ENTREGA FINAL PROYECTO INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

ALEJANDRO HERRERA RIVERA

JUAN JOSÉ ISAZA LUJÁN

DOCENTE:

Raúl Ramos Pollán

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

MEDELLÍN -ANTIOQUIA

Tabla de contenido

Introducción

El presente trabajo se refiere a la implementación de un algoritmo de regresión para el problema planteado en la competición "*ADAMS SoSe23*" en Kaggle. Donde el objetivo se enfoca en predecir el precio de nuevas viviendas para Airbnb basado en sus características.

1. Descripción del problema predictivo

En el competitivo mercado del alquiler de propiedades tanto los huéspedes como los anfitriones siempre tienen la inquietud de saber ¿cómo determinar el precio adecuado para una propiedad dada? Los anfitriones desean establecer tarifas competitivas que atraigan a huéspedes potenciales, maximicen sus ingresos y reflejen con precisión el valor de su propiedad. Por otro lado, los huéspedes buscan ofertas atractivas que se ajusten a su presupuesto mientras satisfacen sus necesidades de alojamiento.

Sin embargo, la fijación de precios en este mercado dinámico y diverso es una tarea complicada. Los precios pueden variar significativamente según la ubicación, el tipo de propiedad, el número de habitaciones, las comodidades ofrecidas y otros factores.

El trabajo consta de la creación de un modelo de predicción de precios para propiedades de Airbnb en Londres, utilizando datos que incluyen información sobre la ubicación, características de las propiedades y descripciones de los anuncios.

1.1 Descripción del dataset

El dataset a utilizar proviene de una competencia de kaggle en la cual se proporcionan datos de las propiedades de Airbnb en Londres. El dataset está compuesto por un conjunto de archivos .csv que proporcionan la información requerida tales como la descripción de las variables, los datos de entrenamiento y los datos de prueba.

1.2 Descripción de las variables

name = Nombre de la propiedad en Airbnb
summary = Resúmen de sus características
space = Descripción de sus áreas
description = Resúmen de sus características
experiences_offered = Experiencias ofrecidas
neighborhood_overview = Descripción del barrio
transit = Descripción de la locación y el transporte
house_rules = Reglas de la propiedad
picture_url = Link de la imágen
host_id = Identificador del anfitrión
host_since = Fecha desde que es anfitrión
host_response_time = Tiempo de respuesta del anfitrión
host_response_rate = Índice de respuesta del anfitrión
host_is_superhost = El anfitrión es superanfitrión (V o F)
host_total_listings_count = Recuento de listados del anfitrión
host_has_profile_pic = El anfitrión tiene foto de perfil (V o F)
host_identity_verified = Identidad del anfitrión verificada (V o F)
neighbourhood = Barrio

neighbourhood_cleansed = Distrito
zipcode = Código postal
latitude = Latitud
longitude = Longitud
property_type = Tipo de propiedad
room_type = Tipo de habitación
accommodates = #Capacidad
bathrooms = #Baños
bedrooms = #Habitaciones
beds = #Camas
bed_type = Tipo de cama
amenities = Servicios
price = Precio (Variable objetivo)
guests_included = #Huéspedes incluidos en el precio
review_scores_rating = Puntuación del huésped
review_scores_accuracy = Precisión en las puntuaciones
review_scores_cleanliness = Puntuación a la limpieza
review_scores_checkin = Puntuación experiencia en el check-in
review_scores_communication = Puntuación comunicación con el anfitrión
review_scores_location = Puntuación a la ubicación de la propiedad
review_scores_value = Puntuación a la relación calidad-precio
cancellation_policy = Política de cancelación del anfitrión
reviews_per_month = Reviews por mes
listing_id = Id listado propiedad

1.3 Métrica de machine learning

La métrica de evaluación principal del modelo es el RMSE (Root Mean Squared Error). Se utiliza para medir la precisión de un modelo de regresión al evaluar cuán cerca están las predicciones del modelo de los valores reales.

$$RECM = \sqrt{\frac{\sum(O_i - E_i)^2}{n}}$$

donde O_i son los valores observados;

E_i son los valores esperados;

∑ es una letra griega llamada sigma que representa la «sumatoria»; y

n es el tamaño de la muestra (el número de observaciones).

1.4 Métrica de negocio y desempeño deseable en producción

La métrica de negocio está basada en que el valor predicho para la propiedad se ajuste adecuadamente

a lo que ofrece, ya que sería injusto para una persona pagar un valor grande por un servicio que realmente no cumple con las necesidades que los clientes requieran. Además permite establecer un precio adecuado para un anfitrión que desea ingresar a la aplicación ofreciendo un lugar.

Además se tendrá en cuenta la métrica R cuadrado que permite saber en cuanto el modelo se ajusta a los datos reales.

Por la razón anterior se espera que el RMSE sea lo más cercano a cero y un R cuadrado cercano a 1, es decir que el modelo sea capaz de tener un buen ajuste para predecir con precisión en relación con los valores reales. Indicando que es capaz de aprender de las características de las propiedades para definirles un precio ajustado y relacionado a lo que ofrece el servicio.
Finalmente para airbnb es importante tener precios competitivos, ofrecer una grata experiencia y aumentar la confianza en su servicio, además un buen ajuste del valor de los servicios permite comodidad tanto para los huéspedes como para los anfitriones.

2. Análisis exploratorio (DEA)

El dataset cuenta con 55284 filas y 42 columnas. Al ver la tabla inicial se pueden ver que algunas variables no se tendrán en cuenta porque son descripciones de las residencias, entre estas encontramos name, summary, space, description entre otras. Siendo un total de 11 variables que se eliminarán para facilitar el manejo del dataset.

2.1 Variable objetivo

Se realizó un histograma para observar el comportamiento de la variable objetivo, observando que la mayor cantidad de precios se encuentran entre los 50 a los 100 dólares:

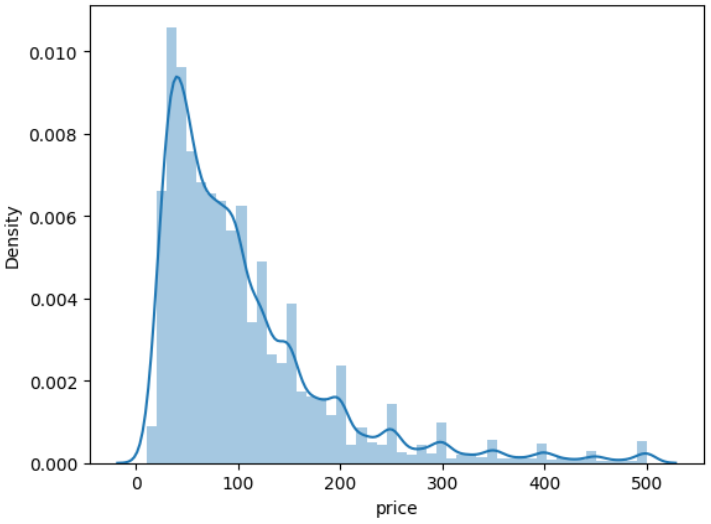

Figura 1. Distribución de la variable objetivo.

2.2 Análisis de variables numéricas

Se realiza un análisis visual de las variables numéricas mediante histogramas para comprender mejor sus distribuciones.

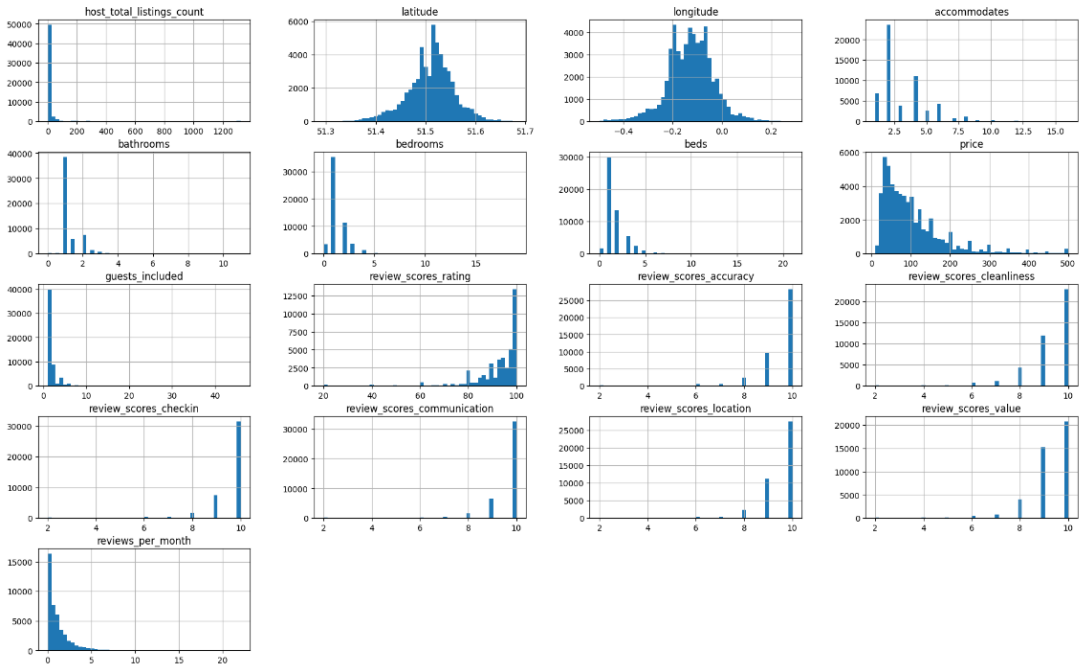

Figura 2. Distribución de variables numéricas.

- Tanto la cantidad de reviews por mes como los invitados que incluye indican una asimetría hacia la izquierda, indicando que la mayor parte de los valores son bajos, sin embargo existen valores altos que provocan esta asimetría.
- Las puntuaciones de las reviews por lo general son positivas, siendo el puntaje 10 el que tiene las mayores cantidades de reviews para los datos.
- Tanto la cantidad de habitaciones como la cantidad de camas tienen cantidades similares, siendo el valor de (1) el que mayor valores tiene en ambas variables.

2.3 Correlación entre variables numéricas

Se realiza una matriz de correlación entre las variables numéricas, para conocer si pueden existir problemas de multicolinealidad al haber 2 o más variables independientes altamente relacionadas:

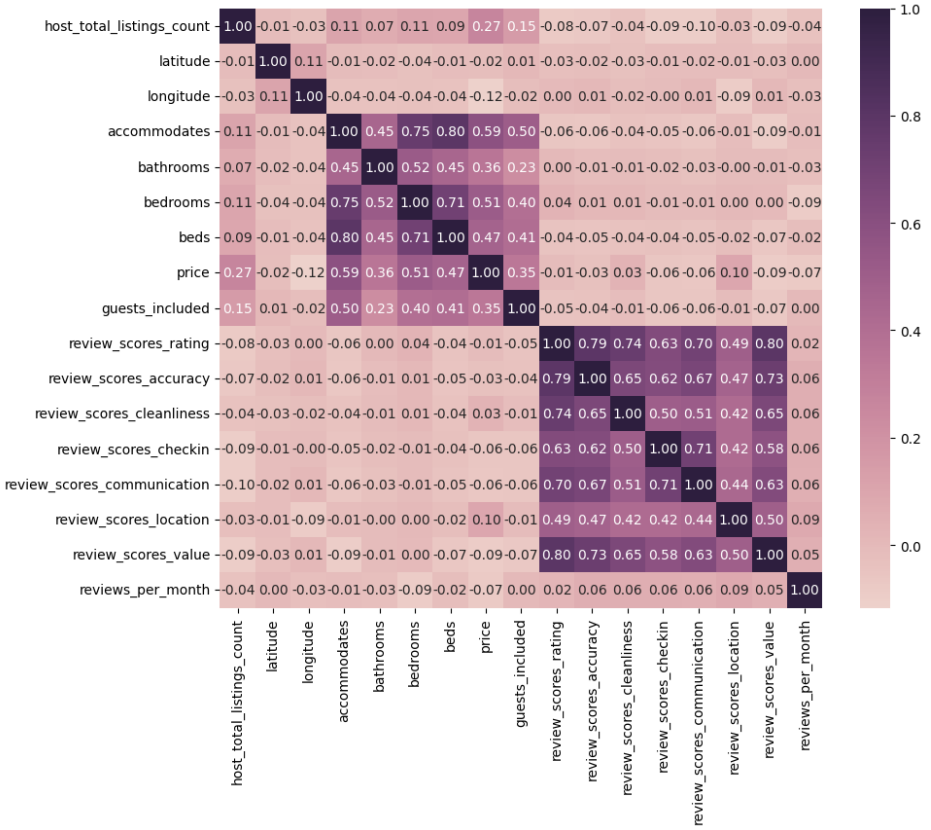

Figura 3. Matriz de correlación entre variables

Las variables de reviews de los clientes presentan bastante correlación entre ellas, como por ejemplo "review scores rating", la cual presenta los valores de correlación más altos con las demás variables de reviews, siendo en casi todos los casos mayor a 0.6, al igual que la variable de capacidad "accommodates" se correlaciona mucho con el número de camas "beds", el número de habitaciones "bedrooms" y el precio. Siendo asi 'review_scores_rating', 'accommodates', 'bedrooms', 'review_scores_accuracy' las cuatro variables que se eliminarán para crear una tabla nueva limpia de esta multicolinealidad.

En cuanto a las variables categóricas se decide aplicar la prueba de chi-cuadrado, seleccionando los resultados mayores a 0.05 los cuales nos indican una alta correlación:

```
('experiences_offered', 'host_has_profile_pic', 0.7262523248759465)
('host_response_time', 'host_has_profile_pic', 0.06519513735833132)
('host_has_profile_pic', 'experiences_offered', 0.7262523248759465)
('host_has_profile_pic', 'host_response_time', 0.06519513735833132)
('host_has_profile_pic', 'neighbourhood', 0.5249147652569358)
('host_has_profile_pic', 'property_type', 0.28057414590348084)
('host_has_profile_pic', 'bed_type', 0.8739667231613887)
('neighbourhood', 'host_has_profile_pic', 0.524914765256935)
('neighbourhood', 'bed_type', 0.06702526850700927)
('property_type', 'host_has_profile_pic', 0.28057414590348084)
('bed_type', 'host_has_profile_pic', 0.8739667231613887)
('bed_type', 'neighbourhood', 0.06702526850700927)
('bed_type', 'cancellation_policy', 0.8928835121337191)
('cancellation_policy', 'bed_type', 0.892883512133719)
```

Figura 4. Prueba Chi-cuadrado variables categóricas

Las variables "host has profile_pic" y "bed_type" se correlacionan mucho con otras variables, por lo que se decide eliminarlas para la tabla limpia.

2.4 Datos faltantes

Se observó que "host_response_rate" y "host_response_time" son las variables que presentan la mayor cantidad de datos faltantes, con un porcentaje de 32.2% cada una y un total de 17802 datos nulos.

Las variables de ‘reviews’ son las siguientes con mayor cantidad de nulos, cada una con un porcentaje cercano al 24% de datos faltantes.

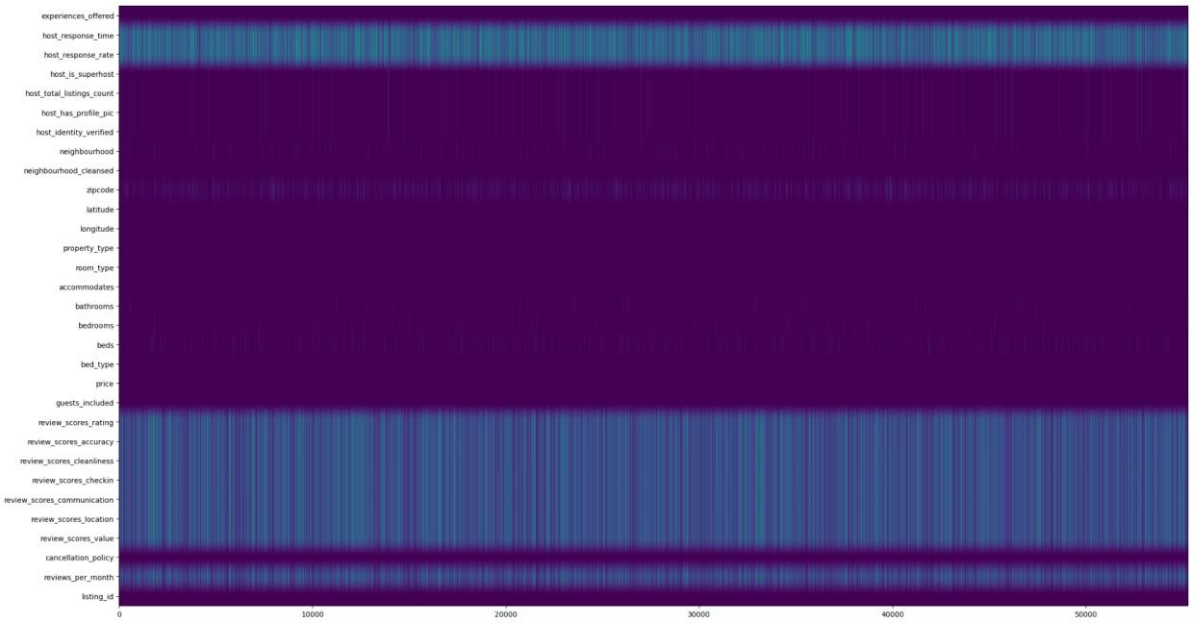

Figura 5. Visualización de datos faltantes

Se eliminan las variables ‘listing_id’ y ‘zipcode’, debido a su gran cantidad de datos únicos, ya que al hacer uso de variables dummies para la generación de modelos estas variables generan más de 80.000 columnas dificultando el uso del dataset.

3. Tratamiento de datos

3.1 Eliminación de variables

- **name, summary, space, amenities, description, neighborhood_overview :** Se elimina las variables por contener mucho texto o descripciones que no aportan información relevante para el entrenamiento de modelos.
- **host_id, listing_id** : Se eliminan las variables con solamente valores únicos ya que no aporta información relevante.
- **zipcode :** Se elimina la variable de código postal por su gran cantidad de categorías, lo que dificulta la manipulación de la tabla.
- **picture_url, host_since, transit, house_rules :** De igual manera se eliminan estas variables por contener muchas categorías o valores con mucho texto que dificulta el análisis de modelos.

3.2 Imputación de nulos

Ya que no hay variables con una cantidad de datos nulos tan significativa que supere el 50% de sus datos, se decide no eliminar ninguna variable. En este caso se optará por llenar esos datos con la mediana en el caso de las variables numéricas para que los datos no se vean afectados por los datos atípicos que pueden influir la media. Las variables categóricas se llenarán con la moda.

3.3 Transformación de variables categóricas

Después de identificadas y tratadas las variables categóricas, se utiliza el siguiente bloque de código para transformarse y ser utilizadas en el modelo:

```
tabla_completa=pd.get_dummies(tabla)
tabla_completa
```

Figura 5. Código dummies variables categóricas.

4. Métodos supervisados

4.1 Métrica de machine learning

Como se mencionó anteriormente, la métrica de machine learning a utilizar será el RMSE dado que se está ante un problema de regresión y queremos conocer el error que existe entre los datos predichos con los reales.

$$RECM = \sqrt{\frac{\sum(O_i - E_i)^2}{n}}$$

4.2 Partición de los datos

Inicialmente se separa la variable objetivo del dataset original para la creación de los modelos predictivos, dejando una base X y la variable objetivo y. Para el entrenamiento de los modelos se divide el dataset en 4 partes, generalmente en unos datos de entrenamiento X_train, y_train con una proporción del 70% de los datos totales, y unos datos de validación X_test, y_test con el 30% de los datos restantes para calcular la métrica de desempeño.

```
X = tablabase1.drop(['price'],axis = 1)
y = tablabase1['price'].values
```

Figura 8. Partición de los datos.

4.3 Selección de modelos

Dado que estamos ante un problema de regresión, seleccionamos los algoritmos que más nos interesan como lo son la Regresión Lineal, Desicion Tree y Random Forest.

4.4 Regresión Lineal

Inicialmente se probó el modelo de regresión lineal ya que es un modelo simple y fácil de entender basado en una ecuación lineal relacionando variables de forma directa o inversa.

```
# Crea el modelo
modelo_rls_1 = LinearRegression()

# Calibra el modelo
modelo_rls_1.fit(x_train_1, y_train_1)
```

Figura 10. Regresión lineal

El modelo de regresión lineal se probó con la base completa sin la eliminación de variables por correlación y una base limpia sin las variables que más se correlacionaron para evaluar el resultado de la métrica en ambos casos y seguir el trabajo con la base de mejor resultado.

```
RMSE train 54.26
RMSE test 54.24

Variance score train: 0.58
Variance score test: 0.57

R2-adjusted train: 0.58
R2-adjusted test: 0.56
```

Figura 11. Métricas tabla completa

```
RMSE train 56.58
RMSE test 56.59

Variance score train: 0.55
Variance score test: 0.53

R2-adjusted train: 0.54
R2-adjusted test: 0.52
```

Figura 12. Métricas tabla limpia

Los resultados del RMSE son muy similares, de 54.24 para la tabla completa y de 56.59 para la tabla limpia, sin embargo podemos notar que la eliminación de estas variables provocan una pérdida de información que afecta el resultado de las métricas, por lo cual se decide seguir utilizando la tabla completa para los demás modelos.

4.5 Árbol de decisión

Los árboles de decisión son un tipo de algoritmo supervisado que no requiere una gran preparación previa de los datos para que funcione, por lo que facilita el proceso y puede predecir tanto variables numéricas como categorías

```
estimator2 = DecisionTreeRegressor(max_depth=5)
```

Figura 13. Árbol de decisión

```
RMSLE Test:  59.34599 (± 0.60334124 )
RMSLE Train:  58.65869 (± 0.39381273 )
```

Figura 14. RMSE Árbol de decisión base

Para este primer caso nuestra métrica RMSE empeoró con respecto al modelo de regresión lineal como se evidencia en la Figura 14, sin embargo se puede buscar aquellos valores óptimos en los hiperparámetros para que el modelo se ajuste y pueda tener un mejor rendimiento.

4.5.1 Mejores hiperparámetros para el árbol de decisión

Ante los resultados del numeral anterior se usará GridSearchCV para encontrar el `'max_depth'` óptimo según el dataset.

```
parametros = {'max_depth': [2,5,8,12,15]}


decission_tree = GridSearchCV(estimator = estimator2,
                              param_grid = parametros,
                              cv = ShuffleSplit(n_splits= 5, test_size=val_size),
                              scoring = 'neg_mean_squared_error',
                              verbose = 1,
                              return_train_score = True,
                              n_jobs = -1)

decission_tree.fit(Xtv, ytv)
```

Figura 15. Búsqueda de los mejores hiperparámetros para árbol de decisión

Con el código anterior se encuentran que: `Mejor estimador Decision Tree: DecisionTreeRegressor(max_depth=8)`

Observamos que el GridSearchV encuentra la profundidad del árbol que nos dará mejor resultado, obteniendo un RMSE de: `RMSE train 53.59, RMSE test 56.23` Siendo unos resultados un poco más confiables.

4.6 Random Forest

Al igual que el modelo de árboles de decisión el random forest es un modelo de aprendizaje supervisado que no requiere un procesamiento muy complejo de los datos antes de su implementación.

```
estimator3 = RandomForestRegressor(n_estimators = 2,max_depth = 5)
```

Figura 16. Modelo Random Forest

```
RMSLE Test:  58.98010 (± 0.65841793 )
RMSLE Train:  57.79868 (± 0.58061561 )
```

Figura 17. RMSE Random Forest base

En este caso la métrica de desempeño RMSE también empeoró con respecto a la regresión lineal, por lo que se le aplicará optimización de hiperparámetros.

4.6.1 Mejores hiperparámetros para Random Forest

Nuevamente se usará el GridSearchCV para encontrar los valores de `‘n_estimators’` y `‘max_depth’` que mejor se ajusten al modelo.

```
parametros = { 'n_estimators': [5,10,15],
               'max_depth':[5,7,9]}

forest_reg = GridSearchCV(estimator = estimator3,
                          param_grid = parametros,
                          cv = ShuffleSplit(n_splits= 5, test_size=val_size),
                          scoring = 'neg_mean_squared_error',
                          verbose = 1,
                          return_train_score = True,
                          n_jobs = -1)
forest_reg.fit(Xtv, ytv)
```

Figura 18. Búsqueda de los mejores hiperparámetros para Random Forest.

Con el código anterior se encuentran que: `Mejor estimador Random Forest: RandomForestRegressor(max_depth=9, n_estimators=15)`

El GridSearchV encuentra la profundidad del árbol de 9 y 15 árboles para tener la mejor estimación, obteniendo métricas de: `RMSE train 50.06, RMSE test 53.41` Siendo los mejores resultados que hemos encontrado hasta el momento.

5. Métodos no supervisados

5.1 PCA + Random Forest

En este caso se realizó una reducción de la dimensionalidad mediante PCA teniendo en cuenta diferentes números de componentes, para posteriormente implementar un Random Forest que generase las predicciones.

```
from sklearn.decomposition import PCA
components = [1,3,5,7,9]
test_size = 0.3
val_size  = test_size/(1-test_size)
perf = [] #desempeños de los modelos
Rdm_forest = RandomForestRegressor(n_estimators = 5,max_depth = 9)
for i in components:
    pca = PCA(n_components = i)
    X_tns = pca.fit_transform(X_ns)
```

Figura 19. Reducción de dimensionalidad PCA + Random Forest.

El resultado del código anterior es el siguiente:

```
(38595, 1) (16542, 1)
RMSLE del modelo con  1 elementos:  74.28238
------------
(38595, 3) (16542, 3)
RMSLE del modelo con  3 elementos:  61.13218
------------
(38595, 5) (16542, 5)
RMSLE del modelo con  5 elementos:  59.59795
------------
(38595, 7) (16542, 7)
RMSLE del modelo con  7 elementos:  59.94273
------------
(38595, 9) (16542, 9)
RMSLE del modelo con  9 elementos:  58.05684
------------
Mejor RMSLE:  58.05684   ; obtenido con  9  componentes para PCA
```

Figura 20. Número de componentes para PCA + Random Forest.

Con los 9 componentes que nos dice la reducción de dimensionaldiad podremos aplicar nuevamente optimizador de hiperparámetros para el caso del Random Forest : `Mejor estimador Random Forest: RandomForestRegressor(max_depth=9, n_estimators=15)`. Obteniendo los siguientes resultados.

```
RMSLE del Random Forest en entrenamiento:  52.60677
RMSLE del Random Forest validación:  57.66817
```

Figura 21. Métricas para PCA + Random Forest.

5.2 NMF + Árbol de decisión

De manera similar al procedimiento anterior, implementamos el algoritmo NMF para descomponer nuestra matriz de entrenamiento y buscar el número de elementos óptimo para obtener el mejor RMSE al implementar un árbol de decisión después de haber reducido el tamaño de la matriz de entrenamiento.

Primeramente asegurarnos de borrar todas las variables con datos negativos, ya que el modelo NMF no funciona si hay este tipo de datos. En nuestro caso solo teníamos una variable con datos negativa que es `'longitude'`, Por lo cuál será eliminada antes de la implementación.

```
(38595, 1) (16542, 1)
RMSLE del modelo con  1 elementos:  82.92100
------------
(38595, 3) (16542, 3)
RMSLE del modelo con  3 elementos:  71.64687
------------
(38595, 5) (16542, 5)
RMSLE del modelo con  5 elementos:  72.30521
------------
(38595, 7) (16542, 7)
RMSLE del modelo con  7 elementos:  72.88217
------------
(38595, 9) (16542, 9)
RMSLE del modelo con  9 elementos:  70.74854
------------
Mejor RMSLE:  70.74854   ; obtenido con  9  componentes para PCA
```

Figura 22. Número de componentes para NMF + Árbol de decisión

Nuevamente se recomienda una reducción a 9 componentes para obtener más acierto en el modelo y nuevamente ajustamos los hiperparámetros para obtener la mejor métrica donde: `Mejor estimador Decision Tree:  DecisionTreeRegressor(max_depth=5).` Obteniendo:

```
RMSLE del Decision Tree en entrenamiento:  57.62298
RMSLE del Decision Tree seleccionado:  61.72791
```

Figura 23. Métricas para NMF + Árbol de decisión.

6. Curvas de Aprendizaje

6.1 Árbol de decisión

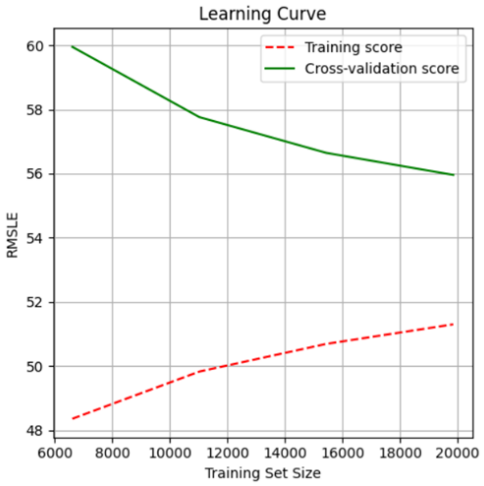

Figura 24. Curva de aprendizaje-Árbol de decisión

La figura 24 muestra al inicio de la implementación del modelo, cuando existen pocos datos, una disparidad con respecto a los resultados en entrenamiento y validación. Al inicio de la implementación del modelo, se observa poca precisión en los datos arrojados por la validación, esto debido a la poca densidad de datos disponibles, sin embargo, se observa un aumento en la precisión a medida que el modelo recibe mayor densidad de datos. Con respecto al comportamiento de los datos obtenidos durante el entrenamiento y la validación, podemos observar una disparidad continua durante la implementación del modelo, aunque mientras más densidad de datos hay esta brecha se hace cada vez más pequeña, sin embargo podemos decir que tiene Overfitting.

6.2 Random Forest

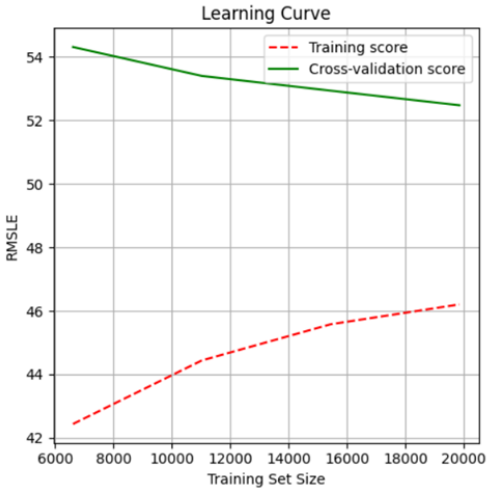

Figura 25. Curva de aprendizaje-Random forest

En la figura 25, se aprecia que tiene un comportamiento similar a la figura 24, en la que al inicio del modelo con una baja densidad de datos se aprecia una gran brecha entre los datos de entrenamiento y los de validación pero al aumentar la densidad de los datos se aprecia cómo la brecha se va cerrando entre los dos conjuntos de datos, sin embargo también estamos en caso de Overfitting por su gran dispariedad.

6.3 PCA + Random Forest

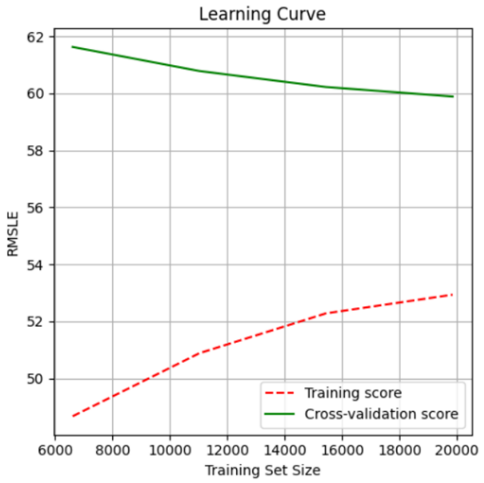

Figura 26. Curva de aprendizaje-PCA+Random Forest.

En la etapa inicial de la implementación del modelo, la Figura 26 revela una disparidad entre los resultados de entrenamiento y validación debido a la escasez de datos. En este punto, la precisión en la validación es baja debido a la limitada cantidad de datos disponibles. A medida que el modelo se expone a más datos, se observa un aumento en la precisión de la validación. A pesar de esto, se puede concluir que hay un fenómeno de sobreajuste (overfitting), ya que la disparidad persiste, aunque disminuye con datos más abundantes.

6.4 NMF + Árbol de decisión

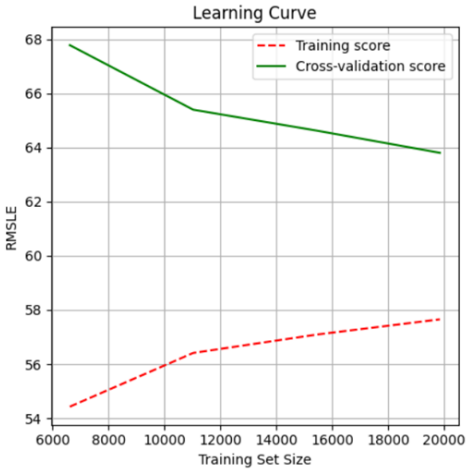

Figura 27. Curva de aprendizaje-NMF+Árbol de decisión.

En la figura 27, se evidencia un patrón similar al observado en las figuras anteriores. En el inicio del modelo, con una escasa densidad de datos, se observa una marcada diferencia entre los resultados de entrenamiento y validación. A medida que la densidad de datos aumenta, se nota cómo esta brecha se reduce entre ambos conjuntos de datos. No obstante, al igual que en el caso anterior, persiste el fenómeno de sobreajuste, ya que la disparidad entre los conjuntos de datos sigue siendo considerable incluso con el aumento de la densidad de datos.

Consideraciones para mejorar: Al desarrollar estos modelos de aprendizaje no tuvimos en cuenta varios hiperparámetros que pueden ayudar a mejorar considerablemente la métrica, como por ejemplo el `weight_class = 'balance'.` Además de jugar un poco más con las variables a eliminar, ya que puede haber problemas de multicolinealidad que no son muy evidentes a primera vista.

7. Retos y condiciones de despliegue del modelo

Para evaluar el despliegue en producción de un modelo de predicción sobre el precio de los Airbnb de Londres, es necesario considerar diversos retos y condiciones. Estos aspectos son fundamentales para asegurar que el modelo funcione de manera efectiva y genere valor en el entorno de mercado.

Nivel de desempeño mínimo: Dado que el RMSE es una métrica que mide la diferencia promedio entre las predicciones y los valores reales, un valor de 53.41 indica que, en promedio, las predicciones del modelo tienen una desviación de 53 dólares con respecto a los precios reales. Es crucial considerar la aceptabilidad de esta diferencia en el contexto del mercado inmobiliario y los objetivos del negocio. La colaboración con expertos en el mercado y partes interesadas puede ayudar a determinar si este nivel de desempeño es aceptable o si se requiere una mejora.

Despliegue en producción:
Implementación de infraestructura: Asegurarse de contar con la infraestructura necesaria para alojar y ejecutar el modelo en tiempo real, considerando la capacidad de procesamiento y almacenamiento adecuada para el manejo de datos.
Integración de datos: Garantizar la integración de datos relevantes para las predicciones, incluyendo información actualizada sobre listados de Airbnb, características de propiedades y factores de mercado.
Implementación del modelo: Llevar a cabo la implementación del modelo en el entorno de

producción, asegurándose de que esté listo para realizar predicciones en tiempo real y manejar solicitudes de manera eficiente.

Pruebas:
Pruebas de integración: Verificar que todas las partes del sistema interactúen de manera adecuada y que el modelo funcione sin problemas en el entorno de producción.
Pruebas de rendimiento: Evaluar el rendimiento del modelo bajo diferentes cargas de trabajo y escenarios para identificar posibles problemas y asegurar que responda eficientemente.

Procesos de monitoreo del desempeño en producción:
Supervisión de las predicciones: Monitorear continuamente las predicciones del modelo comparándolas con los precios reales, observando si la desviación de 53 dólares sigue siendo aceptable en el contexto operativo.
Actualización periódica: Considerar actualizaciones periódicas del modelo para adaptarse a cambios en el mercado o en los datos, con el objetivo de mejorar la precisión de las predicciones y reducir el RMSE.
Retroalimentación de los usuarios: Recopilar retroalimentación de usuarios y partes interesadas, teniendo en cuenta sus percepciones sobre la precisión de las predicciones y cualquier área de mejora identificada en la experiencia de uso del modelo.

8. Conclusiones

1. Los modelos predictivos son una alternativa de solución para conocer la información que nos será útil en contextos de mercado, como poder saber el precio de una propiedad para ponerla en servicio de Airbnb en base a sus características.
2. El RMSE nos indica que el valor no es muy bueno, sin embargo en el contexto se podría usar como una aproximación de precio, más no el precio exacto de una propiedad.
3. El rendimiento de los modelos no parece muy sensible a los ajustes de hiperparámetos, aunque se puede evidenciar que siempre mejoran con los hiperparámetros adecuados.
4. La documentación detallada de los ajustes de hiperparámetros y las decisiones de modelado permite la reproducibilidad y facilita futuras iteraciones del trabajo. Esto es fundamental para mantener y mejorar la calidad del modelo a lo largo del tiempo.

Bibliografía

- georgvelev. (2023). ADAMS SoSe23. Kaggle. https://kaggle.com/competitions/adams-sose23