## Introducción

En el transcurso de esta investigación, nos sumergimos en el análisis detallado de conjuntos de datos que describen las reservas de dos hoteles diferentes, uno de tipo resort (H1) y otro situado en la ciudad (H2). El foco principal de nuestro trabajo es abordar la compleja tarea de prever la demanda futura en la gestión hotelera, una habilidad vital para optimizar la ocupación de habitaciones y asignar recursos eficientemente.

La primera etapa de nuestro proyecto involucró la exploración de conjuntos de datos que abarcan un periodo considerable, desde julio de 2015 hasta agosto de 2017. El objetivo fundamental fue ajustar estrategias operativas en respuesta a las tendencias de reservas y cancelaciones. Establecimos métricas clave, desde la precisión hasta el impacto en los ingresos netos, para evaluar el rendimiento de nuestros modelos predictivos.

En una fase posterior, procedimos con tareas esenciales como la importación de bibliotecas, limpieza de datos y la implementación de modelos predictivos, en particular, un modelo de regresión logística. Esta fase también abordó la visualización de datos y el análisis de patrones, destacando la procedencia de los huéspedes, la variación de precios y otros factores relevantes.

A pesar de lograr resultados prometedores, particularmente en la predicción de reservas confirmadas, identificamos áreas de mejora, especialmente en el manejo de cancelaciones. En esta etapa final, nuestro enfoque se centra en la optimización continua del modelo, la exploración de técnicas avanzadas de aprendizaje automático y un análisis más profundo de los factores influyentes en la demanda hotelera.

Con este proyecto, aspiramos no solo a ofrecer predicciones precisas sino también a contribuir de manera significativa a la eficiencia operativa y la rentabilidad de la industria hotelera. Este resumen sirve como antesala a la culminación de nuestro trabajo, donde convergen análisis avanzados, ajustes de modelo y la implementación de soluciones prácticas para abordar desafíos reales en la gestión hotelera.

## Exploración Detallada del Dataset de Demanda Hotelera

Conjuntos de datos con información sobre la demanda hotelera. Uno de los hoteles (H1) es un hotel de resort y el otro es un hotel en la ciudad (H2). Ambos conjuntos de datos comparten la misma estructura, con 31 variables (columnas) que describen las 40,060 observaciones de H1 y las 79,330 observaciones de H2. Cada observación representa una reserva de hotel. Ambos conjuntos de datos abarcan reservas programadas para llegar entre el 1 de julio de 2015 y el 31 de agosto de 2017, incluyendo reservas que se materializaron y reservas que fueron canceladas. El dataset brinda información acerca de los planes tomados por las reservaciones (comida, fechas, número de integrantes, tipo de reserva, país de

origen, estado de la reserva, número de noches, canal de distribución de la reserva, etc.) Se puede encontrar en el siguiente link: https://www.kaggle.com/datasets/jessemostipak/hotel-booking demand?select=hotel_bookings.cs

La exploración de datos comenzó con una minuciosa limpieza, abordando valores nulos en variables clave y eliminando reservas sin huéspedes registrados. La distribución geográfica de los huéspedes se visualizó a través de un gráfico de pastel, proporcionando una representación gráfica de su origen.

El análisis de precios fue un componente esencial, calculando el costo por habitación por persona (adr_pp) para ambos tipos de hoteles y evaluando las tendencias mensuales de precios. Este enfoque arrojó luz sobre las dinámicas de precios a lo largo del tiempo.

Adicionalmente, se implementaron modelos de aprendizaje automático, incluyendo regresión logística, K-Nearest Neighbors (KNN) y Random Forest, con ajustes cuidados para optimizar su rendimiento. Estos modelos se evaluaron mediante métricas clave para proporcionar percepciones sobre su eficacia en la predicción de cancelaciones de reservas.

En resumen, esta exploración detallada establece las bases para comprender los patrones de demanda hotelera, aprovechando la combinación de métodos exploratorios y modelos predictivos. El objetivo último es anticipar tendencias y mejorar la gestión hotelera, contribuyendo a la eficiencia y rentabilidad de la industria.

## Procesado de datos.

En la primera fase del proyecto, se realiza una exploración descriptiva de un extenso conjunto de datos relacionado con reservas hoteleras. Además, implementamos modelos de aprendizaje automático con el objetivo de prever cancelaciones de reservas.

1. Limpieza y Preprocesamiento de Datos:

```
# Remplazamos los valores vacios:
# agent: Si no hay agencia, se agendo la reserva sin ella
# company: Si no hay compañia, la reserva se hizo de forma privada
nan_replacements = {"children:": 0.0,"country": "Unknown", "agent": 0, "company": 0}
full_data_cln = full_data.fillna(nan_replacements)

# "meal" contiene valores "Undefined", que es igual a SC segun las especificaciones del hotel.
full_data_cln["meal"].replace("Undefined", "SC", inplace=True)

# Las filas donde adultos, niños y bebes sean 0, tampoco seran tenidas en cuenta
# Borramos las entradas del dataset
zero_guests = list(full_data_cln.loc[full_data_cln["adults"]
                   + full_data_cln["children"]
                   + full_data_cln["babies"]==0].index)
full_data_cln.drop(full_data_cln.index[zero_guests], inplace=True)

# Despues de la limpieza, el dataset queda de la forma
full_data_cln.shape
```

Para la limpieza de datos se identificó y se gestionaron los valores nulos en columnas críticas, como "children," "country," "agent," y "company."

Eliminamos reservas con un número total de huéspedes igual a cero para garantizar la coherencia de nuestros datos.

Actualizamos la categoría "Undefined" en la columna "meal" a "SC" siguiendo las especificaciones del hotel.

2. Análisis Geográfico:

Se realiza una segmentación en los datos para discriminar las reservas que son canceladas por ciudad

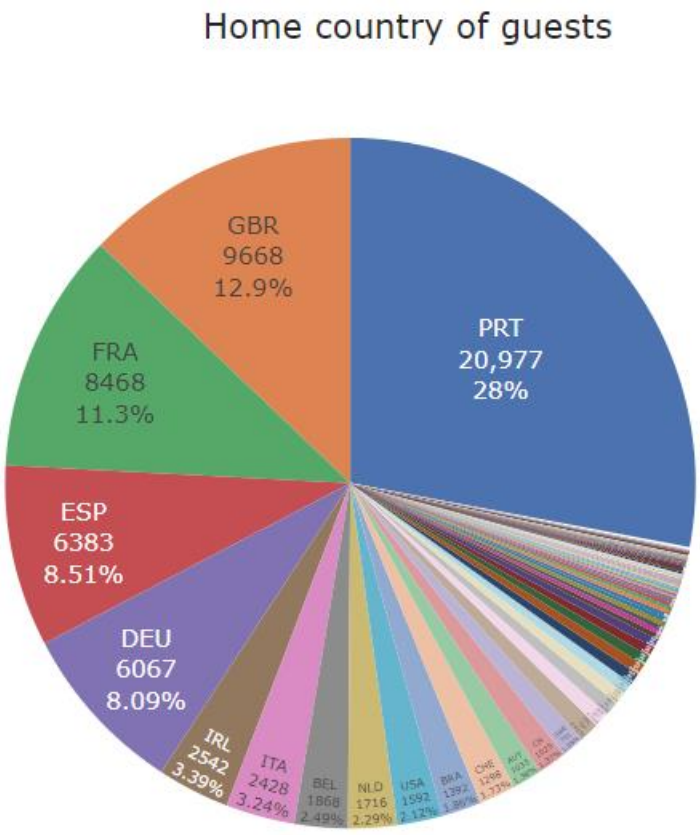
```
# Separamos el Resort del hotel de la ciudad, y para tener una medida exacta de los huespedes eliminamos las reservas que hayan sido canceladas
rh = full_data_cln.loc[(full_data_cln["hotel"] == "Resort Hotel") & (full_data_cln["is_canceled"] == 0)]
ch = full_data_cln.loc[(full_data_cln["hotel"] == "City Hotel") & (full_data_cln["is_canceled"] == 0)]
rh
```

Visualizamos la distribución geográfica de los huéspedes mediante un gráfico de pastel.

Se destacaron países como Portugal y otras naciones europeas como principales contribuyentes al flujo de visitantes.

```
# Obtenemos los huespedes por paises
country_data = pd.DataFrame(full_data_cln.loc[full_data_cln["is_canceled"] == 0]["country"].value_counts())
country_data.rename(columns={"country": "Number of Guests"}, inplace=True)
total_guests = country_data["Number of Guests"].sum()
country_data["Guests in %"] = round(country_data["Number of Guests"] / total_guests * 100, 2)
country_data["country"] = country_data.index

# Hacemos un grafico de barras para ver los resultados
fig = px.pie(country_data,
             values="Number of Guests",
             names="country",
             title="Home country of guests",
             template="seaborn")
fig.update_traces(textposition="inside", textinfo="value+percent+label")
fig.show()
```

De esta sección de código obtenemos los resultados en un diagrama de torta




3. Análisis de Precios:

Calculamos el precio por habitación por persona para el Resort y el Hotel de la Ciudad.

Presentamos la variación mensual de precios mediante un gráfico de líneas, destacando las dinámicas específicas de ambos tipos de hoteles.

4. Modelos de Aprendizaje Automático:

Aplicamos varios modelos predictivos, incluyendo regresión logística, K-Nearest Neighbors (KNN), y Random Forest para prever cancelaciones de reservas.

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

# Definir los hiperparámetros a probar
param_grid_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance']
}

# Crear el modelo KNN
knn = KNeighborsClassifier()

# Realizar la búsqueda de hiperparámetros
grid_search_knn = GridSearchCV(knn, param_grid=param_grid_knn, cv=5)
grid_search_knn.fit(x_train, y_train)

# Obtener los mejores hiperparámetros
best_params_knn = grid_search_knn.best_params_
print("Best Parameters for KNN:", best_params_knn)
```

```
[ ] from sklearn.model_selection import learning_curve
    import matplotlib.pyplot as plt

    # Crear el modelo KNN con los mejores parámetros
    best_knn = KNeighborsClassifier(n_neighbors=best_params_knn['n_neighbors'], weights=best_params_knn['weights'])

    # Obtener curvas de aprendizaje
    train_sizes, train_scores, test_scores = learning_curve(best_knn, x_train, y_train, cv=5)

    # Calcular la media y desviación estándar de los scores
    train_scores_mean = np.mean(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
```

```
print(test_scores_mean)
# Graficar curvas de aprendizaje
plt.figure()
plt.plot(train_sizes, test_scores_mean, 'o-', label='Cross-validation score')
plt.title('Learning Curve for KNN')
plt.xlabel('Training examples')
plt.ylabel('Score')
plt.legend(loc='best')
plt.show()
```

Experimentamos con la técnica de K-Means para etiquetar clusters y exploramos su impacto en los modelos.

Evaluamos el rendimiento de los modelos utilizando métricas de precisión.

5. Mejora de Modelos:

Ajustamos hiperparámetros de KNN y Random Forest mediante GridSearchCV, buscando optimizar su rendimiento.

Presentamos curvas de aprendizaje que ilustran la evolución del rendimiento en función del tamaño del conjunto de entrenamiento.

6. Conclusiones y Perspectivas Futuras:

El modelo de regresión logística demostró una sólida precisión, siendo KNN y Random Forest modelos más complejos que exploramos y mejoramos.

La inclusión de etiquetas de clusters a través de K-Means no resultó en mejoras sustanciales en la precisión de los modelos.

Subrayamos la importancia de seguir afinando nuestros modelos y explorar técnicas avanzadas para perfeccionar las predicciones de cancelación de reservas.

**Iteraciones de desarrollo.**

Luego de procesar los datos para separar los que son de nuestro interés y borrar ciertos datos que estaban erróneos, procedimos a aplicar 3 modelos predictivos para las futuras cancelaciones del hotel. Tomamos como referencia 3 modelos, que son los siguientes:

Modelo de regresión logística:es un algoritmo de clasificación que se utiliza en problemas donde la variable de salida es categórica, con dos o más clases. A través de una función logística, transforma la combinación lineal de las variables de entrada en una probabilidad que se interpreta como la probabilidad de que la observación pertenezca a una clase particular. Es particularmente útil en problemas de clasificación binaria. Las métricas obtenidas en este primer modelo predictivo fueron las siguientes:

```
Classification Report is:
              precision    recall  f1-score   support

           0       0.77      0.88      0.82     15926
           1       0.66      0.46      0.54      7916

    accuracy                           0.74     23842
   macro avg       0.71      0.67      0.68     23842
weighted avg       0.73      0.74      0.73     23842


Confusion Matrix:
 [[14064  1862]
 [ 4309  3607]]


Training Score:
 74.02478922863975
```

Modelo K-Nearest Neighbors (KNN): es un algoritmo de aprendizaje supervisado que se utiliza tanto para problemas de clasificación como de regresión. En el contexto de clasificación, clasifica una observación basándose en la mayoría de las clases de sus "k" vecinos más cercanos en el espacio de características. En regresión, predice el valor medio de los "k" vecinos más cercanos. KNN es un enfoque simple y no paramétrico que se basa en la suposición de que las instancias similares tienden a estar en la misma clase o tener valores de salida similares. Las métricas obtenidas al ejecutar este primer modelo de KNN fueron las siguientes:

```
Accuracy Score of KNN is : 0.8871319520174482
Confusion Matrix :
[[14965   961]
 [ 1730  6186]]
Classification Report :
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     15926
           1       0.87      0.78      0.82      7916

    accuracy                           0.89     23842
   macro avg       0.88      0.86      0.87     23842
weighted avg       0.89      0.89      0.89     23842
```

Modelo RandomForest: es un modelo de ensamble basado en árboles de decisión. Construye múltiples árboles de decisión durante el entrenamiento y los combina para obtener una predicción más robusta y generalizable. Cada árbol se

entrena con un subconjunto aleatorio de datos y características, y luego la predicción final se obtiene promediando las predicciones individuales (en el caso de regresión) o votando por la clase más frecuente (en el caso de clasificación). Las métricas obtenidas con este primer modelo de RandomForest fueron:

```
Accuracy Score of Random Forest is : 0.9659843972821072
Confusion Matrix :
[[15585   341]
 [  470  7446]]
Classification Report :
              precision    recall  f1-score   support

           0       0.97      0.98      0.97     15926
           1       0.96      0.94      0.95      7916

    accuracy                           0.97     23842
   macro avg       0.96      0.96      0.96     23842
weighted avg       0.97      0.97      0.97     23842
```

Luego de tener los 3 primeros modelos predictivos como base de nuestro proyecto, un primer análisis, groso modo, es que de los 3 modelos ejecutados, el de regresión logística presenta una precisión menor (74.025%) que es la métrica en la que basaremos nuestros futuros análisis, y que finalmente intentaremos mejorar. Sentada esta premisa, procedemos a descartar el modelo mencionado de nuestros futuros procesos, incluyendo sólo RandomForest y KNN para el ajuste de hiperparámetros y combinación de modelos con no supervisados.

**AJUSTE DE HIPERPARÁMETROS PARA LOS MODELOS PREDICTIVOS:**
Luego de tener nuestros modelos predictivos, el siguiente paso será encontrar los mejores hiperparámetros para aumentar la precisión de nuestro modelo. Para el caso del modelo de KNN ajustamos los hiperparámetros estableciendo un número de combinaciones basadas en algoritmos encontrados en la web, de la siguiente manera:

```python
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

# Definir los hiperparámetros a probar
param_grid_knn = {
    'n_neighbors': [3, 5, 7],
    'weights': ['uniform', 'distance']
}

# Crear el modelo KNN
knn = KNeighborsClassifier()

# Realizar la búsqueda de hiperparámetros
grid_search_knn = GridSearchCV(knn, param_grid=param_grid_knn, cv=5)
grid_search_knn.fit(x_train, y_train)

# Obtener los mejores hiperparámetros
best_params_knn = grid_search_knn.best_params_
print("Best Parameters for KNN:", best_params_knn)
```

La mejor combinación dentro de las posibles establecidas para los hiperparámetros escogidos (n_neighbors y weights) nos dieron el siguiente resultado:

```
Best Parameters for KNN: {'n_neighbors': 5, 'weights': 'distance'}
```

Para el caso de RandomForest hicimos un procedimiento similar, incluyendo 4 hiperparámetros con la idea de aumentar aún más el rendimiento del modelo:

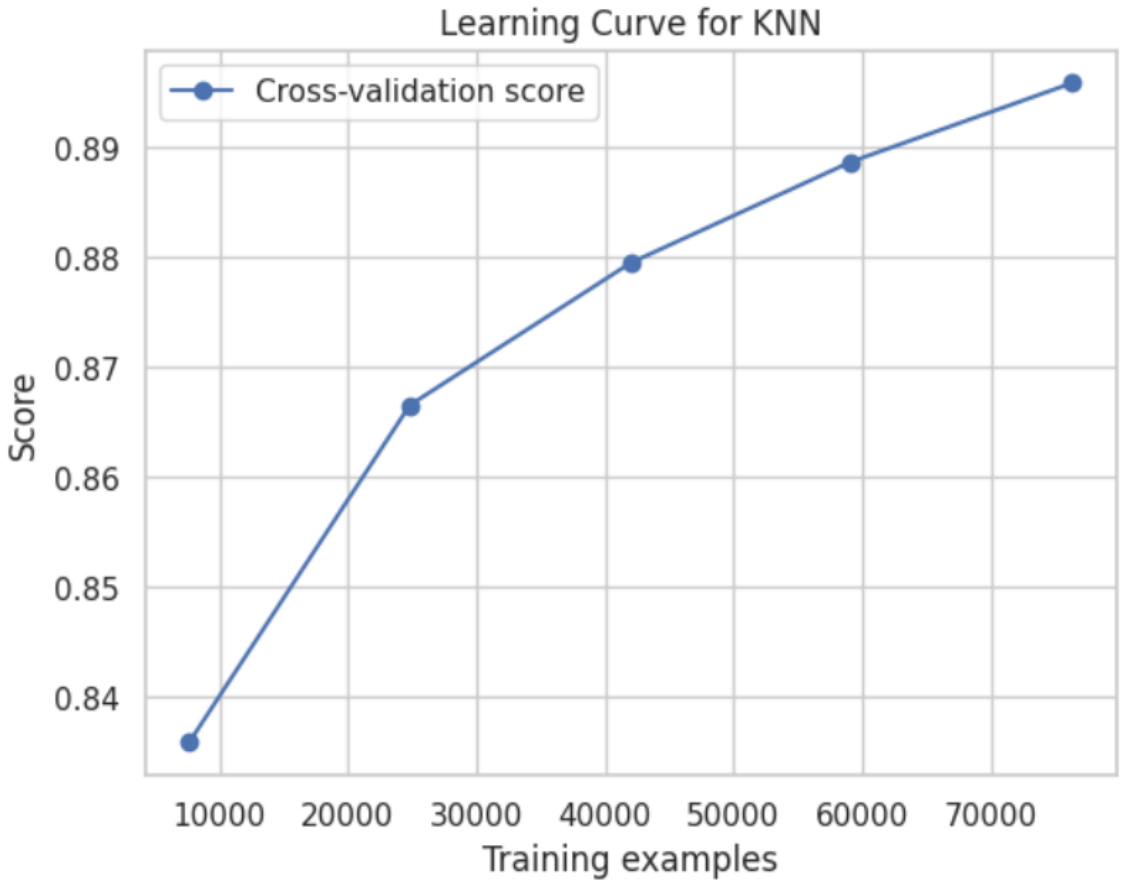
```python
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

# Definir los hiperparámetros a probar
param_grid_rf = {
    'n_estimators': [100],
    'max_depth': [5, 10],
    'min_samples_split': [10],
    'min_samples_leaf': [4]
}

# Crear el modelo Random Forest
rf = RandomForestClassifier()

# Realizar la búsqueda de hiperparámetros
grid_search_rf = GridSearchCV(rf, param_grid=param_grid_rf, cv=5)
grid_search_rf.fit(x_train, y_train)

# Obtener los mejores hiperparámetros
best_params_rf = grid_search_rf.best_params_
print("Best Parameters for Random Forest:", best_params_rf)
```

Los resultados obtenidos para los hiperparámetros establecidos fueron:

```
Best Parameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators': 100}
```

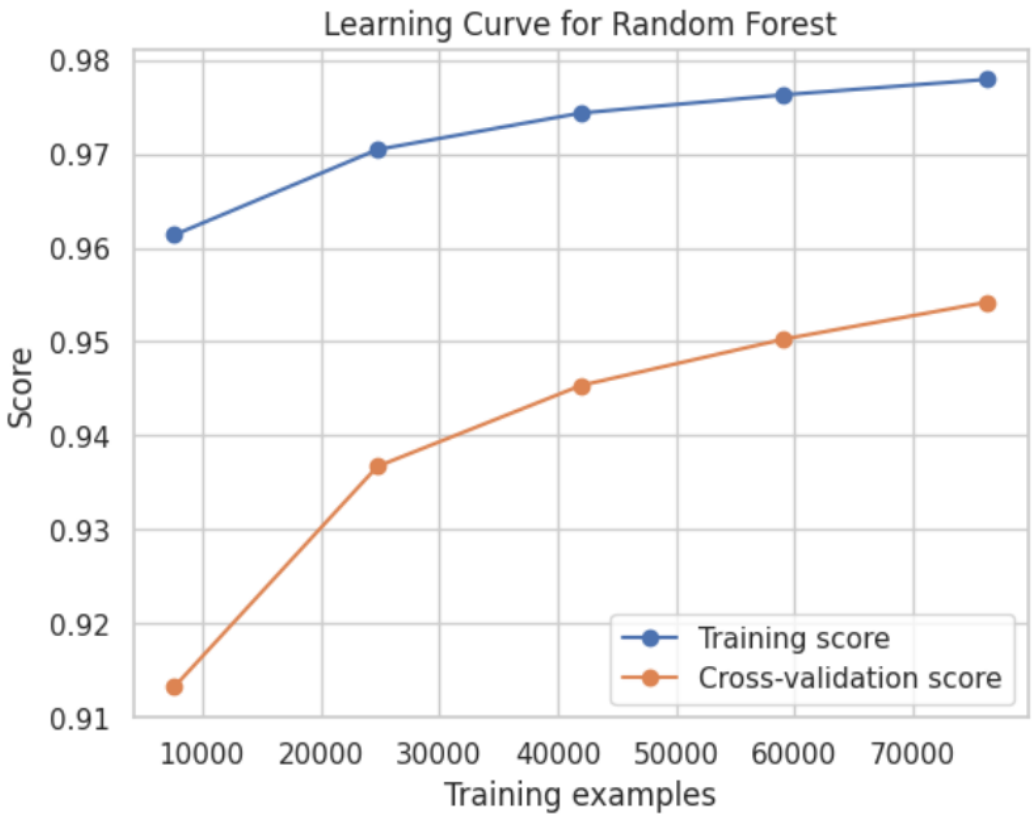
**CURVAS DE APRENDIZAJE:**

Mostraremos, a continuación, las curvas de aprendizaje de los modelos obtenidos con el ajuste de hiperparámetros. Para el modelo de KNN mostraremos la curva de aprendizaje tomando como referencia los datos de prueba:




Efectivamente podemos ver que el modelo va aumentando su rendimiento con respecto a los datos de prueba siempre y cuando aumente el conjunto de datos de entrenamiento. Este fue un resultado esperado pues la idea es que el modelo se entrene de una mejor forma a medida que se le aumentan los datos de entrada.

Evaluaremos ahora las curvas de aprendizaje para el modelo de RandomForest luego de hacerle el ajuste de hiperparámetros pero esta vez tomaremos como referencia el rendimiento del modelo con respecto al conjunto de entrenamiento y al conjunto de prueba:




Podemos ver que el modelo obtiene una mayor precisión al evaluarse con respecto a los datos de entrenamiento comparados con los datos de prueba. Se presenta el incremento de precisión en el modelo esperado al aumentar los datos para ambos casos de los conjuntos de estudio. Con las dos curvas de aprendizaje mostradas, podemos hacer un diagnóstico de overfitting, el cuál va bajando en la medida en la que incrementamos nuestros datos, arrojando una precisión no muy alejada del modelo al evaluar su comportamiento con respecto a los datos de entrenamiento y prueba cuando se tiene un alto número de datos. Podemos concluir que nuestro modelo de RandomForest con los hiperparámetros ajustados, cumple con los requerimientos establecidos y puede ser funcional en un modelo de negocio para la predicción de las cancelaciones del hotel estudiado.

**COMBINACIÓN DE KMEANS + LOS MODELOS PREDICTIVOS USADOS.**

En esta sección haremos una combinación del modelo no supervisado Kmeans, que es un algoritmo de agrupamiento (clustering) que se utiliza para dividir un conjunto de datos en grupos o clusters. Su objetivo principal es asignar cada punto de datos a uno de los "k" clusters, donde "k" es un número predefinido de clusters que se debe especificar antes de ejecutar el algoritmo. El algoritmo trabaja iterativamente para asignar puntos de datos a clusters y ajustar la posición de los centroides (puntos representativos) de los clusters de manera que la suma de las distancias cuadradas entre los puntos y sus centroides respectivos sea mínima. La estrategia fue, entonces, crear un nuevo conjunto de datos de entrenamiento y prueba usando Kmeans, para luego evaluar el rendimiento de los modelos predictivos con sus respectivos ajustes de hiperparámetros con los nuevos

conjuntos de datos obtenidos. La creación del nuevo conjunto de datos con los “clusters” se hizo de la siguiente manera:

```
from sklearn.cluster import KMeans
data = x
# Aplicar K-Means a los datos
kmeans = KMeans(n_clusters=2)  # Define el número de clusters según tu preferencia
kmeans.fit(data)  # Donde 'data' son tus datos sin la variable objetivo

# Obtener las etiquetas de los clusters
cluster_labels = kmeans.labels_
```

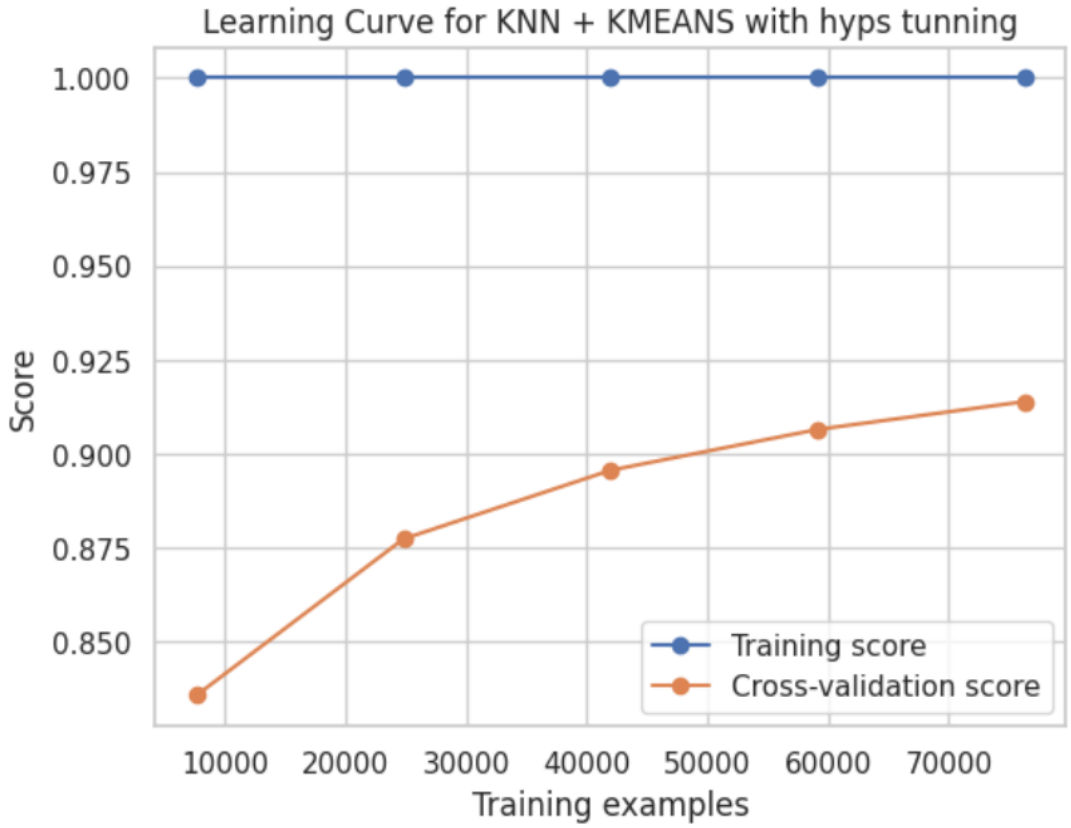
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Combina las etiquetas de clusters con tus datos originales
data_with_clusters = data.copy()
data_with_clusters['cluster_labels'] = cluster_labels

# Dividir los datos en conjunto de entrenamiento y prueba
X_n = data_with_clusters.drop('is_canceled', axis=1)
y_n = data_with_clusters['is_canceled']
X_train_n, X_test_n, y_train_n, y_test_n = train_test_split(X_n, y_n, test_size=0.2, random_state=42)
```

Posterior a esto, evaluamos la precisión de nuestros modelos con los hiperparámetros encontrados previamente y el conjunto de datos nuevos obtenidos. El resultado fue el siguiente:

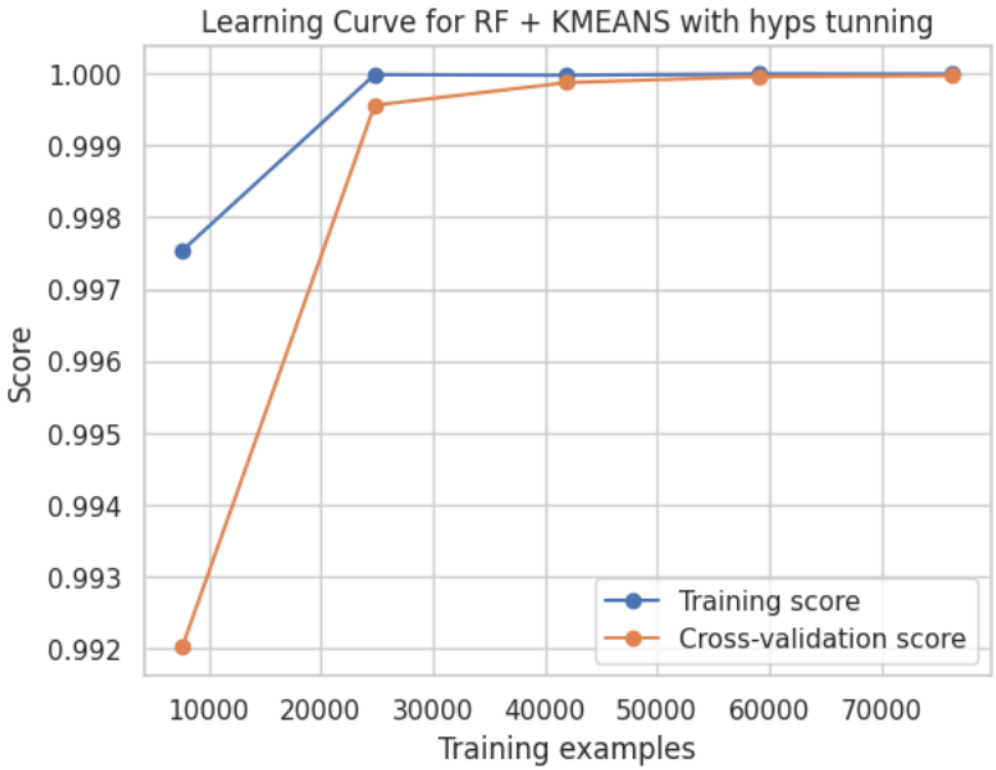
```
Accuracy Score of KNN: 0.9179599026927271
```

```
Accuracy Score of Random Forest + kmeans with hyperparameter tuning: 0.9999161144199312
```

Ahora, las curvas de aprendizaje para nuestros modelos con el nuevo conjunto de datos fueron:

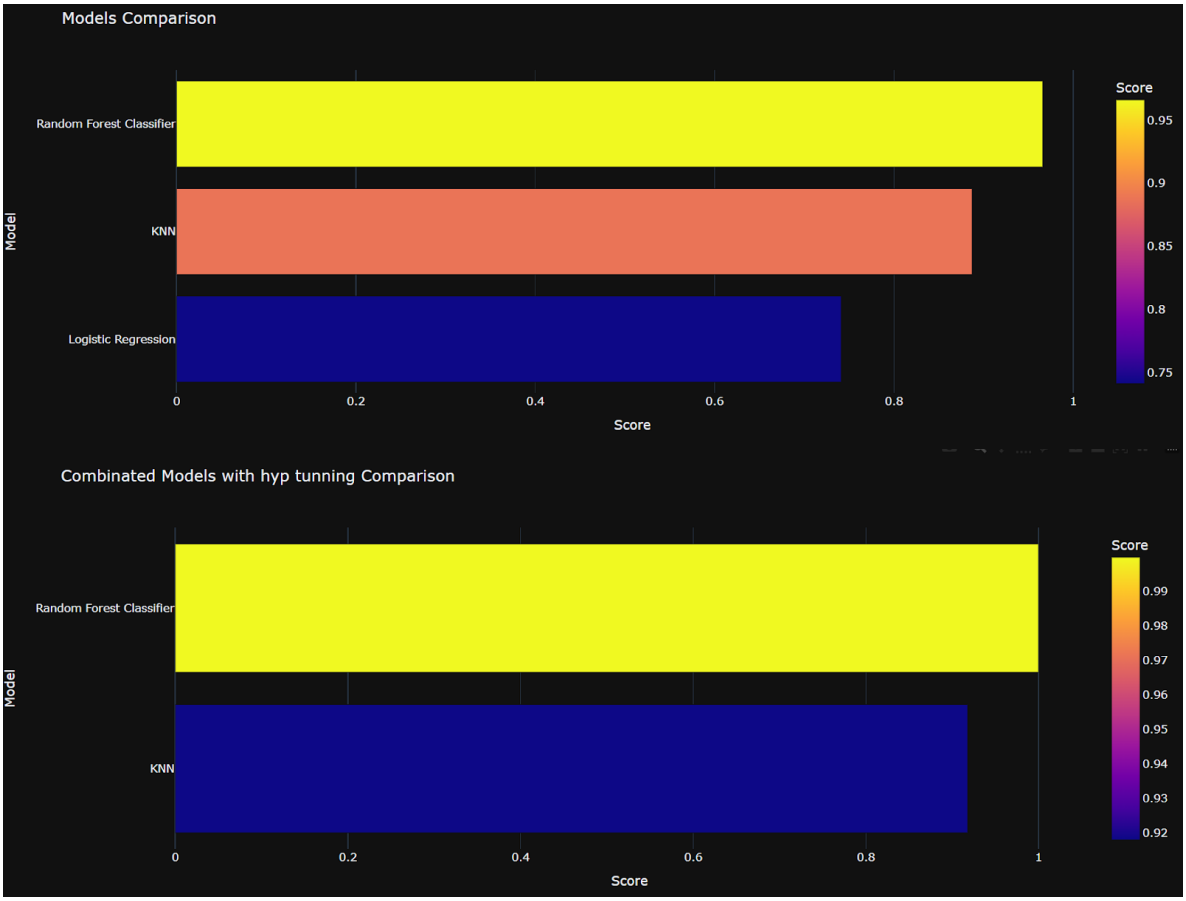



Podemos ver que para el caso de KNN con hiperparámetros +Kmeans se presenta un rendimiento del 100% con respecto a los datos de entrenamiento, este comportamiento no era el esperado pues teníamos la idea de que incrementaría el rendimiento del modelo al aumentar los datos, no que sería constante. El modelo, en cambio, aumenta su rendimiento con respecto a los datos de prueba a medida que el conjunto de entrenamiento se hace más grande. Podemos ver que existe una diferencia clara del orden del 9% de precisiones, en el mejor de los casos, en los rendimientos del modelo para cada caso de estudio, lo que nos da un índice de overfitting en nuestro modelo de aprendizaje.




Para el caso de RandomFores con hiperparámetros ajustados + Kmeans, podemos ver que el rendimiento del modelo cuando los datos pasan de 20000, tiende a ser del 100% con respecto a los datos de entrenamiento. Mientras tanto, podemos ver que la precisión del modelo con respecto a los datos de prueba es también muy cercana a 1 al aumentar los datos. Realmente la precisión en general de este modelo es bastante alta, del orden del 99% desde los 10000 datos de entrenamiento. La razón de este hecho es que para este modelo se hizo un mayor ajuste de hiperparámetros, por lo que se cree que el modelo tiene un rendimiento más acertado.

Haremos, por último, una comparación del rendimiento de los 3 primeros modelos que se tuvieron como referencia. Podemos observar en las siguientes imágenes una primera comparación del rendimiento de los modelos de regresión logística, RF y KNN. Como se había dicho anteriormente, el modelo de regresión logística presenta la más baja precisión, por lo que se descarta de los futuros análisis. Por otro lado, observamos que la precisión de nuestros modelos de RF y KNN aumentó hasta un 99% y 91% respectivamente, luego de aplicar un ajuste de hiperparámentros y una combinación de Kmeans para mejorar los conjuntos de datos de entrenamiento para nuestros modelos. Se alcanzó, entonces, el logro propuesto que era finalmente llegar a una mayor precisión en nuestros modelos predictivos, que en principio se había fijado mayor a un 90%.




**RETOS Y CONSIDERACIONES DE DESPLIEGUE:**

Como vimos, el modelo de RF tiende a ganar una mayor precisión que el modelo de KNN en todas las cantidades de datos para los conjuntos de entrenamiento. Esto se da debido a un mejor ajuste de hiperparámetros, por lo cual el reto para aumentar el rendimiento de algún otro modelo a usar es efectivamente hacer un arduo ajuste de hiperparámetros, brindando un mayor espectro de combinaciones para hallarlos. Esto requiere de tiempo y de una buena unidad de cómputo para optimizar los procesos. Aunque con nuestro modelo de RF creemos que se puede pensar en una opción de despliegue de oportunidad de negocio. Este modelo se puede implementar en el hotel para así tener un mejor balance de inversión/rentabilidad en el flujo de caja del hotel. Es decir, con las predicciones de las futuras cancelaciones se puede establecer un abastecimiento general y más consciente de los recursos necesarios del hotel para suplir las necesidades de los clientes que demanden los servicios. Disminuyendo finalmente los sobrecostos y sobreabastecimiento, no sólo se aumentan las ganancias del hotel, sino que se disminuiría el exceso de desperdicios de comida, en el caso de la compra de alimentos, por ejemplo.

## CONCLUSIONES.

Este análisis exhaustivo proporciona un fundamento robusto para futuras mejoras y desarrollos en la gestión de reservas hoteleras. Nuestra combinación de exploración detallada y aplicación de técnicas de aprendizaje automático nos posiciona herramientas para abordar eficazmente los desafíos de la industria hotelera, optimizando la eficiencia operativa, que es el enfoque dado al querer optimizar los recursos de la empresa. Este enfoque integral nos permite no solo comprender los patrones de comportamiento pasados, sino también anticiparnos a las futuras cancelaciones como medidas de prevenciones de sobrecostos y contribución a la disminución del consumismo generado por las cadenas hoteleras, como parámetro importante referente a nuestro compromiso ambiental.