Accidentes Urbanos y Clima: Una Perspectiva Predictiva mediante Aprendizaje Automático

F. Morales López

Maestría en Ciencia de Datos, Facultad de Ciencias Físico Matemáticas, UANL

DOI: XXXXXXXX 1943-0655/

Index Terms: aprendizaje automático, datos meteorológicos, seguridad vial.

1. Introducción

En un contexto global, las estadísticas de la Organización Mundial de la Salud (OMS) subrayan la magnitud del impacto de los accidentes de tráfico, con aproximadamente 1.3 millones de personas falleciendo anualmente y entre 20 y 50 millones sufriendo lesiones no mortales. Estas cifras resaltan la urgencia de abordar la seguridad vial y destacan la importancia de iniciativas de investigación que buscan comprender y mitigar estos eventos a nivel local.

La combinación de datos de incidentes viales con información climatológica abre nuevas posibilidades de análisis. La identificación de patrones y factores que contribuyen a los incidentes, especialmente en condiciones meteorológicas adversas, podría ofrecer perspectivas clave para el desarrollo de medidas preventivas y estrategias de gestión del tráfico.

Los datos de incidentes viales provienen de la Secretaría de Seguridad y Protección a la Ciudadanía, asegurando la calidad y precisión necesarias para llevar a cabo un análisis robusto y confiable.

2. Conjunto de datos

El conjunto de datos utilizado proviene de un agrupamiento diario de accidentes en el municipio de Monterrey [1], proporcionado por la Secretaría de Seguridad y Protección a la Ciudadanía. Los datos climatológicos diarios fueron obtenidos de la API histórica de la página web Open Meteo [2].

En nuestro conjunto de datos, la variable es_dia_laborable actúa como un indicador binario para identificar si un día es laborable o no. Un valor de 1 representa un día laborable, mientras que un valor de 0 indica lo contrario. Sin embargo, al profundizar en los datos, nos enfrentamos a una discrepancia significativa: el valor 1 tiene un número considerablemente mayor de ocurrencias que el valor 0, casi el doble de veces.

Este desbalance, donde una clase (en este caso, el día laborable) está sobrerrepresentada en comparación con la otra, tuvo consecuencias significativas en el análisis predictivo y los modelos que se construyen sobre estos datos.

Esta disparidad en la frecuencia de las clases es exclusiva de la variable es_dia_laborable y no se observa en otras variables del conjunto de datos.

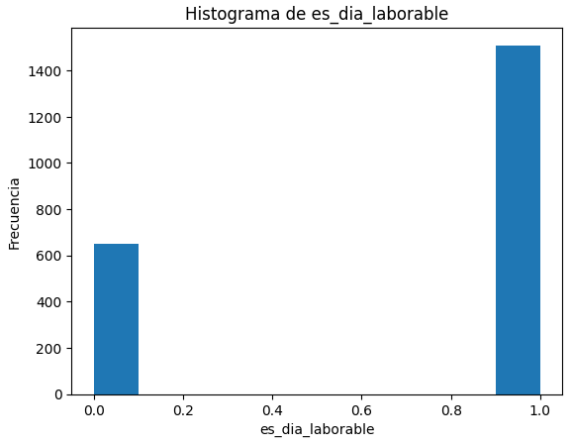

Figura 1: Frecuencia de las clases es_dia_laborable

2.1. Reajustar un conjunto de datos desequilibrado

Dependiendo del tamaño y tipo de los datos, el siguiente paso es elegir un método para corregir el desequilibrio entre las clases:

- Realizar submuestreo aleatorio para reducir la cantidad de la clase mayoritaria, crean un subconjunto del conjunto de datos original al eliminar instancias (generalmente, instancias de la clase mayoritaria).
- Realizar sobremuestreo aleatorio para aumentar la cantidad de la clase minoritaria, crean un superconjunto del conjunto de datos original al replicar algunas instancias o crear nuevas instancias a partir de las existentes.

En general, se puede decir que el submuestreo aleatorio es una técnica más robusta y versátil que el sobremuestreo aleatorio. Esto se debe a que el submuestreo aleatorio no introduce ruido en los datos, mientras que el sobremuestreo aleatorio puede hacerlo.

Ventajas del submuestreo aleatorio

El submuestreo aleatorio tiene las siguientes ventajas:

- No introduce ruido en los datos: El submuestreo aleatorio simplemente elimina puntos de datos de la clase mayoritaria, sin modificar los datos de la clase minoritaria. Esto significa que el submuestreo aleatorio no introduce ruido en los datos, lo que puede mejorar la precisión de los modelos de aprendizaje automático.
- Es menos susceptible a la sobreajuste: El submuestreo aleatorio puede reducir el riesgo de sobreajuste al reducir el tamaño del conjunto de entrenamiento.
- Es más eficiente: El submuestreo aleatorio es generalmente más eficiente que el sobremuestreo aleatorio, ya que no requiere generar nuevos datos.

Imbalanced Learning: Techniques for Learning from Imbalanced Data Sets [3]

Después, se aplicaron técnicas de filtrado para encontrar las variables que mas influyeran en los distintos modelos a probar.

1. Winddirection 10m Dominante: Describe la dirección dominante del viento a 10 metros de altura.
2. Semana del Mes: Indica la semana del mes del día registrado.
3. Semana del Año: Representa la semana del año correspondiente al día registrado.
4. Es Festivo en México: Indica si el día registrado es festivo en México.

5. Año: Muestra el año del día registrado.
6. Día de la Semana: Refleja el día de la semana del día registrado.
7. Total de Accidentes: Representa el número total de accidentes en el agrupamiento diario.

3. Aprendizaje no supervisado

El aprendizaje no supervisado es un tipo de problema en aprendizaje maquina en el cual los datos de entrenamiento consisten en un conjunto de vectores de entrada pero no hay valores objetivo correspondientes. La idea detrás de este tipo de aprendizaje es agrupar la información basándose en similitudes, patrones y diferencias.

El papel de un algoritmo de aprendizaje no supervisado es descubrir la estructura subyacente de un conjunto de datos no etiquetado por sí mismo.

3.1. K-medias y Método del codo

K-medias es un algoritmo de agrupamiento que busca organizar un conjunto de datos en k grupos, donde k es un número predefinido. La idea subyacente es minimizar la suma de las distancias cuadradas entre los puntos de datos y los centroides de sus respectivos grupos.

Esta técnica no solo proporciona una visión interesante durante el análisis, sino que también puede desempeñar un papel crucial como característica en algoritmos de aprendizaje supervisado.

En términos matemáticos, dada una colección de datos $\{x_1, x_2, \ldots, x_n\}$, y un conjunto de centroides $\{\mu_1, \mu_2, \ldots, \mu_k\}$, el objetivo es minimizar la función de costo conocida inercia:

$$J = \sum_{i=1}^{n} \min_{j} \|x_i - \mu_j\|^2. \tag{1}$$

En términos más simples, la inercia es una medida de lo cerca que están los puntos de datos de sus respectivos centroides. Cuanto más cerca estén los puntos de datos de sus centroides, menor será la inercia.

Este proceso de minimización se realiza iterativamente mediante dos pasos:
1. Asignación de grupos: Cada punto de datos se asigna al grupo cuyo centroide está más cercano. Esto se expresa como $c^{(i)} = \arg\min_j \|x_i - \mu_j\|^2$.

2. Actualización de centroides: Los centroides se recalculan como el promedio de los puntos asignados a cada grupo. Esto se expresa como $\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$, donde C_j es el conjunto de puntos asignados al grupo j.

El número óptimo de grupos k puede ser un parámetro crítico y a menudo se determina utilizando el método del codo. Este método implica ajustar el algoritmo para diferentes valores de k y graficar la variación del costo J. El punto en el que la disminución de J se desacelera, formando un çodo.[en] la gráfica, se elige como el número óptimo de grupos.

3.2. Aplicación Práctica: Estudio de Caso

En este trabajo, se aplicó el algoritmo de K-medias para determinar el número óptimo de grupos en nuestro conjunto de datos. El método del codo se utilizó para identificar el punto en el que la adición de más grupos ya no mejora significativamente la calidad del agrupamiento.

Definición de parámetros iniciales

Se definió una lista de números de grupos k que se probarían en el rango de 2 a 14 $k = \{2, 3, ..., 14\}$.

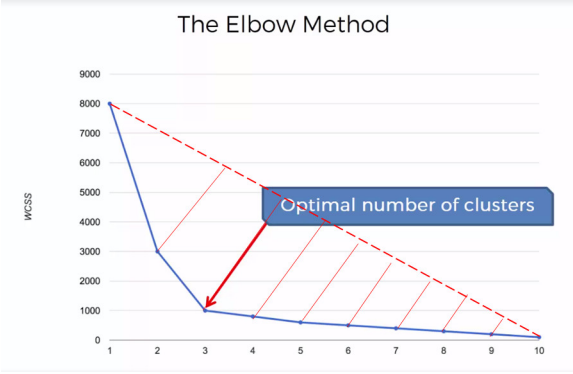

Figura 2: Visualización del método del Codo k.

Cálculo de inercias

Se inicializó una lista vacía para almacenar las inercias, que representan la suma de las distancias cuadradas de cada punto al centro de su cluster. Se utilizó un bucle para iterar sobre diferentes valores de k. Para cada k, se ajustó el modelo K-medias a los datos y se calculó la inercia, que se agregó a la lista de inertia.

Ajuste de la recta de mínimos cuadrados

Se creó un DataFrame que contenía el número de grupos (n_grupos) y sus respectivas inercias (inertia). Se aplicó el método de mínimos cuadrados para ajustar una recta a los datos, representando la relación entre el número de grupos y la inercia. Las distancias entre cada punto y la recta de ajuste se calcularon para evaluar la dispersión de los datos.

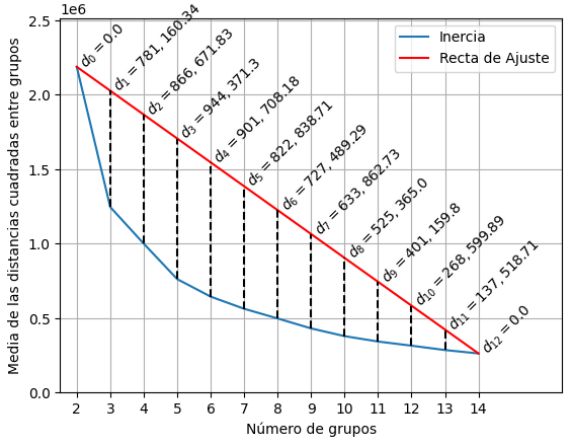

Figura 3: Relación entre el número de grupos y la inercia, con la recta de ajuste. El punto de codo indica el número óptimo de grupos.

Se identificó $k = 5$ como el punto en el que la distancia entre los puntos y la recta de ajuste era máxima. Este punto se consideró como el óptimo en términos del método del codo, indicando el número óptimo de grupos.

4. Aprendizaje Supervisado

El aprendizaje supervisado es una categoría de aprendizaje automático que busca entender la relación entre las entradas y las salidas. Las entradas se conocen como características o variables

X, mientras que la salida se denomina generalmente objetivo o variable y. El conjunto de datos que contiene tanto las características como el objetivo se conoce como datos etiquetados. Esta distinción es fundamental en el contraste entre el aprendizaje supervisado y el no supervisado en el aprendizaje automático.

A continuación, se presentan algunas implementaciones de algoritmos de aprendizaje supervisado que utilizaremos en nuestro modelo, empleando la biblioteca scikit-learn[4] de Python:

Bosque Aleatorio[5]: RandomForestRegressor()

El Bosque Aleatorio es un algoritmo de aprendizaje supervisado versátil que se puede utilizar para tareas de clasificación y regresión. Opera mediante la creación de una multitud de árboles de decisión y promediando sus predicciones. Esta característica hace que Bosque Aleatorio sea robusto y preciso, siendo menos propenso a ser influenciado por valores atípicos o ruido en los datos.

Máquina de Vectores de Soporte (SVR)[6]: SVR()

La regresión por soporte vectorial es un algoritmo de aprendizaje supervisado diseñado para tareas de regresión. Su funcionamiento implica la búsqueda de un hiperplano en el espacio de características que maximice el margen entre los puntos de datos. Aunque potente, SVR puede ser computacionalmente costoso de entrenar.

Regresión lineal[7]: LinearRegression()

La regresión lineal, un algoritmo sencillo y eficiente, se utiliza para tareas de regresión. Busca establecer una relación lineal entre las características de entrada y la variable objetivo. Sin embargo, su simplicidad puede traducirse en una menor precisión en comparación con algoritmos más complejos como Bosque Aleatorio o SVR, especialmente en conjuntos de datos complejos.

K vecinos más cercanos (K-NN)[8]: KNeighborsRegressor()

El algoritmo de los K vecinos más cercanos puede emplearse tanto para clasificación como para regresión. Opera identificando los K puntos de datos más similares al nuevo punto y promediando sus valores objetivo. A pesar de su simplicidad y eficiencia, K-NN puede ser sensible a la elección de K y la métrica de distancia utilizada.

AdaBoost[9]: AdaBoostRegressor()

AdaBoost es un algoritmo de aprendizaje en conjunto aplicable a clasificación y regresión. Entrena una secuencia de aprendices débiles y combina sus predicciones para producir una predicción final, mejorando así la precisión de cualquier algoritmo de aprendizaje automático.

Impulso Gradiente[9]: GradientBoostingRegressor()

El impulso por gradiente, similar a AdaBoost, entrena una secuencia de árboles de decisión en lugar de aprendices débiles. Este algoritmo es muy poderoso y puede lograr resultados de vanguardia en diversas tareas de aprendizaje automático.

Red neuronal artificial[10] (ANN): MLPRegressor()

Las redes neuronales artificiales (ANN) se inspiran en el cerebro humano y consisten en una red de nodos interconectados. MLPRegressor es una implementación de ANN utilizada para tareas de regresión, clasificación y procesamiento del lenguaje natural en el ámbito del aprendizaje automático.

Metodología para la determinación del mejor modelo de regresión

División de Datos

Los datos se dividen en conjuntos de entrenamiento y prueba utilizando la función train_test_split de la biblioteca scikit-learn. Diferentes tamaños de conjunto de prueba se consideran para evaluar el rendimiento del modelo en diferentes escenarios.

Modelos de Regresión

Se evaluan los modelos de regresión antes mencionados: Bosque Aleatorio, SVR, Regresión Lineal, K-NN, AdaBoost, Impulso Gradiente y Redes Neuronales (MLP).

Evaluación del Modelo

Se realiza un bucle sobre los modelos y se evalúan utilizando métricas como MAE, MAPE, MSE, RMSE, MSLE, R^2, y tiempo de ejecución. Estas métricas proporcionan una visión integral del rendimiento de cada modelo.

Iteraciones y Almacenamiento de Resultados

El proceso se repite varias veces para reducir la variabilidad. Los resultados de cada iteración se almacenan en un DataFrame para un análisis comparativo.

Resultados

MAPE	Tamaño de prueba	Modelo	R^2	MAE	MSE	MSLE	Tiempo (segundos)
16.11	0.30	Impulso Gradiente	0.71	9.66	160.89	12.68	0.10
16.11	0.30	Impulso Gradiente	0.71	9.66	160.89	12.68	0.10
16.11	0.10	Impulso Gradiente	0.71	9.66	160.89	12.68	0.10
16.08	0.20	Impulso Gradiente	0.71	9.65	160.91	12.69	0.10
16.10	0.40	Impulso Gradiente	0.71	9.65	160.96	12.69	0.10

Cuadro I: Se presentan las cinco iteraciones que muestran los menores errores de predicción en función del MAPE (Error Porcentual Absoluto Medio).

El MAPE, como métrica de evaluación, indica la precisión del modelo en la predicción de los datos, y en nuestro caso, el modelo de Impulso Gradiente ha demostrado consistentemente un rendimiento superior al producir el menor MAPE entre todos los modelos considerados.

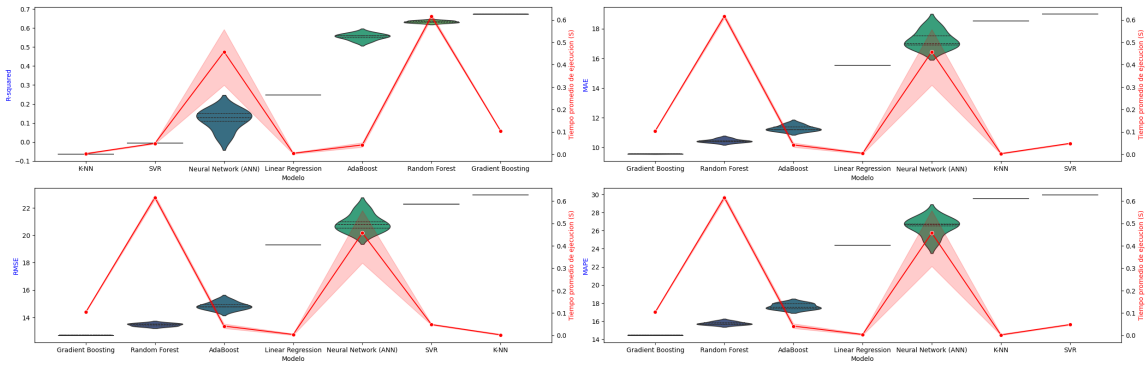

Figura 4: Gráficas de violín comparando distintas métricas de desempeño y tiempo de ejecución promedio entre modelos.

En base a los resultados obtenidos en nuestro estudio comparativo de modelos, hemos determinado que el modelo de Impulso Gradiente es la opción más favorable para nuestro problema. Esta conclusión se basa en 3 aspectos : presenta el MAPE (Error Porcentual Absoluto Medio) más

bajo, el R^2 más alto y el tiempo de ejecución promedio más eficiente en comparación con los otros modelos evaluados.

Además, al tener en cuenta el tiempo de ejecución, es evidente que el modelo de Impulso Gradiente no solo ofrece una mejora en términos de precisión, sino también en eficiencia computacional. El menor tiempo de ejecución implica que el modelo puede ser implementado de manera más rápida y con menor carga computacional, lo que resulta beneficioso tanto en términos de recursos como de tiempos de respuesta.

5. Diseño de experimentos para optimización de hiperparámetros

En este estudio, se llevó a cabo un diseño de experimentos utilizando un enfoque de búsqueda exhaustiva para optimizar los hiperparámetros de un modelo de regresión con refuerzo de gradiente.

5.1. Selección de hiperparámetros

Se identificaron varios hiperparámetros clave que podrían afectar el rendimiento del modelo. Estos incluyeron el número de estimadores, la profundidad máxima del árbol, la cantidad mínima de muestras requeridas para dividir un nodo, entre otros. La cuadrícula de hiperparámetros diseñada para explorar múltiples combinaciones posibles incluyó:

Número de estimadores:	[100, 177, 316, 562, 1000]
Profundidad máxima del árbol:	[5, 7, 9, 11]
Muestras mínimas para dividir un nodo:	[5, 10, 15]
Muestras mínimas en hoja:	[2, 4, 6]
Tasa de aprendizaje:	[0,01, 0,05, 0,1]
Submuestra:	[0,6, 0,9]
Características máximas:	['sqrt', 'log2']

5.2. Validación cruzada

Para evaluar el rendimiento de cada configuración de hiperparámetros, se empleó una estrategia de validación cruzada. Se utilizó _Stratified_K-Fold con 5 divisiones y se mantuvo el conjunto de datos aleatorio para garantizar la robustez de los resultados.

5.3. Métricas de evaluación

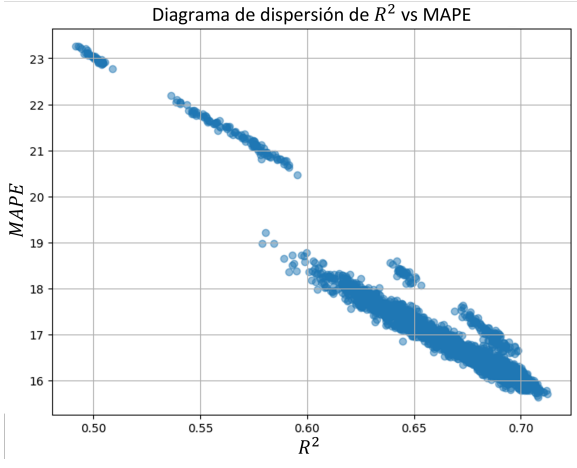

Figura 5: Relacion entre R^2 y MAPE para las distintas combinaciones de hiperparámetros

Para la evaluación de los modelos, se priorizaron las combinaciones que alcanzaron un coeficiente de determinación R^2 mayor al 70 % y un error porcentual absoluto medio (MAPE) menor al 16 %.

Esta selección se realizó para destacar únicamente aquellos modelos que mostraron un rendimiento sustancialmente alto en términos de ajuste y precisión predictiva.

5.4. Proceso de optimización

En el proceso de optimización, se exploraron diversas combinaciones de hiperparámetros utilizando una cuadrícula específica, donde se identificó que la configuración que generó los valores deseados de coeficiente de determinación (R^2) superior al 70 % y un error porcentual absoluto medio (MAPE) inferior al 16 % fue la siguiente:

Número de estimadores:	[562]
Profundidad máxima del árbol:	[7]
Muestras mínimas para dividir un nodo:	[5]
Muestras mínimas en hoja:	[4]
Tasa de aprendizaje:	[0,01]
Submuestra:	[0,6]
Características máximas:	['sqrt']

5.5. Resultados

MAPE	Modelo	R^2	Tiempo (segundos)
16.11	Impulso Gradiente No optimizado	0.71	0.10
15.96	Impulso Gradiente Optimizado	0.70	0.05

Cuadro II: Comparación del previo mejor modelo sin optimizar vs el mejor modelo con hiperparámetros optimizados.

La optimización de hiperparámetros mostró mejoras sustanciales en el rendimiento del modelo de impulso gradiente. Al comparar el modelo no optimizado con el modelo optimizado, se observa una reducción notable en el error porcentual absoluto medio (MAPE) del 16,11 % al 15,96 %, lo que indica una mejora en la precisión de las predicciones.

A pesar de una ligera disminución en el coeficiente de determinación (R^2) del 0.71 al 0.70 al implementar la optimización, se logró este incremento en la precisión sin comprometer significativamente la capacidad del modelo para explicar la variabilidad en los datos.

Además, la optimización de hiperparámetros permitió lograr estos avances manteniendo un tiempo de entrenamiento eficiente, reduciendo el tiempo necesario para ajustar el modelo de 0.10 segundos a 0.05 segundos.

6. Conclusión

En este estudio, hemos explorado la intersección entre los accidentes urbanos y las condiciones climáticas adversas utilizando técnicas de aprendizaje automático. Nuestra investigación resalta la importancia de integrar datos de incidentes viales con información meteorológica para comprender mejor los patrones y factores que contribuyen a estos eventos.

Identificamos desafíos, como el desequilibrio en las clases de variables y su impacto en el análisis predictivo, lo que nos llevó a emplear técnicas de submuestreo aleatorio para abordar este problema. Además, el uso del aprendizaje no supervisado, específicamente el algoritmo de K-medias, nos permitió encontrar patrones en los datos y establecer grupos que pueden ser útiles en futuros análisis y modelos predictivos.

Al evaluar varios algoritmos de aprendizaje supervisado, encontramos que el modelo de Impulso Gradiente demostró ser el más efectivo en la predicción de accidentes urbanos bajo condiciones climáticas variables. La optimización de hiperparámetros mejoró aún más su desempeño, logrando

una reducción significativa en el error porcentual absoluto medio (MAPE) sin comprometer significativamente el coeficiente de determinación (R^2). Además, esta optimización se logró manteniendo tiempos de entrenamiento eficientes.

Nuestro trabajo destaca la relevancia y la promesa del aprendizaje automático en la mitigación de accidentes urbanos al ofrecer perspectivas valiosas para el desarrollo de estrategias preventivas y de gestión del tráfico. Las futuras investigaciones podrían profundizar en la relación entre condiciones climáticas específicas y tipos de accidentes, así como explorar el impacto de variables socioeconómicas en estos eventos. Estas direcciones podrían enriquecer aún más nuestra comprensión y capacidad predictiva en este campo crucial de la seguridad vial.

Referencias

[1] S. de Seguridad y Protección a la Ciudadanía de Monterrey, "Incidentes viales en monterrey," 2023. [Online]. Available: https://datos.monterrey.gob.mx/dataset/incidentes-viales
[2] Equipo de Open Meteo, "Open Meteo: Servicio de Pronóstico del Tiempo," https://open-meteo.com/, accedido el 21 de noviembre de 2023.
[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and P. K. Dhariwal, "Imbalanced learning: Techniques for learning from imbalanced data sets," 2015.
[4] S. learn developers, "Getting started with scikit-learn," 2023. [Online]. Available: https://scikit-learn.org/stable/getting_started.html
[5] G. James, D. Witten, T. Hastie, and R. Tibshirani, "An introduction to random forests," 2013.
[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and . Duchesnay, "Scikit-learn: Machine learning in python," 2023.
[7] G. James, D. Witten, T. Hastie, and R. Tibshirani, "An introduction to linear regression," 2013.
[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and . Duchesnay, "Scikit-learn: Machine learning in python," 2023.
[9] T. Hastie, R. Tibshirani, and J. H. Friedman, "The elements of statistical learning," 2009.
[10] A. Géron, "Hands-on machine learning with scikit-learn, keras, and tensorflow," 2019.