"Máquinas que Aprenden: Explorando las Maravillas del Aprendizaje Automático"

El objetivo de este proyecto es familiarizarse con la librería scikit-learn y las herramientas que ofrece para el pre-procesamiento de datos, la implementación de modelos y la evaluación de métricas. También se busca comprender el uso de TensorFlow para el entrenamiento de redes neuronales. El dataset utilizado, llamado weatherAUS.csv, contiene información climática de Australia de los últimos diez años. Se centra en predecir si lloverá o no, así como la cantidad de lluvia para el día siguiente, a partir de diversas características. La predicción se limita a las ciudades de Sydney, Sydney Airport, Canberra, Melbourne y Melbourne Airport en la costa sureste de Australia, considerándolas como una única ubicación. El paso inicial implica descartar el resto de los datos para concentrarse en esta región específica.

Autores: Lo Menzo Alejo y Sagrera Ramiro

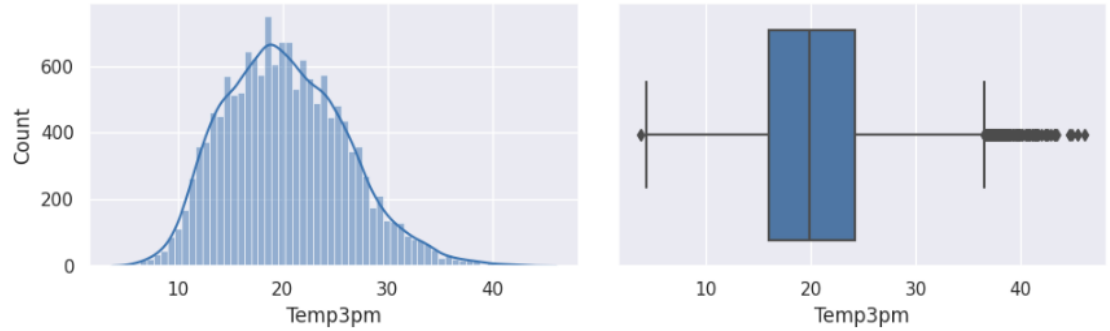
FCEIA
FACULTAD DE
CIENCIAS EXACTAS,
INGENIERIA Y AGRIMENSURA

El informe irá detallando cómo se desarrolló el trabajo, con sus problemáticas y las decisiones que tomamos para solucionarlas.

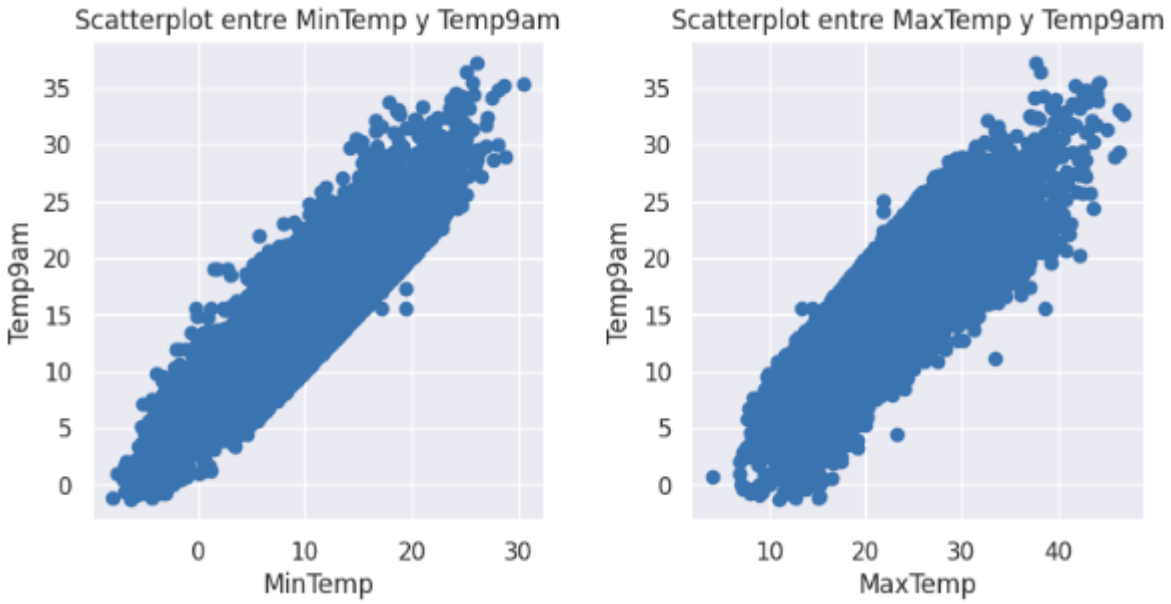
La Transformación y Limpieza del mismo fue un punto clave a la hora de tomar decisiones. Como primer medida filtramos el dataset por sobre las ciudades con las que íbamos a trabajar y eliminamos los registros de nuestras columnas a predecir ya que el total de valores faltantes no iban a influir en la predicción de los mismos.
La variable “Date” fue transformada y dividida por cada mes. Por último eliminamos las columnas innecesarias.

En el Análisis Exploratorio nos centramos en identificar qué variables tienen mayor correlación con las variables a predecir. Podemos notar que las que poseen el mismo nombre pero distinto horario están altamente relacionadas.

También graficamos histogramas y diagramas de cajas para visualizar las distribuciones de las mismas.



Por último, graficamos los ScatterPlots de las variables con mayor y menor correlación.



La División de Datos fue implementada de una manera más legible y prolija ya que asignamos nuestra variable para la regresión y otra para la clasificación. Por último se visualizan los datos de entrenamiento y testeo.

Otras de las decisiones que tomamos fue ¿qué hacíamos con los valores faltantes?
Los valores numéricos faltantes fueron reemplazados por su media, mientras que el resto con la moda.

La Codificación de las variables categóricas la realizamos con la función get_dummies de Pandas para aplicar One-Hot Encoding.En la misma dividimos los datos de entrenamiento y testeo ya sea para regresión o clasificación. Por último mapeamos las palabras “Yes” y “No” por valores binarios 1 y 0 de 'RainToday' y 'RainTomorrow'.

Una de las preguntas que nos hicimos fue si el Dataset estaba balanceado? La respuesta fue claramente “No” ya que nuestra variable a predecir contenía resultados muy dispares. Por lo cual procedimos a balancearlos aplicando Random Under Sampler, técnica de remuestreo que se utiliza especialmente en problemas de clasificación con conjuntos de datos desbalanceados. Su función principal es equilibrar el número de muestras entre las clases minoritarias y mayoritarias reduciendo aleatoriamente el número de muestras de la clase mayoritaria.

Como último punto de la transformación del Dataset, realizamos la estandarización de las variables numéricas con el método Standar Scaler tanto para el conjunto de entrenamiento como el de prueba.

Para la selección de características de predicción utilizamos “PCA” ya que simplifica y reduce la dimensionalidad de los datos, preservando al mismo tiempo la mayor cantidad posible de información importante.
Creamos un gráfico para visualizar la varianza explicativa acumulada y la del codo. Por último obtuvimos los autovalores de los componentes para analizar el criterio de Kaiser y creamos el dataset con los componentes principales.

Para la Regresión Lineal se estableció la función mape (Mean Absolute Percentage Error) para calcular el error porcentual absoluto medio entre las predicciones y los valores reales. Esta función incorpora una pequeña constante para evitar divisiones por cero.
Posteriormente ajustamos el modelo con un conjunto de datos de entrenamiento (X_train_reg y y_train_reg) y se calcularon diversas métricas para evaluar la precisión del modelo:

- Coeficiente de Determinación (R2 Score)
- Error Cuadrático Medio (MSE)
- Raíz Cuadrada del Error Cuadrático Medio (RMSE)
- Error Absoluto Medio (MAE)
- Error Porcentual Absoluto Medio (MAPE)

Por último realizamos predicciones adicionales para el conjunto de prueba, y se presentaron junto con los valores reales.

También implementamos un modelo base de Regresión Lineal que usa la media para predecir y otro que utiliza el gradiente descendiente.

Los métodos de regularización que utilizamos fueron “Lasso”, “Ridge” y “Elasticnet” en los cuales a cada uno le calculamos sus métricas.

Para la Regresión Logística se creó y entrenó un modelo con el conjunto de datos de entrenamiento (X_train_clas y y_train_clas). Luego, se realizaron predicciones en el conjunto de prueba (X_test_clas) y se imprimieron tanto las predicciones del modelo como las probabilidades asociadas a cada clase.
Posteriormente, se evaluó el rendimiento del modelo mediante el cálculo de la precisión en el conjunto de prueba utilizando el método score. Además, se imprimieron los coeficientes de las características y el intercepto del mismo.

También implementamos un modelo base de Regresión Logística que usa la media para predecir.

Las métricas de clasificación que utilizamos fueron:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

Redes Neuronales con la variable RainTomorrow:
Inicialmente, se determinó el número de características del conjunto de datos. Luego, se creó un modelo secuencial que consta de una capa densa de 32 unidades con función de activación ReLU, seguida por una capa de salida con una única unidad y función de activación sigmoide. El modelo se compiló utilizando el optimizador 'adam' y la función de pérdida 'binary_crossentropy', con la métrica de precisión ('accuracy') para su evaluación.
A continuación, se llevó a cabo el entrenamiento del modelo utilizando el conjunto de entrenamiento (X_train_clas y y_train_clas) durante 10 épocas con un tamaño de lote de 128. Además, se proporcionó un conjunto de validación (X_test_clas y y_test_clas) para chequear el rendimiento durante el entrenamiento.
Finalmente, se evaluó el modelo en el conjunto de prueba, y se imprimieron la pérdida y la exactitud obtenidas. Estos resultados proporcionan una medida de la capacidad del modelo para generalizar datos no vistos.

Capas convolucionales con la variable RainTomorrow:
Se experimentó con un modelo de red neuronal utilizando capas densas. El modelo fue configurado con una capa de entrada de 77 dimensiones, seguida por una capa oculta de 32 unidades con activación ReLU y una capa de salida con activación sigmoide. Se utilizó el optimizador SGD con una tasa de aprendizaje de 0.01 y momentum de 0.9.
El modelo se compiló con la función de pérdida 'binary_crossentropy' y se entrenó durante 10 épocas con un tamaño de lote de 128. Se registraron las métricas de entrenamiento y validación. Finalmente, se evaluó el modelo en el conjunto de prueba y se imprimieron la pérdida y la precisión obtenidas.

Redes Neuronales con la variable RainfallTomorrow:
Implementamos una función para calcular el R2 ajustado.
Creamos un modelo secuencial con una capa densa de 32 unidades y activación ReLU como capa de entrada, seguida por una capa de salida con una única unidad. El modelo se

compiló utilizando el optimizador 'adam' y la función de pérdida 'mean_squared_error'. Además, se definió una métrica personalizada, en este caso, el R2 ajustado
Luego, el modelo se entrenó con datos de entrenamiento (X_train_reg y y_train_reg) durante 10 épocas, utilizando un tamaño de lote de 128. Se proporcionó un conjunto de validación (X_test_reg y y_test_reg) para evaluar el rendimiento del modelo durante el entrenamiento.

Capas convolucionales con la variable RainfallTomorrow:
También fue creado un modelo secuencial con una capa densa de 32 unidades y activación ReLU como capa de entrada, seguida por una capa de salida con una única unidad. Se utilizó el optimizador SGD (Gradiente Descendente Estocástico) con una tasa de aprendizaje de 0.01 y momentum de 0.9. El modelo se compiló con la función de pérdida 'mean_squared_error' y se incorporó la métrica personalizada R2 ajustado.
Posteriormente, el modelo se entrenó con datos de entrenamiento (X_train_reg y y_train_reg) durante 20 épocas, utilizando un tamaño de lote de 128.

En la Optimización de Hiperparametros se realiza una validación cruzada para evaluar el rendimiento del modelo de regresión Gradient Descent. Se imprime el R2 Score promedio y su desviación estándar en cada fold. Luego, se repite la validación cruzada utilizando KFold y se imprimen los resultados.
Posteriormente, se dividen los datos de entrenamiento en un conjunto temporal y un conjunto de validación. Se define un conjunto de hiper parámetros para ajustar utilizando GridSearchCV, y se realiza la búsqueda de cuadrícula para encontrar los mejores hiper parámetros.
Obtenemos los mejores hiper parámetros y el mejor modelo entrenado a partir de la búsqueda de cuadrícula. Por último hacemos predicciones en un conjunto de validación y calculamos métricas de evaluación como R2 Score, MSE, RMSE, MAE y MAPE.

Para el modelo de clasificación logística realizamos la validación cruzada utilizando cross_val_score e imprimimos puntajes de cada fold, junto con el puntaje promedio y la desviación estándar.
Luego, repetimos la validación cruzada utilizando StratifiedKFold, una variante que asegura la preservación de la proporción de clases en cada fold.
Después, dividimos los datos de entrenamiento en un conjunto temporal y un conjunto de validación. Definimos un espacio de búsqueda para hiper parámetros utilizando la regularización (l1 o l2) y la inversa de la fuerza de regularización (C).
Empleamos GridSearchCV para encontrar los mejores hiper parámetros mediante una búsqueda de cuadrícula y posteriormente imprimimos los mejores.
Finalmente, utilizamos el mejor modelo para hacer predicciones en el conjunto de validación, y calculamos las diversas métricas de evaluación como precisión, recall, F1 Score y ROC-AUC.

Optimización de Hiper parámetros de Redes Neuronales para Regresión:
Definimos una función de objetivo para la regresión en Optuna (objective_reg), exploramos diferentes valores para la cantidad de unidades, funciones de activación oculta y de salida. Utilizamos un modelo de red neuronal para regresión con los valores sugeridos por Optuna.
En el entrenamiento del Modelo compilamos y entrenamos el modelo con los mejores hiper parámetros encontrados. Utilizamos el conjunto de validación para ajustar el modelo.

Finalmente evaluamos el rendimiento del modelo optimizado en el conjunto de prueba e imprimimos las métricas como R2 Score, MSE, RMSE, MAE y MAPE.

Optimización de Hiper parámetros de Redes Neuronales para Clasificación:
Definimos una función de objetivo para la clasificación en Optuna (objective_clas) y exploramos los diferentes valores para la cantidad de unidades, funciones de activación oculta y de salida. Posteriormente utilizamos un modelo de red neuronal para clasificación con los valores sugeridos por Optuna. En el entrenamiento del Modelo se compilo y entrenó con los mejores hiper parámetros encontrados. También utilizamos el conjunto de validación para ajustarlo. Por último evaluamos el rendimiento del modelo optimizado en el conjunto de prueba e imprimimos las métricas como Accuracy, Precision, Recall y F1 Score.

Estos pasos nos aseguraron que los modelos de regresión y clasificación estén ajustados de manera óptima utilizando la optimización de hiper parámetros con Optuna.

Explicación del Modelo de Regresión y Clasificacion con SHAP:
Creamos un objeto explainer que se configura con los datos de entrada X_shap_clas y calculamos los valores SHAP (shap_values) para el conjunto de datos X_shap_clas. Estos valores representan la contribución de cada característica a las diferencias entre las predicciones del modelo y los valores base.

Visualización de Resumen (Bar Chart):
- Se genera un gráfico de resumen en forma de barra que muestra la importancia media de cada característica en todas las predicciones del modelo.

Visualización de Bar Chart Individual:
- Se crea un gráfico de barras que destaca la importancia media de cada característica. Este gráfico proporciona una perspectiva adicional sobre la contribución de cada característica a las predicciones del modelo.

Visualización de Beeswarm Plot:
- Se produce un gráfico de abanico ("beeswarm") que ilustra cómo cada punto de datos contribuye a las predicciones para todas las características.

Visualización de Force Plot:
- Se genera un gráfico de fuerza que detalla la contribución de cada característica para una predicción específica, en este caso, la primera predicción.

Waterfall Plot:
- Se produce un gráfico de cascada que muestra cómo se pasa de los valores base de SHAP a la predicción del modelo para una instancia específica. Este gráfico visualiza la contribución acumulativa de cada característica.

Estos pasos nos permitieron una comprensión detallada de cómo cada característica impacta las predicciones del modelo de regresión y clasificación. La visualización de SHAP facilita la interpretación y explicación de las decisiones del modelo.

Explicación de las predicciones de dos modelos (regresión lineal y clasificación) mediante el método de Kernel Explainer:

Definimos la función f(X) que toma un conjunto de datos X y devuelve las predicciones del modelo. La función utiliza el método predict del modelo y se aplana para obtener un arreglo unidimensional.

Kernel Explainer para Regresión Lineal:

- Creamos un objeto explainer utilizando el Kernel Explainer de SHAP para el modelo de regresión lineal. Utilizamos el conjunto de datos X_shap_reg y se calculan los valores SHAP (ex_shap_values) para una instancia específica (fila 299). Por último implementamos el método de fuerza para visualizar la explicación detallada de cómo las características contribuyen a la predicción del modelo para esa instancia.

Kernel Explainer para Clasificación:

- Repetimos el proceso para el modelo de clasificación y creamos otro objeto explainer utilizando el Kernel Explainer de SHAP para el modelo de clasificación. Utilizamos datos diferentes (X_shap_clas) y calculamos los valores SHAP (shap_values) para la misma instancia (fila 299). También se visualiza la explicación utilizando el método de fuerza.

El método de Kernel Explainer de SHAP lo utilizamos para proporcionar explicaciones detalladas sobre cómo las características contribuyen a las predicciones de dos modelos (regresión lineal y clasificación) para una instancia específica.

Como último punto proporcionamos una aplicación web mediante Streamlit que utiliza un modelo de clasificación pre entrenado para predecir si lloverá o no en función de las variables meteorológicas ingresadas por el usuario. Los valores de las variables se recopilan a través de controles interactivos en la barra lateral, y la predicción del modelo se muestra en la interfaz principal. La aplicación destaca por su capacidad de personalización, permitiendo al usuario especificar valores para variables numéricas y seleccionar opciones para variables categóricas, brindando así una experiencia interactiva y comprensible.

Después de resolver todas las consignas queríamos hablar sobre las dificultades que tuvimos a la hora de trabajar en el TP:
Uno de los mayores problemas fue el proceso de ordenar todos los pasos necesarios entre la carga del dataset y las implementaciones de los modelos. En un principio habíamos realizado el split de los datos en última instancia, lo cual llevaba al data leakage. Así que tuvimos que reordenar los procesos de imputación de datos faltantes, codificación de variables, etc.

Otra cosa que nos llevó mucho tiempo fue la prueba que hicimos con PCA, ya que más allá de encontrar la cantidad óptima de componentes, no vimos resultados concretos de mejora en los modelos. Lo que nos llevó a probar con distintos parámetros.

Como conclusión queremos remarcar que la modalidad del trabajo se hizo bastante llevadera, también agregar que el trabajo fue muy fructífero ya que sentimos que nos va a ayudar a la hora de implementarlo en nuestras profesiones en un futuro.