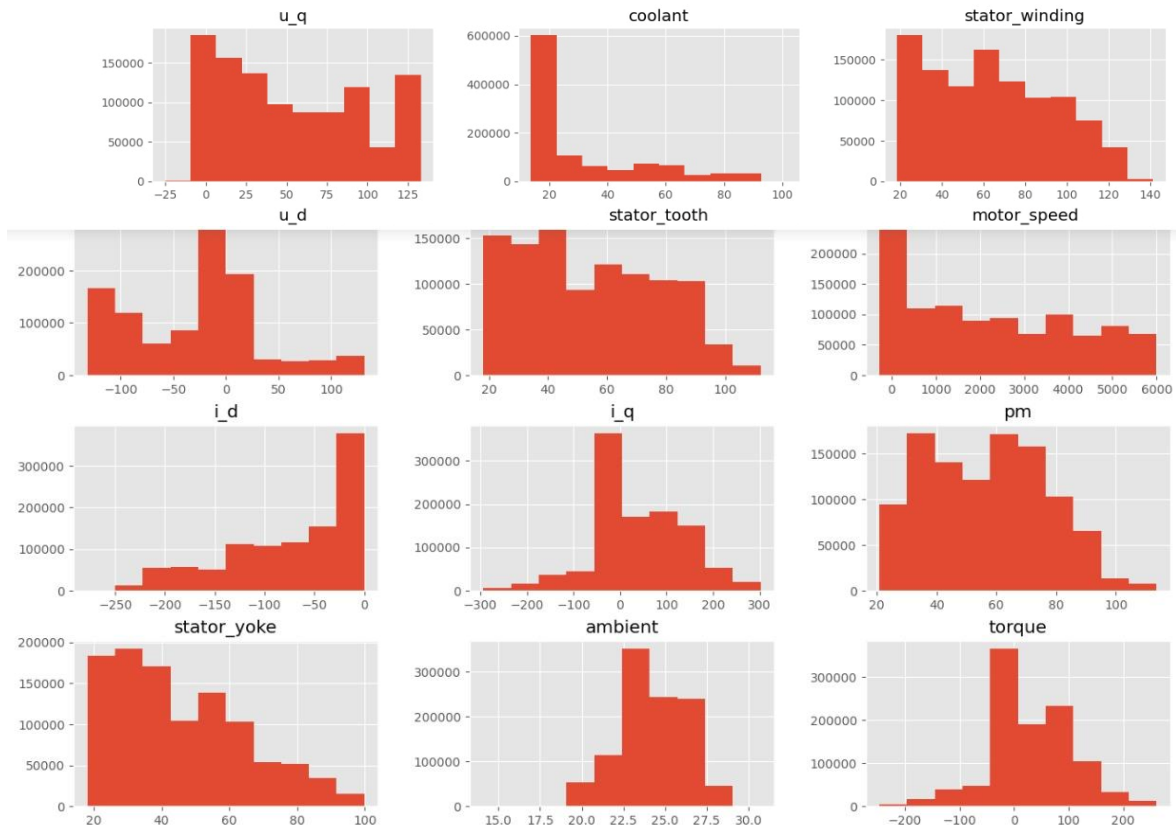


Figura 142. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',  
#          'glazing_area', 'glazing_area_distribution']].values
```

```
In [18]: X = data2.drop(['pm'],axis=1)  
Y1 = data2[['pm']]
```

```
In [19]: X
```

Out[19]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	49.492062	25.878452	-162.685047

Figura 143. Selección de variables

liberías para la ejecución del modelo

```
In [24]: from sklearn.metrics import roc_curve, auc  
from sklearn import datasets  
from sklearn.multiclass import OneVsRestClassifier  
from sklearn.svm import LinearSVC  
from sklearn.preprocessing import label_binarize  
from sklearn.model_selection import train_test_split  
import matplotlib.pyplot as plt
```

Correlación de las variables

```
In [25]: #correlacion de las variables  
import pandas as pds  
import seaborn as sns  
import matplotlib.pyplot as plt  
data  
corr_df = data.corr(method='pearson')  
  
plt.figure(figsize=(12,6))  
sns.heatmap(corr_df, annot=True)  
plt.show()
```

Figura 144. Librerías para ejecutar el modelo

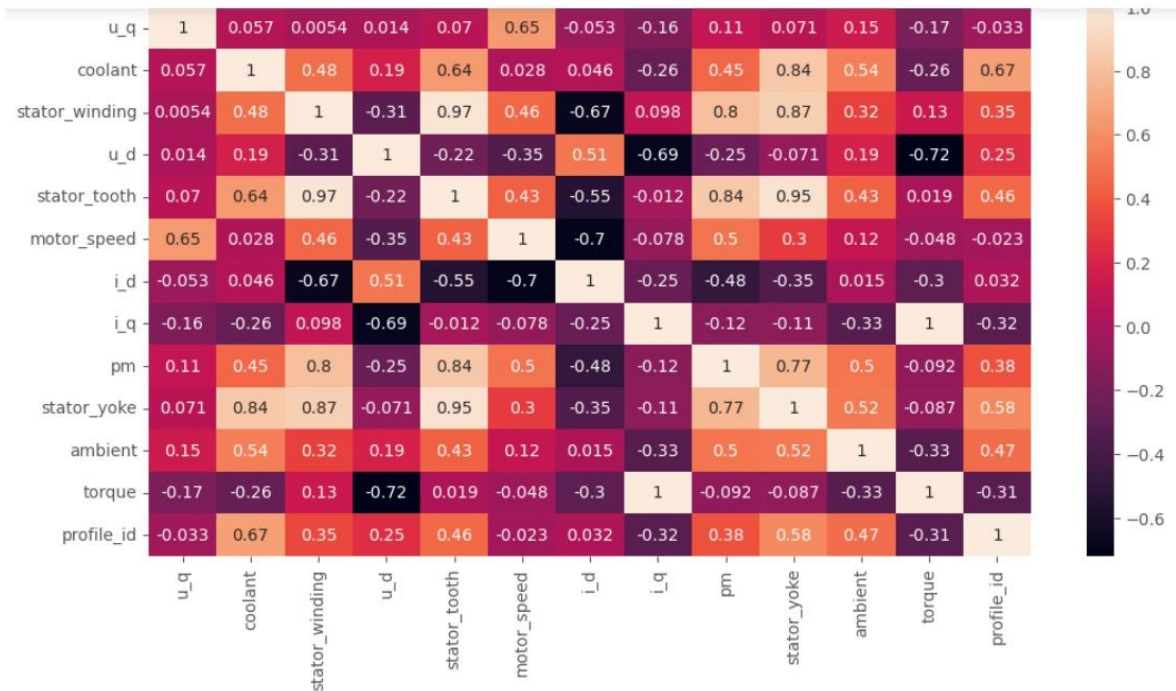


Figura 145. Correlación de variables

Pair Plots

```
In [26]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

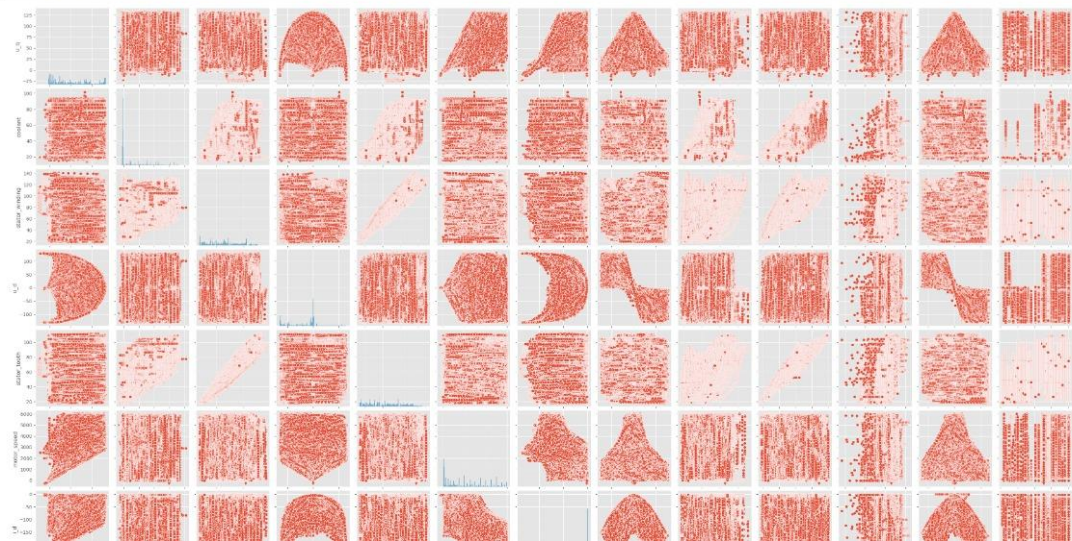


Figura 146. Visualización de Pair plots

Particionamiento de la data

```
In [51]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print (X_testset_1.shape, y_testset_1.shape)

(838860, 11) (838860, 1)
(209715, 11) (209715, 1)
```

Figura 147. Data particionada

Librerías de Modelos

```
In [47]: from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score

from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from xgboost import XGBRegressor

# metricas

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

```
In [ ]: #stacking 1
#model_Y1 = DecisionTreeRegressor()
# model_Y1 = LinearRegression()
# model_Y1 = ElasticNet()
# model_Y1 = KNeighborsRegressor()

# salida : RandomForestRegressor

# stacking 2
# model_Y1 = Lasso()
# model_Y1 = RandomForestRegressor()
# model_Y1 = Ridge()
# model_Y1 = XGBRegressor(objective='reg:squarederror')

# salida : RandomForestRegressor
```

Figura 148. Librería de modelos

probando modelo cambiando stacking

```
In [56]: # get a list of models to evaluate
def get_models():
    models = dict()
    # models['Lr'] = LogisticRegression()
    models['Arbol'] = DecisionTreeRegressor()
    models['Regression'] = LinearRegression()
    models['Elastic'] = ElasticNet()
    models['Knn'] = KNeighborsRegressor()

    return models

In [57]: # evaluate a given model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X_trainset_1, y_trainset_1, scoring='accuracy', cv=cv, n_jobs=-1, error_score='raise')
    return scores

In [58]: from statistics import mean
# define dataset
#X, Y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()

for name, model in models.items():

    scores = evaluate_model(model, X_trainset_1, y_trainset_1)

    results.append(scores)
```

Figura 149. Modelo Stacking 1

Modelo para Y1 - temperatura del motor

```
In [73]: #model_Y1 = DecisionTreeRegressor()
#model_Y1.fit(X_trainset_1, y_trainset_1)
#model_Y1 = model.fit(X_trainset_1, y_trainset_1)
```

Métricas del Modelo StackingRegresion (Entrenamiento)

```
In [79]: pred_Y1 = model_Y1.predict(X_trainset_1)

In [81]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)

print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.9998556308674367
Mean absolute error: 0.0679756245432824
Mean squared error: 0.05774082654921131
Root mean squared error: 0.24029320953620664
Mean absolute percentage error: 0.0012644652610229574
Mean squared log error: 2.003675978356455e-05
Root mean squared log error: 0.004476243937003942
```

Figura 150. Resultados de las métricas del modelo Stacking 1 (Train)

Métricas del Modelo (Testeo)

```
In [83]: pred_Y1 = model_Y1.predict(X_testset_1)
```

```
In [84]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)
```

```
print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))
```

```
R2 score: 0.9986874304304758
Mean absolute error: 0.15142595152027402
Mean squared error: 0.5265245326080147
Root mean squared error: 0.7256201021250822
Mean absolute percentage error: 0.002820048344134324
Mean squared log error: 0.00017780971339776568
Root mean squared log error: 0.013334530865304775
```

Figura 151. Resultados de las métricas del modelo Stacking1 (Test)

MODELO 2 – STACKING2

CONECTAR CON LA BASE DE DATOS - Stacking2

```
In [2]: #Librerías para el desarrollo del proyecto
#Librería para graficas #Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones.

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap
import matplotlib.patches as mpatches
from matplotlib.ticker import StrMethodFormatter
import seaborn as sb

#Librerías pandas para el manejo de los datos
# Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos
import pandas as pd
import pandas as pq
import pandas as pf
import pandas as filtro_filas

data = pd.read_csv('data/datos_motor.csv')

import numpy as np
#invocando a la librería de clasificación
from sklearn.tree import DecisionTreeClassifier
```

```
In [3]: data
```

Figura 152. Conexión con la base de datos – Stacking 2

```
In [3]: data
Out[3]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.869611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459660	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 153. Visualización de los atributos de la temperatura del motor

Mostrar Datos

```
In [6]: data
```

```
Out[6]:
```

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	pm	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	24.554214	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	24.538078	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	24.544693	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	24.554018	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	24.565397	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	66.739855	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	66.864130	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	66.918489	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	66.985567	49.492062	25.878452	-162.685047
1048574	50.837811	24.559166	91.662692	106.296999	68.699325	1874.340648	-83.758749	-196.992317	67.326579	49.520132	25.882027	-159.677618

1048575 rows x 13 columns

Figura 154. Visualización de los atributos de la temperatura del motor

Mostrar cantidad de filas y columnas

```
In [7]: data.shape
```

```
Out[7]: (1048575, 13)
```

Cantidad de columnas que tienen valores nulos

```
In [9]: data.isnull().sum()
```

```
Out[9]:
```

u_q	0
coolant	0
stator_winding	0
u_d	0
stator_tooth	0
motor_speed	0
i_d	0
i_q	0
pm	0
stator_yoke	0
ambient	0
torque	0
profile_id	0
dtype: int64	

```
In [10]: data.info()
```

Figura 155. Visualización de la cantidad de filas y columnas

verificando valor perdido

```
In [11]: #numero de missing por columnas  
data.isnull().sum()
```

```
Out[11]: u_q      0  
coolant    0  
stator_winding  0  
u_d      0  
stator_tooth  0  
motor_speed  0  
i_d      0  
i_q      0  
pm        0  
stator_yoke  0  
ambient    0  
torque     0  
profile_id  0  
dtype: int64
```

```
In [12]: #Columnas con missing  
null_columns=data.columns[data.isnull().any()]
```

```
print(null_columns)  
Index([], dtype='object')
```

Figura 156. Verificación de valores perdidos.

Datos Limpios para ser procesado por el algoritmo

```
In [13]: data2=data.drop(columns = ["profile_id"])
```

```
In [14]: import missingno as msno  
msno.matrix(data2)
```

```
Out[14]: <AxesSubplot:>
```

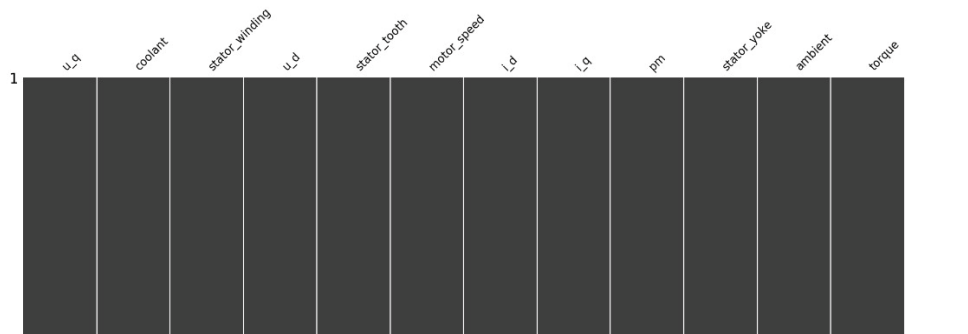


Figura 157. Visualización de datos limpios para ser procesado por el algoritmo

Analisis Exploratorio de las variables

```
In [17]: import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (16, 12)
plt.style.use('ggplot')
data2.hist()
plt.show()
```

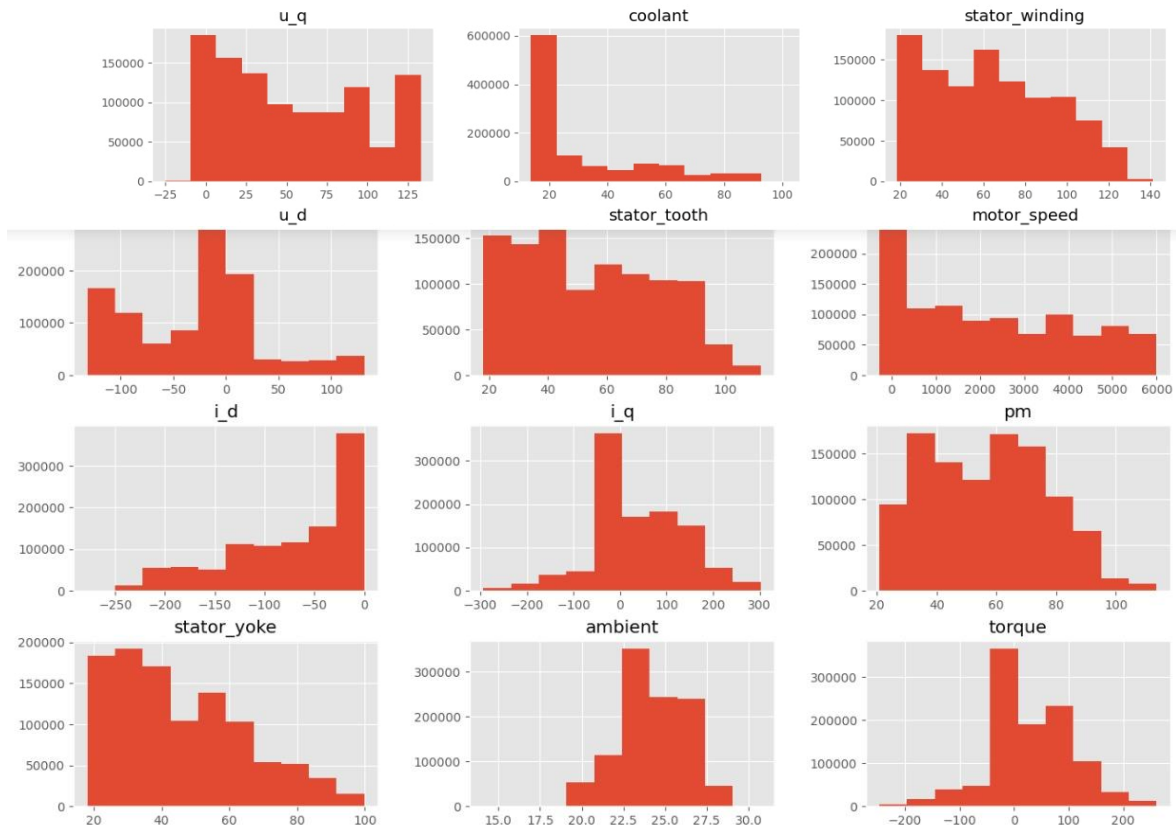


Figura 158. Exploración de las variables

Selección de Variables para el modelo

```
In [13]: #X = data[['relative_compactness', 'surface_area', 'wall_area', 'roof_area', 'overall_height', 'orientation',  
#          'glazing_area', 'glazing_area_distribution']].values
```

```
In [18]: X = data2.drop(['pm'],axis=1)  
Y1 = data2[['pm']]
```

```
In [19]: X
```

Out[19]:

	u_q	coolant	stator_winding	u_d	stator_tooth	motor_speed	i_d	i_q	stator_yoke	ambient	torque
0	-0.450682	18.805172	19.086670	-0.350055	18.293219	0.002866	0.004419	0.000328	18.316547	19.850691	0.187101
1	-0.325737	18.818571	19.092390	-0.305803	18.294807	0.000257	0.000606	-0.000785	18.314955	19.850672	0.245417
2	-0.440864	18.828770	19.089380	-0.372503	18.294094	0.002355	0.001290	0.000386	18.326307	19.850657	0.176615
3	-0.327026	18.835567	19.083031	-0.316199	18.292542	0.006105	0.000026	0.002046	18.330833	19.850647	0.238303
4	-0.471150	18.857033	19.082525	-0.332272	18.291428	0.003133	-0.064317	0.037184	18.326662	19.850639	0.208197
...
1048570	48.159700	24.322307	91.457177	119.363505	68.457606	2096.735442	-105.000718	-201.094224	49.465098	25.871140	-167.782798
1048571	49.347272	24.365932	91.492095	116.890796	68.455415	2039.850834	-98.726047	-201.925423	49.466701	25.868611	-166.929560
1048572	50.191015	24.445247	91.577188	114.053296	68.518171	1989.249199	-93.459560	-201.205594	49.467069	25.873463	-165.156581
1048573	50.476838	24.511613	91.629253	110.198509	68.617628	1928.984749	-88.545222	-199.463876	49.492062	25.878452	-162.685047

Figura 159. Selección de variables

librerías para la ejecución del modelo

```
In [24]: from sklearn.metrics import roc_curve, auc  
from sklearn import datasets  
from sklearn.multiclass import OneVsRestClassifier  
from sklearn.svm import LinearSVC  
from sklearn.preprocessing import label_binarize  
from sklearn.model_selection import train_test_split  
import matplotlib.pyplot as plt
```

Correlación de las variables

```
In [25]: #correlacion de las variables  
import pandas as pds  
import seaborn as sns  
import matplotlib.pyplot as plt  
data  
corr_df = data.corr(method='pearson')  
  
plt.figure(figsize=(12,6))  
sns.heatmap(corr_df, annot=True)  
plt.show()
```

Figura 160. Librerías para ejecutar el modelo

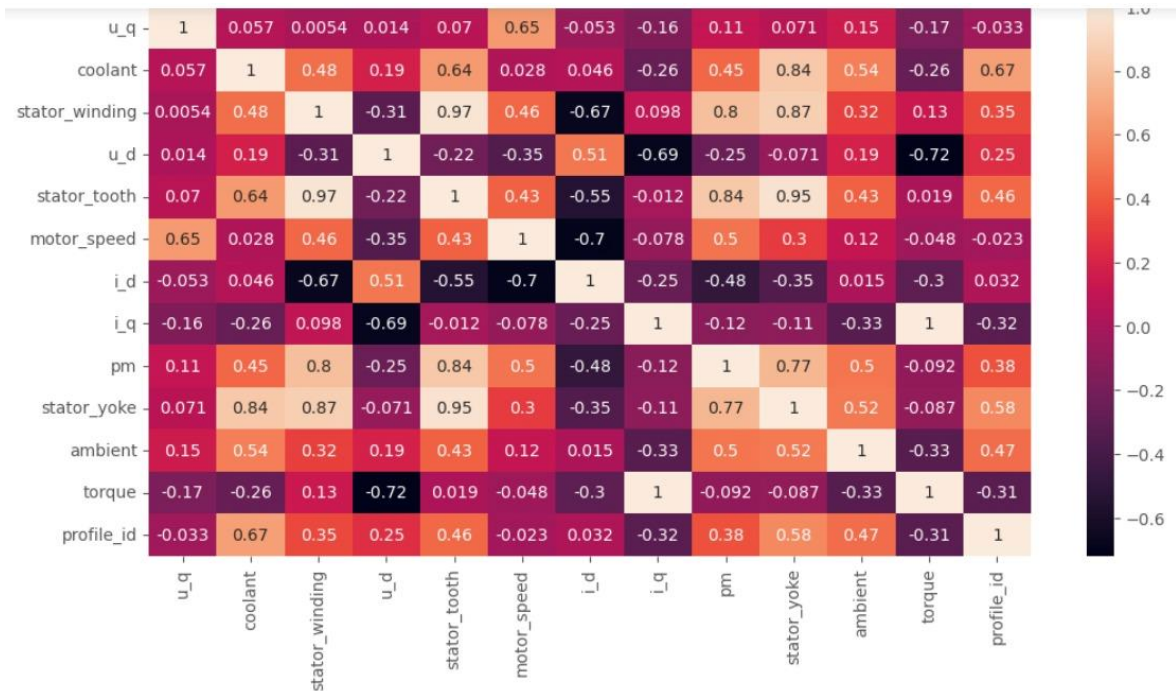


Figura 161. Correlación de variables

Pair Plots

```
In [26]: # check the distribution and relationship between variables
sns.pairplot(data)
plt.show()
```

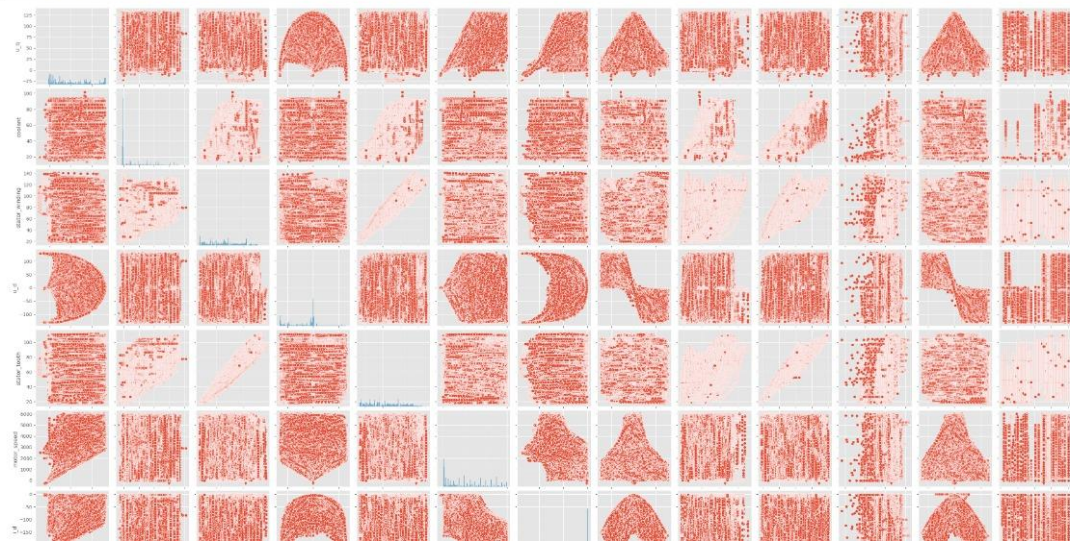


Figura 162. Visualización de Pair plots

Particionamiento de la data

```
In [51]: #particionando la data en 0.8 para entrenar y 0.2 para validar
from sklearn.model_selection import train_test_split
X_trainset_1, X_testset_1, y_trainset_1, y_testset_1 = train_test_split(X, Y1, test_size=0.2, random_state=0)
#X_trainset_2, X_testset_2, y_trainset_2, y_testset_2 = train_test_split(X, Y2, test_size=0.2, random_state=0)

print(X_trainset_1.shape, y_trainset_1.shape)
print (X_testset_1.shape, y_testset_1.shape)

(838860, 11) (838860, 1)
(209715, 11) (209715, 1)
```

Figura 163. Data particionada

Librerías del Modelos

```
In [47]: from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score

from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from xgboost import XGBRegressor

# metricas

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_log_error
from math import sqrt
```

```
In [ ]: #stacking 1
#model_Y1 = DecisionTreeRegressor()
# model_Y1 = LinearRegression()
# model_Y1 = ElasticNet()
# model_Y1 = KNeighborsRegressor()

# salida : RandomForestRegressor

# stacking 2
# model_Y1 = Lasso()
# model_Y1 = RandomForestRegressor()
# model_Y1 = Ridge()
# model_Y1 = XGBRegressor(objective='reg:squarederror')

# salida : RandomForestRegressor
```

Figura 164. Librería de modelos

probando modelo cambiando stacking 2

```
In [164]: # get a list of models to evaluate
def get_models():
    models = dict()
    # models['Lr'] = LogisticRegression()
    models['Lasso'] = Lasso()
    models['Ranforest'] = LinearRegression()
    models['Rigde'] = Ridge()
    models['Xgboost'] = XGBRegressor(objective='reg:squarederror')

    return models

In [178]: # evaluate a given model using cross-validation
def evaluate_model(model, X, y):
    cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
    scores = cross_val_score(model, X_trainset_1, y_trainset_1, scoring='r2', cv=cv, n_jobs=-1, error_score='raise')
    return scores

In [179]: from statistics import mean
# define dataset
#X, Y = get_dataset()
# get the models to evaluate
models = get_models()
# evaluate the models and store results
results, names = list(), list()

for name, model in models.items():
    scores = evaluate_model(model, X_trainset_1, y_trainset_1)
    results.append(scores)
    names.append(name)
print('> %s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

Figura 165. Modelo Stacking 2

Modelo para Y1 - temperatura del motor

```
In [132]: #model_Y1 = DecisionTreeRegressor()
#model_Y1.fit(X_trainset_1, y_trainset_1)
#model_Y1= model.fit(X_trainset_1,y_trainset_1)
```

Métricas del Modelo StackingRegresion (Entrenamiento)

```
In [138]: pred_Y1 = model_Y1.predict(X_trainset_1)
```

```
In [139]: r2_Y1 = r2_score(pred_Y1, y_trainset_1)
mae_Y1 = mean_absolute_error(y_trainset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_trainset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_trainset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_trainset_1, pred_Y1)
```

```
print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.9876110646066429
Mean absolute error: 1.4681574314956218
Mean squared error: 4.930434989240531
Root mean squared error: 2.2204582836073574
Mean absolute percentage error: 0.026945969065740004
Mean squared log error: 0.001594375649347094
Root mean squared log error: 0.039929633724179014
```

Figura 166. Resultados de las métricas del modelo Stacking 2 (Train)

Métricas del Modelo (Testeo)

```
In [141]: pred_Y1 = model_Y1.predict(X_testset_1)
```

```
In [142]: r2_Y1 = r2_score(pred_Y1, y_testset_1)
mae_Y1 = mean_absolute_error(y_testset_1, pred_Y1)
mse_Y1 = mean_squared_error(y_testset_1, pred_Y1)
mape_Y1 = mean_absolute_percentage_error(y_testset_1, pred_Y1)
msle_Y1 = mean_squared_log_error(y_testset_1, pred_Y1)
```

```
print("R2 score:", r2_Y1)
print("Mean absolute error:", mae_Y1)
print("Mean squared error:", mse_Y1)
print("Root mean squared error:", sqrt(mse_Y1))
print("Mean absolute percentage error:", mape_Y1)
print("Mean squared log error:", msle_Y1)
print("Root mean squared log error:", sqrt(msle_Y1))

R2 score: 0.9867996080447217
Mean absolute error: 1.5206088152847925
Mean squared error: 5.266880640047638
Root mean squared error: 2.2949685488144795
Mean absolute percentage error: 0.027858639986228874
Mean squared log error: 0.001696428678508253
Root mean squared log error: 0.04118772485229371
```

```
In [143]: plt.figure(figsize=(4, 4))
plt.scatter(y_testset_1, pred_Y1)
plt.xlabel('True')
plt.ylabel('Predicted')
plt.title('Temperatura del motor Staxking (Test)')
plt.show()
```

Figura 167. Resultados de las métricas del modelo Stacking2 (Test)