Implementación de una técnica de aprendizaje máquina sin el uso de un framework

Oskar Adolfo Villa López

Abstract

Se presenta la implementación de modelos de aprendizaje máquina con algoritmos de regresión logística, redes neuronales y random forest para la predicción de las cancelaciones de reservaciones en un hotel. Se busca ajustar cada modelo para que sea lo más efectivo posible, y se comparan los resultados de los tres modelos. Finalmente se llega a un modelo con 20 % de error. Las implementaciones de red neuronal y random forest se realizaron con el uso de frameworks, a diferencia del modelo de regresión logística.

Introducción

En este trabajo se busca implementar modelos de aprendizaje máquina para predecir un fenómeno real. El conjunto de datos seleccionado es Hotel Reservations, obtenido de la plataforma Kaggle [4]. Las filas contienen datos correspondientes a una reservación de hotel, e incluyen columnas como número de habitaciones, días de anticipación de la reservación o tipo de plan de comida. También se incluye una columna que indica si la reservación fue cancelada o no. El objetivo del modelo es predecir si la reservación será cancelada utilizando los datos de ésta. Se analiza la implementación de un modelo de regresión logística, una red neuronal y un random forest.

Obtención y tratamiento de datos

Extracción y descripción

El archivo de datos contiene 36,275 muestras, con 19 columnas. Cada fila de la tabla contiene los datos de una reservación. Las columnas contenidas son las siguientes:

- **Booking_ID**: Identificador único de cada reserva
- **no_of_adults**: Número de adultos
- **no_of_children**: Número de niños
- **no_of_weekend_nights**: Número de noches de fin de semana (sábado o domingo) que el huésped se quedó o reservó para quedarse en el hotel
- **no_of_week_nights**: Número de noches entre semana (lunes a viernes) que el huésped se quedó o reservó para quedarse en el hotel
- **type_of_meal_plan**: Tipo de plan de comidas reservado por el cliente:
- **required_car_parking_space**: ¿El cliente requiere un espacio de estacionamiento? (0 - No, 1- Sí)
- **room_type_reserved**: Tipo de habitación reservada por el cliente. Los valores están cifrados (codificados) por INN Hotels.
- **lead_time**: Número de días entre la fecha de la reserva y la fecha de llegada
- **arrival_year**: Año de la fecha de llegada
- **arrival_month**: Mes de la fecha de llegada
- **arrival_date**: Día del mes de la fecha de llegada
- **market_segment_type**: Designación del segmento de mercado.
- **repeated_guest**: ¿Es el cliente un huésped recurrente? (0 - No, 1- Sí)
- **no_of_previous_cancellations**: Número de reservas anteriores que fueron canceladas por el cliente antes de la reserva actual
- **no_of_previous_bookings_not_canceled**: Número de reservas anteriores que no fueron canceladas por el cliente antes de la reserva actual
- **avg_price_per_room**: Precio medio por día de la reserva; los precios de las habitaciones son dinámicos. (en euros)
- **no_of_special_requests**: Número total de solicitudes especiales realizadas por el cliente (por ejemplo, piso alto, vista desde la habitación, etc.)
- **booking_status**: Bandera que indica si la reservación fue cancelada.

Selección de features y label

De acuerdo con el objetivo del análisis, la columna a predecir (label) será **booking_status**, mientras que el resto de columnas se usarán como variables independientes (features). No todas las columnas dentro de los features contienen información relevante para el análisis, por lo que que las siguientes columnas fueron descartadas:

- **Booking_ID**: El identificador es aleatorio y no contiene información relevante.

- **arrival_month, arrival_year, arrival_date**: Las fechas no se consideraron para el análisis, ya que se encuentran fuera del objetivo de aprendizaje de éste.

Columnas no numéricas

Los datos contienen columnas no numéricas. Éstas fueron transformadas a columnas numéricas utilizando el método One Hot Encoding, el cual crea una columna binaria para cada una de las opciones dentro de una columna no numérica, y elimina la columna original. Se eligió este método porque funciona bien cuando la columna contiene más de dos opciones y éstas no cuentan con relaciones escalares. Para la columna de label se utilizó Label Encoding, el cual cambia los valores contenidos en la columna (en este caso True y False) por valores binarios. Se utilizó este método porque la columna contiene solo valores correspondientes a verdadero y falso, los cuales pueden ser representados con 0 y 1 dentro de la misma columna.

Selección de conjuntos de entrenamiento y pruebas

Antes de seleccionar los subconjuntos, se utilizó la función *sample* de Pandas para reordenar de manera aleatoria el conjunto completo de datos, con el objetivo de evitar sesgos en caso de que los datos se encuentren ordenados. Es importante tener en cuenta aleatoriedad provoca que el código arroje resultados distintos cada vez que se ejecuta. Después de ser reordenados, se seleccionaron dos subconjuntos de datos, seleccionando el 80 % para entrenamiento y el 20 % para pruebas. Para la selección se utilizó la función *train_test_split* de Sklearn. Esto se debe a que el modelo tenía resultados inconsistentes entre ejecuciones por un balance incorrecto entre categorías presentes en el set de entrenamiento. La función se asegura de que las muestras se encuentren distribuidas correctamente.

Identificación de outliers y valores nulos

Se utilizaron las funciones *info* y *describe* de Pandas para poder identificar columnas con datos faltantes y outliers en los datos. Después de inspeccionar los resultados, no se encontraron valores nulos. Se detectaron outliers, presentes en la Figura 17 en los anexos, en las siguientes columnas:

- **no_of_week_nights**
- **lead_time**
- **no_of_previous_cancellations**
- **no_of_previous_bookings_not_canceled**

Para eliminarlos, se utilizó el método IQR, que etiqueta como valores atípicos los valores que se encuentran fuera de $Q3 + 1,5 * IQR$ o $Q1 - 1,5 * IQR$, donde $Q1$ y $Q3$ son los límites del primer y tercer cuartil, respectivamente, e IQR es el rango intercuartílico, o el rango que existe entre $Q1$ y $Q3$ [3]. Los valores atípicos fueron reemplazados por su límite más cercano.

Escalamiento de datos

Después de analizar los datos, se detectó que, a pesar de que cada columna tiene un rango aceptable de datos, los rangos difieren mucho entre columnas. Es por ello que se utilizó un escalamiento, con el objetivo de simplificar el proceso de entrenamiento al acercar todos los valores de los features a un mismo rango. Si los valores tienen una variación muy alta, es posible que el modelo tenga dificultad para adaptar la magnitud de cada uno de los parámetros para contrarrestar la diferencia en magnitud de los features.

Regresión logística

Selección del modelo

Se seleccionó un modelo de regresión logistica, el cual es útil para predecir resultados binarios mutuamente excluyentes [1], como es el caso de el estado de una reservación de hotel.

El modelo requiere una funciones de hipótesis, activación, costo y optimización, las cuales se describen a continuación. Utilizando la función de optimización, se ajusta la función de hipótesis en cada iteración de aprendizaje, llamada época, hasta alcanzar un costo mínimo.

Hipótesis (h)

La hipótesis es la función que busca predecir el resultado con relación a las variables independientes del modelo [2]. En la regresión logísitica se utiliza una función lineal con múltiples variables independientes, de la forma:

$$\hat{\mathbf{y}}(x_1, ..., x_n) = \theta_0 + \theta_1 x_1 + ... + \theta_n x_n$$

Donde $\hat{y}$ representa el label o variable dependiente, θ son los parámetros y las x representan los features o variables dependientes.

Activación

La función de hipótesis devuelve un valor con magnitud arbitraria que es poco interpretable. Es por ello que se utiliza una función de activación, que en el caso de la regresión logística tiene el objetivo de normalizar el resultado para acotarlo a un rango [2]. Para el modelo se utiliza la función sigmoidal, la cual acota los resultados a un valor entre 0 y 1, lo cual es útil para interpretar el resultado binario esperado. La función sigmoidal está definida como:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Costo

El costo es la función necesaria para calcular la diferencia entre la predicción del modelo y el valor real. Para el caso de clasificación con la función sigmoidal se puede utilizar la función cross-entropy, la cual resulta de la siguiente manera:

$$loss = -y(log(h(x)) - (1 - y)(log(1 - h(x)))$$

Donde $h(x)$ es el resultado de la función de hipótesis y y es el valor real de label.

Gradiente descendiente

Finalmente, se requiere una función para entrenar el modelo y darle la capacidad de aprender cuando se alimenta con datos. La función seleccionada para este modelo es gradiente descendiente, que consiste en una función de optimización que busca minimizar el costo de las predicciones [2]. La función es la siguiente:

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\theta(x_i) - y)x_{ij}]$$

Donde α es learning rate y m es la cantidad de muestras. Esta función debe aplicarse para cada una de las tetas, guardando el valor resultante y actualizando todos los valores al finalizar el cálculo.

Hiperparámetros

El modelo contiene los hiperparámetros learning rate y número de épocas. Para encontrar los valores óptimos de éstos, se entrenó y probó el modelo con diferentes combinaciones hasta encontrar un mínimo local en el error, lo que corresponde con un valor mayor en accuracy. El mejor modelo resultante se entrenó con un learning rate de 0.1 en 10000 épocas. La Figura 1 muestra una comparación de accuracy y error entre las diferentes combinaciones. La combinación 2 corresponde a la descrita anteriormente.

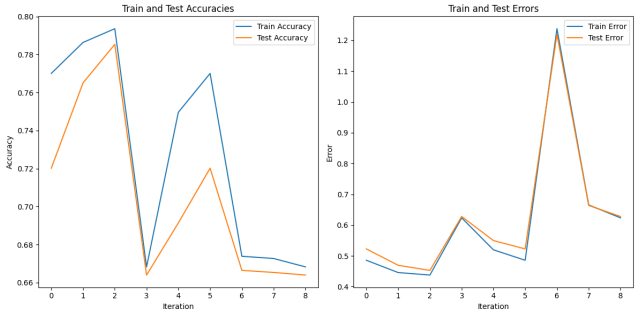

Figura 1: Accuracy y error con distintos hiperparámetros.

Resultados

Error en entrenamiento Como se puede observar en la Figura 2, el error del modelo se redujo con cada época, lo que indica que el modelo fue ajustando los parámetros para predecir con más certeza el label. La magnitud del error no es interpretable, pero el comportamiento es el esperado.

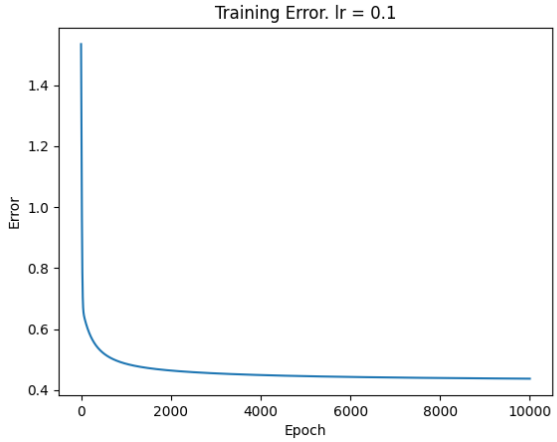

Figura 2: Error a lo largo del entrenamiento.

Matriz de confusión La Figura 3 muestra la matriz de confusión para el conjunto de entrenamiento, mientras que la Figura 4 muestra la matriz para el conjunto de prueba. Como se puede observar, el modelo tiene respuestas correctas para la mayoría de las muestras del conjunto de entrenamiento. En el caso de las pruebas, el modelo predice correctamente los resultados 1, pero tiene dificultad con los valores 0. Con mucha frecuencia, el error produce falsos positivos.

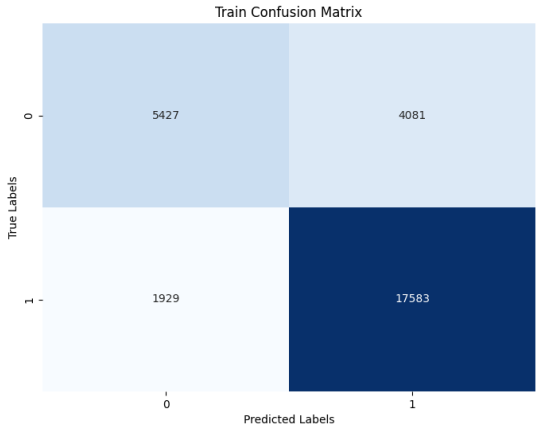

Figura 3: Matriz de confusión para conjunto de entrenamiento.

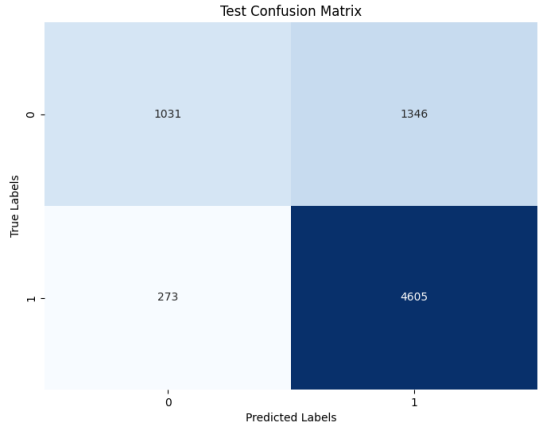

Figura 4: Matriz de confusión para conjunto de pruebas

Accuracy y F1 Las métricas de acierto utilizadas son accuracy y F1. Accuracy es el procentaje de acierto, el cual se calcula dividiendo el número de predicciones correctas entre las predicciones totales. La métrica F1 mide la exactitud del modelo cuando hay clases desequilibradas, y toma en cuenta la precisión y la recuperación del modelo. Los valores obtenidos son los siguientes:

- **Accuracy entrenamiento:** 0.792901
- **Accuracy pruebas:** 0.776843
- **F1 entrenamiento:** 0.854041
- **F1 pruebas:** 0.850494

Ambas métricas tienen magnitudes similares para entrenamiento y pruebas. Los valores para entrenamiento son ligeramente más altos, lo cual es esperado ya que el modelo conoce con anterioridad los datos de entrenamiento.

Evaluación del modelo

Varianza Debido a que las predicciones de los conjuntos de entrenamiento y pruebas son similares en cuanto a accuracy y F1, se puede deducir que la varianza es baja, ya que el modelo realiza predicciones estables. La varianza baja es correcta y esperada.

Sesgo El sesgo del modelo es medio, ya que el modelo es incorrecto en aproxiamadamente 20 % de las ocasiones. Un sesgo maś bajo otorgaría mejores resultados en accuracy y F1, lo que significaría que el modelo es más adecuado. Este sesgo medio puede deberse a que el modelo no es lo suficientemente complejo para seguir la tendencia de los valores reales.

Ajuste El ajuste del modelo está determinado por su capacidad de predecir resultados. En este caso, el modelo puede considerarse como *underfitted*, ya que tanto la métrica accuracy como F1 pueden mejorarse para llegar a un modelo más robusto, y los valores fueron similares en el entrenamiento, lo que descarta una clasificación de *overfitted*.

Red Neuronal

Selección del modelo

Una red neuronal es un alogoritmo computacional que busca replicar el funcionamiento biológico de las neuronal. Utiliza capas de ”neuronasïnterconectadas que se entrenan ajustando los valores de salida de cada neurona con respecto a los valores de entrada hasta conseguir resultados consistentes [5]. Las red neuronal implementada se entrenó usando la función de pérdida Cross Entropy mediante el método de Back Propagation, que consiste en ajustar los pesos desde la capa de salida hasta la capa de entrada con respecto al error obtenido después de correr una predicción.

Hiperparámetros

Los diferentes modelos de redes neuronales fueron entrenados en 200 épocas. Se hicieron pruebas con un mayor número de épocas y se descubrió que no existe una diferencia significativa en accuracy, como se puede ver en la Figura 5. Además, se utilizó este número porque tiene un tiempo de entrenamiento menor.

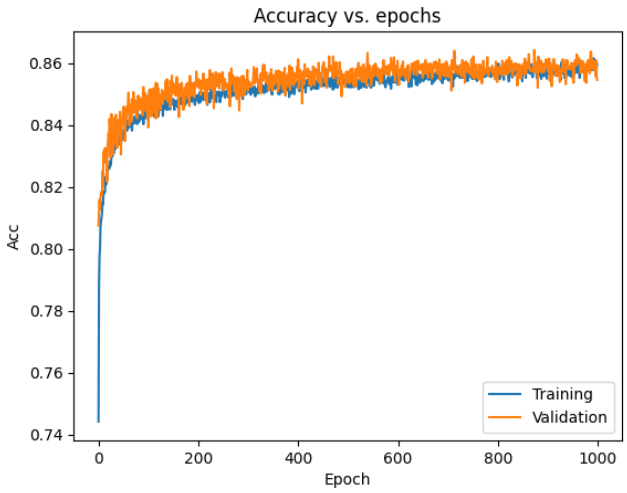

Figura 5: Accuracy en entrenamiento para 1000 épocas.

Iteraciones

Se realizaron distintas iteraciones con diferentes capas para buscar el modelo más adecuado. Para cada uno se midió accuracy y error a lo largo del entrenamiento, mediante un conjunto de validación de 15 % del tamaño del conjunto de entrenamiento. También se reporta la métrica F1 y accuracy para el conjunto de pruebas para cada uno de los modelos.

Modelo 1 Para el modelo 1 se utilizó una red neuronal secuencial con la siguiente configuración:

- Capa densa con activación ReLu. 64 nodos
- Capa de batch normalization.
- Capa densa con activación ReLu. 32 nodos.
- Capa de salida con activación sigmoidal. 1 nodo.

En la Figura 6 se observa la comparación de accuracy entre validación y entrenamiento. Se observa un comportamiento con mejoras a lo largo de las épocas.

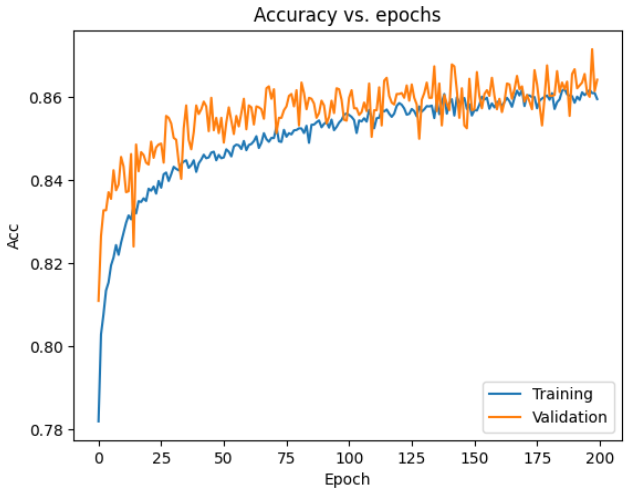

Figura 6: Accuracy de validación contra entrenamiento.

En la Figura 7 se observa la comparación de error entre validación y entrenamiento. Se observa un ligero overfitting al final del entrenamiento.

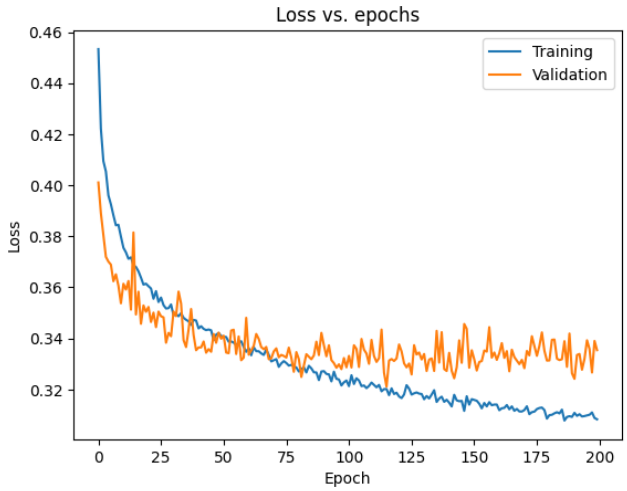

Figura 7: Error de validación contra entrenamiento.

Los resultados del modelo con el conjunto de pruebas son los siguientes:

- **Accuracy**: 0.7793
- **F1**: 0.8364

El resultado de accuracy final es bajo comparado con el obtenido con el conjunto de validación, lo que cataloga el modelo como *overfitted*.

Modelo 2 Para el modelo 2 se utilizó una red neuronal secuencial con la siguiente configuración:

- Capa densa con activación ReLu. 64 nodos
- Capa de batch normalization.
- Capa densa con activación ReLu. 32 nodos.
- Capa densa con activación ReLu. 32 nodos.
- Capa densa con activación ReLu. 32 nodos.
- Capa de salida con activación sigmoidal. 1 nodo.

En la Figura 8 se observa la comparación de accuracy entre validación y entrenamiento. Se observa un comportamiento con mejoras a lo largo de las épocas.

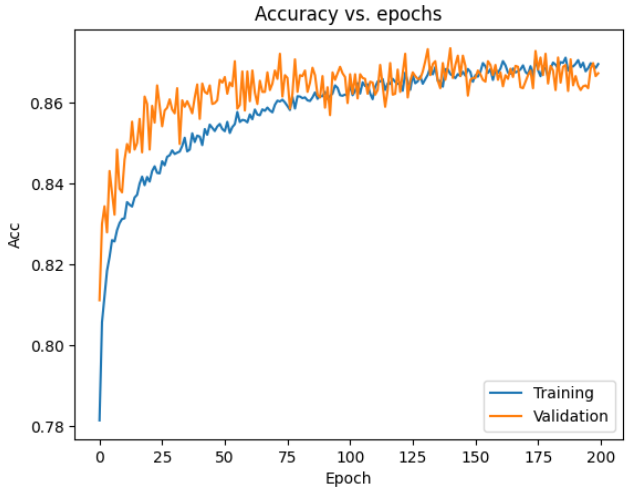

Figura 8: Accuracy de validación contra entrenamiento.

En la Figura 9 se observa la comparación de error entre validación y entrenamiento. Se observa un overfitting mayor que en el modelo 1.

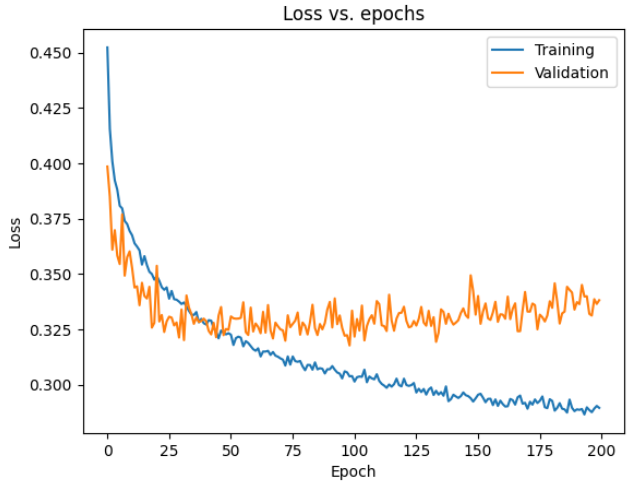

Figura 9: Error de validación contra entrenamiento.

Los resultados del modelo con el conjunto de pruebas son los siguientes:

- **Accuracy**: 0.7818
- **F1**: 0.8446

Al igual que el modelo 1, el modelo 2 muestra un overfitting cuando se comparan los conjuntos de pruebas y de validación. Sin embargo, el modelo 2 reporta resultados ligeramente mayores.

Modelo 3 Debido a que el modelo 2 presentó mejores resultados, se decidió agregar métodos de regularización a éste para buscar un mejor ajuste. El método a utilizar es Drop Out, el cual consiste en desactivar nodos aleatorios en cada época para introducir aleatoriedad en el entranamiento, evitando que los nodos se sobre ajusten al conjunto de entrenamiento. En el código, esto se logra mediante una capa matricial extra que actúa como una máscara, ya que al multiplicarse por los valores de la capa anterior anula algunos resultados, desactivando así el nodo correspondiente.

Para el modelo 3 se utilizó la misma configuración que el modelo 2, añadiendo la regularización con Drop Out:

- Capa densa con activación ReLu. 64 nodos
- Capa de batch normalization.
- Capa densa con activación ReLu. 32 nodos.
- Capa de Drop Out.
- Capa densa con activación ReLu. 32 nodos.
- Capa densa con activación ReLu. 32 nodos.
- Capa de Drop Out.
- Capa de salida con activación sigmoidal. 1 nodo.

En la Figura 10 se observa la comparación de accuracy entre validación y entrenamiento. Se observa un comportamiento con mejoras a lo largo de las épocas.

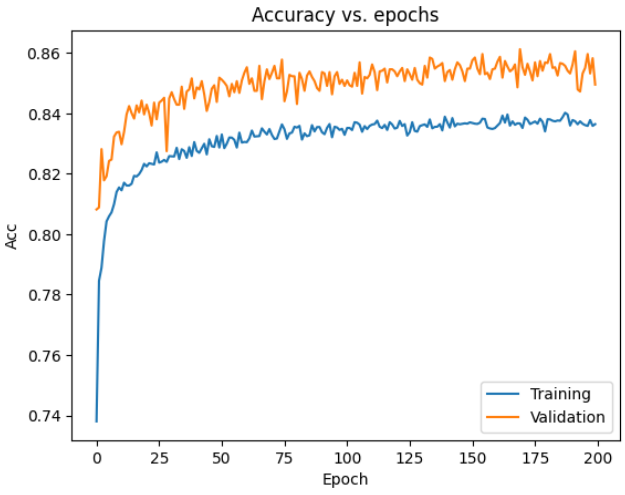

Figura 10: Accuracy de validación contra entrenamiento.

En la Figura 11 se observa la comparación de error entre validación y entrenamiento. La regularización logra disminuir el overfitting, igualando los errores.

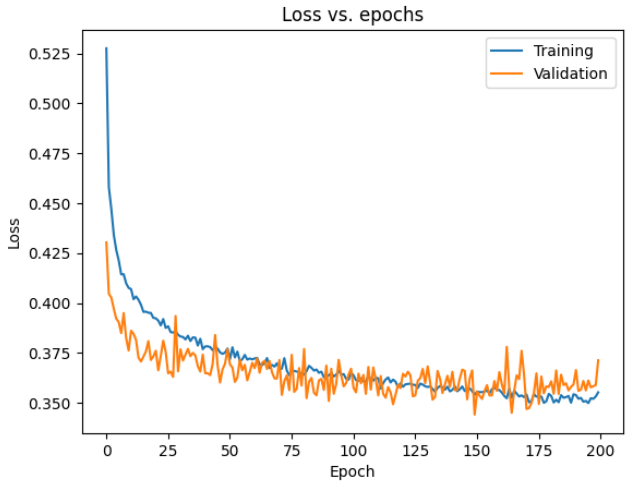

Figura 11: Error de validación contra entrenamiento.

Los resultados del modelo con el conjunto de pruebas son los siguientes:

- **Accuracy**: 0.8011
- **F1**: 0.8515

El modelo 3 tiene mejores resultados que los modelos 1 y 2, además de no presentar overfitting.

Resultados

El modelo 3 es el que presenta un mejor ajuste, por lo que fue el modelo seccionado. En la Figura 12 se observa la matriz de confusión para pruebas, mientras que en la Figura 13 se observa la matriz para un conjunto de validación. En ambos casos los resultados son buenos, haciendo predicciones correctas para ambas categorías y con errores bajos.

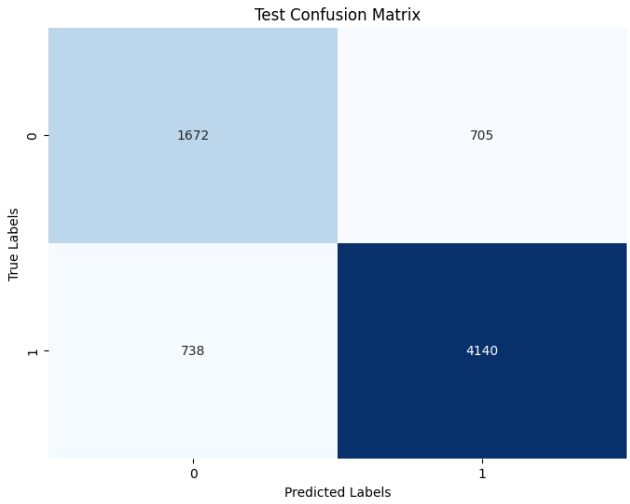

Figura 12: Matriz de confusión para pruebas de red neuronal.

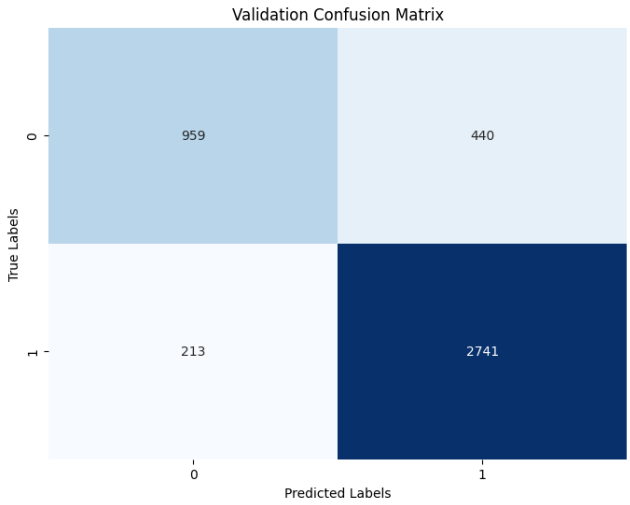

Figura 13: Matriz de confusión para validación de red neuronal.

Evaluación del modelo

El modelo 3 se considera como *fitted*, ya que logra efectivamente predecir las categorías requeridas, y muestra una efectividad equivalente en entrenamiento y pruebas.

Random Forest

Selección del modelo

Un modelo de Random Forest es un algoritmo que consiste en la creación de múltiples árboles de decisión, lo que permite seleccionar un resultado basado en los resultados de todos los árboles. Para el modelo, se utilizó el algoritmo de la librería Scikit-learn.

Hiperparámetros

Para la selección de la profundidad máxima de los árboles se utilizó un ciclo para probar distintos valores y comparar los resultados. Se utilizaron valores pequeños ya que el algoritmo de Random Forest funciona mejor con profundidades pequeñas, lo que evita el overfitting. Los resultados de las pruebas de muestran en la Figura 14, donde se puede observar que la profundidad óptima resultó con un valor de 10.

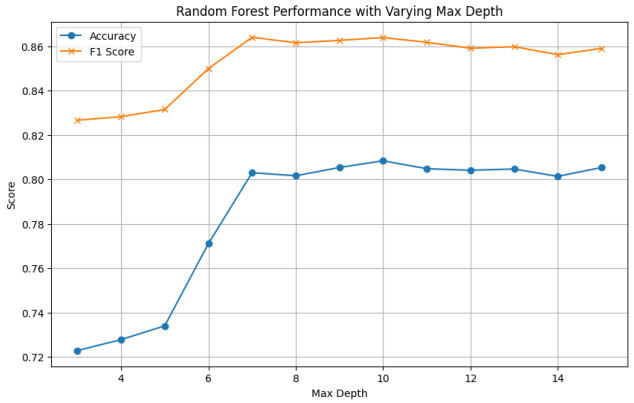

Figura 14: Comparación de modelos con distintos valores de max depth.

Resultados

Utilizando una profundidad máxima de 10 niveles, se presentan la matriz de confusión para el conjunto de pruebas en la Figura 15 y para el conjunto de entrenamiento en la Figura 16. Ambas matrices muestran predicciones adecuadas.

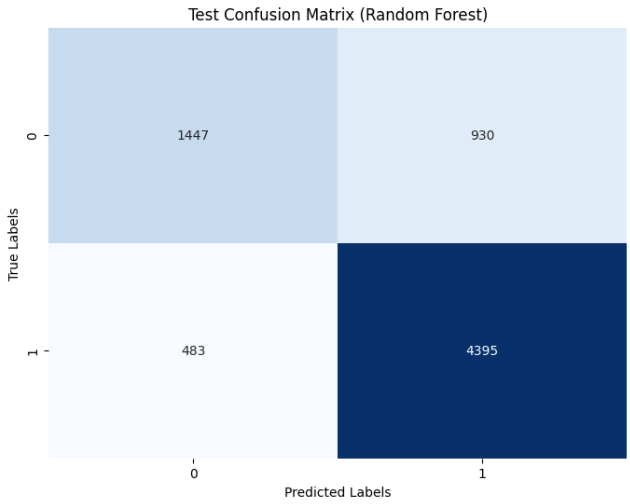

Figura 15: Matriz de confusión para pruebas de random forest.

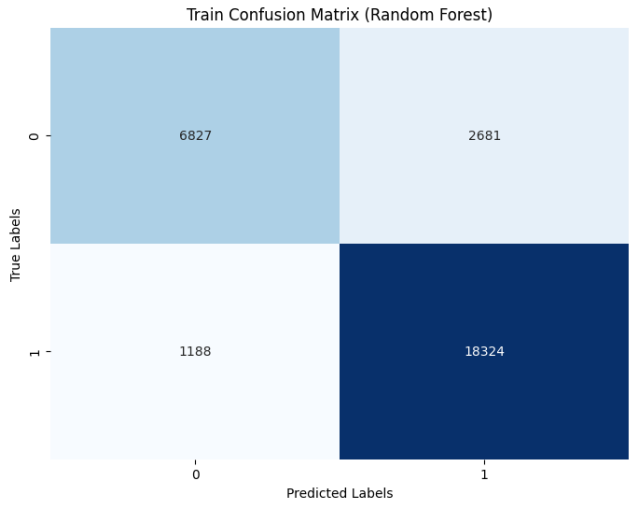

Figura 16: Matriz de confusión para entrenamiento de random forest.

Las métricas del modelo con el conjunto de pruebas son las siguientes:

- **Accuracy**: 0.8052
- **F1**: 0.8615

Evaluación del modelo

El modelo realiza predicciones con un accuracy y un F1 altos, además de obtener resultados similares para las matrices de confusión entre entrenamiento y pruebas, por lo que se considera como *fitted*.

Resultados generales

En la Tabla 1 se muestra la comparación de las métricas de el modelo más óptimo de cada algoritmo utilizado. El modelo con los mejores resultados es el Random forest. Además, el modelo Random forest es preferido sobre la red neuronal porque es menos costoso computacionalmente.

Tabla 1: Accuracy y F1-Score de tres modelos diferentes

Modelo	Accuracy	F1
Regresión logística	0.7768	0.8504
Red neuronal	0.8011	0.8515
Random forest	0.8052	0.8615

Es muy importante mencionar que, a pesar de tener un accuracy y F1 superior, el modelo de random forest tiene una desventaje. Si se observan las Figuras 12 y 15, se puede notar que, a pesar de que el modelo de random forest realiza más predicciones correctas en general, comete más errores al identificar los 0s en comparación con la red neuronal.

Conclusiones

Los tres algoritmos logran realizar predicciones con un accuaracy y F1 aceptables. El modelo de random forest es ligeramente superior a los demás en ambas métricas. Al final de las implementación se obtuvo un modelo *fitted* con un accuracy de alrededor de 80 %, lo cual es suficientemente bueno para predecir cancelaciones de hoteles, debido a que no es una decisión crítica.

Referencias

[1] Michael P. LaValley. “Statistical Primer for Cardiovascular Research”. En: *Circulation* 117.18 (2008). DOI: https://doi.org/10.1161/CIRCULATIONAHA.106.682658.

[2] Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 2015.

[3] Amerah Alabrah. “An Improved CCF Detector to Handle the Problem of Class Imbalance with Outlier Normalization Using IQR Method”. En: *MDPI* 23.9 (2023). DOI: https://doi.org/10.3390/s23094406.

[4] Ahsan Raza. *Hotel Reservations Dataset*. URL: https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset (visitado 15-08-2024).

[5] John V Urbas. *Neural Networks*. URL: https://research.ebsco.com/linkprocessor/plink?id=eb183947-db43-3008-be63-9d3daec6536e (visitado 15-08-2024).

Anexos

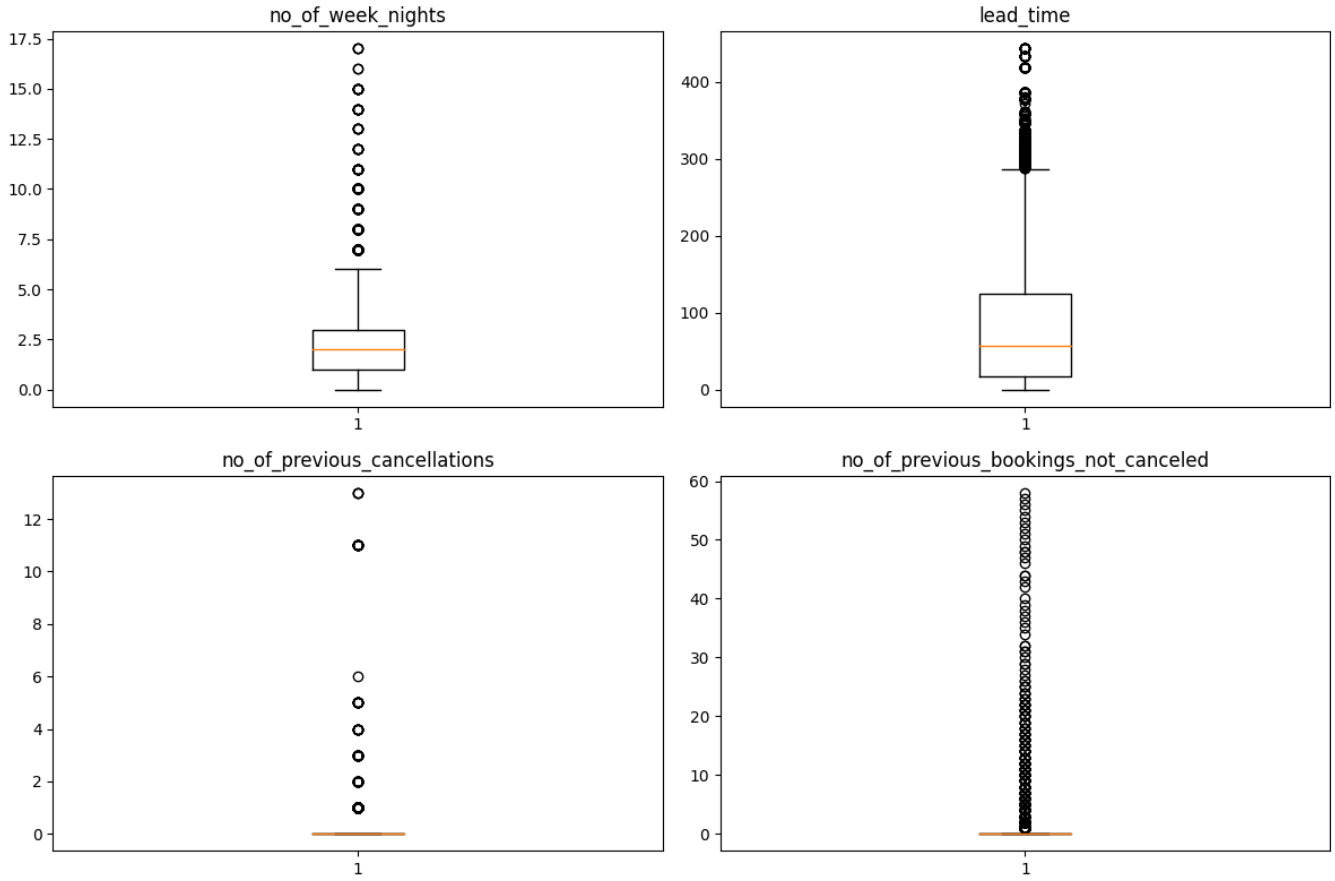

Figura 17: Distribución de datos para identificación de outliers.