

UdeCataluña

Cree una copia de este Notebook en su Drive y desarrolle el taller sobre ese documento copia. Para crear la copia debe dar click en el botón **Copiar en Drive** ubicado en el panel superior, se abrirá un nuevo notebook que podrá manipular a su antojo y que se almacenará en su Drive dentro de la Carpeta *Colab Notebooks*.

▼ Predicción del precio de hospedajes en Airbnb para la ciudad de Nueva York

Contexto: Airbnb es una empresa que ofrece una plataforma de software dedicada a la oferta de alojamientos particulares y turísticos mediante la cual los anfitriones pueden publicitar y contratar el arriendo de sus propiedades con sus huéspedes; anfitriones y huéspedes pueden valorarse mutuamente, como referencia para futuros usuarios. Muchos nuevos anfitriones no cuentan con información global de tendencias del mercado por lo que sus precios no son óptimos. Airbnb gana una comisión por cada arrendamiento, por lo tanto, está interesado en que sus anfitriones cobren una tarifa óptima de acuerdo a las características del hospedaje. Si los anfitriones ganan más... Airbnb también.

Problema de Negocio: La empresa Airbnb lo ha contratado para desarrollar un modelo que permita responder la siguiente pregunta: ¿Cuál es la variable o característica más relevante para determinar el precio de un hospedaje en Airbnb? Y además que permita predecir el precio de un hospedaje dadas ciertas características del mismo.

Sistema de información: El conjunto de datos objetivo posee información acerca de 30.000 hospedajes de la plataforma Airbnb en la ciudad de Nueva York. Los datos a usar son datos públicos creados por Inside Airbnb, para más información puede consultar [aquí](#).

▼ Indicaciones para resolver el Taller

El objetivo de este taller es que cree varios modelos que permitan predecir el precio de un hospedaje dadas ciertas características del mismo, además de responder algunas preguntas de negocio. Para ello tendrá que programar, investigar y analizar todos los resultados que vaya obteniendo. Tenga en cuenta las siguientes indicaciones:

- Añada comentarios al código para que documente sus soluciones.
- Coloque su análisis en una celda de Texto luego de cada resultado.
- Para resolver un ejercicio puede usar tantas celdas de Código o Texto como requiera.

Si se le presenta un error de código o duda. Siga los siguientes pasos:

1. Lea y entienda el error, para ello puede buscar en la documentación de la librería o googlearlo
2. Intente resolverlo
3. Comuníquese con el experto temático usando el Foro, recuerde enviar un pantallazo del error y mencionar que Ejercicio está solucionando. **Abstengase de compartir el link de su Notebook en el Foro.**

A continuación, se listan algunos recursos que pueden ser valiosos para su análisis.

- En esta página puede encontrar las gráficas que se pueden construir dependiendo de las variables disponibles, una breve explicación de cada gráfica y código para construir cada visualización. <https://www.data-to-viz.com/>
- Documentación de Pandas. <https://pandas.pydata.org/docs/index.html>
- Documentación de Seaborn. <https://seaborn.pydata.org/index.html>
- Documentación de Scikit-learn. <https://scikit-learn.org/stable/>

▼ Instalar, actualizar y cargar las librerías necesarias

A continuación vamos a cargar las librerías necesarias para el desarrollo de este caso.

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib as mpl
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6 import sklearn
7 %matplotlib inline
8 sns.set()
```

```
1 !python --version
2 print('NumPy', np.__version__)
```

```
3 print('Pandas', pd.__version__)
4 print('Matplotlib', mpl.__version__)
5 print('Seaborn', sns.__version__)
6 print('Scikit-learn', sklearn.__version__)
```

```
Python 3.7.13
NumPy 1.21.6
Pandas 1.3.5
Matplotlib 3.2.2
Seaborn 0.11.2
Scikit-learn 1.0.2
```

Este caso fue creado con las siguientes versiones:

```
Python 3.7.13
NumPy 1.21.6
Pandas 1.3.5
Matplotlib 3.2.2
Seaborn 0.11.2
Scikit-learn 1.0.2
```

▼ Etapas de la Fase de modelamiento

Antes de comenzar, debemos revisar el cumplimiento de algunas condiciones y los pasos a seguir durante esta fase.

Condiciones iniciales:

1. Se hizo una fase de exploración previa en la cuál se encontraron relaciones importantes, se encontraron errores o problemas que debían ser corregidos.
2. Se hizo la limpieza de valores faltantes, atípicos y además la transformación de las variables cualitativas.

Pasos en la fase de modelamiento

3. Se tiene una clara variable objetivo, y se ha definido cual es el problema a resolver.
4. Definir los algoritmos de trabajo.
5. Entrenar modelos, optimizarlos y evaluar su desempeño
6. Extraer valor de los modelos construidos

Análisis de nuestra situación

En las fases previas nos hemos encargado de los dos primeros pasos. Tenemos una clara variable objetivo, el precio por noche de nuestros hospedajes en la ciudad de Nueva York, al tratarse de una variable cuantitativa estaremos resolviendo un **problema de**

regresión. Como modelos preliminares estaremos empleando los algoritmos de Vecinos más Cercanos y Árbol de Decisión. Para la evaluación de modelos emplearemos el error cuadrático medio. En la siguiente fase realizaremos dos modelos más complejos usando Redes Neuronales y Bosque Aleatorio.

A continuación se listan las tareas que realizaremos en esta Fase:

1. Exploración rápida del conjunto de datos
2. Partición de prueba y entrenamiento
3. Entrenamiento y optimización de un modelo usando Vecinos más Cercanos
4. Entrenamiento y optimización de un modelo usando Árboles de Decisión.
5. Aprovechamiento del árbol
6. Comparación de los modelos obtenidos
7. Conclusiones

▼ Cargar el dataset de trabajo

Vamos a cargar el archivo producto de la fase de Limpieza y Transformación. Un conjunto de datos sin datos atípicos, valores faltantes y con las transformaciones necesarias.

```
1 airbnb = pd.read_csv('https://raw.githubusercontent.com/HarryVargas96/U
2 airbnb.head()
```

	price	neighbourhood	latitude	longitude	property_type	bathrooms	bedroom
0	149	18.0	40.64749	-73.97237	1.0	1.0	
1	150	80.0	40.80902	-73.94190	1.0	1.0	
2	190	80.0	40.79685	-73.94872	1.0	1.0	
3	60	18.0	40.65599	-73.97519	12.0	1.0	
4	80	101.0	40.86754	-73.92639	1.0	1.0	



```
1 # Dimensiones del dataframe
2 airbnb.shape
```

```
(30173, 13)
```

```
1 # Resumen de las variables del dataframe
2 airbnb.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30173 entries, 0 to 30172
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   price                                30173 non-null  int64
1   neighbourhood                        30173 non-null  float64
2   latitude                            30173 non-null  float64
3   longitude                           30173 non-null  float64
4   property_type                       30173 non-null  float64
5   bathrooms                           30173 non-null  float64
6   bedrooms                           30173 non-null  int64
7   beds                                30173 non-null  int64
8   host_is_superhost                   30173 non-null  float64
9   parking                             30173 non-null  float64
10  room_type_Entire home/apt           30173 non-null  int64
11  room_type_Private room              30173 non-null  int64
12  room_type_Shared room               30173 non-null  int64
dtypes: float64(7), int64(6)
memory usage: 3.0 MB

```

```

1 # El dataset de trabajo no contiene ningún valor nulo
2
3 airbnb.isnull().sum()

```

```

price                                0
neighbourhood                        0
latitude                            0
longitude                           0
property_type                       0
bathrooms                           0
bedrooms                           0
beds                                0
host_is_superhost                   0
parking                             0
room_type_Entire home/apt           0
room_type_Private room              0
room_type_Shared room               0
dtype: int64

```

▼ Partición del dataset: Subconjuntos de prueba y entrenamiento

▼ Ejercicio 1

1. Divida el dataset en variables predictoras con el nombre `x` y variable objetivo con el nombre `y`.
2. Use la función `train_test_split()` para obtener los subconjuntos de prueba y entrenamiento. La partición de prueba debe ser del 30%, emplee `random_state = 0`.

► Pistas

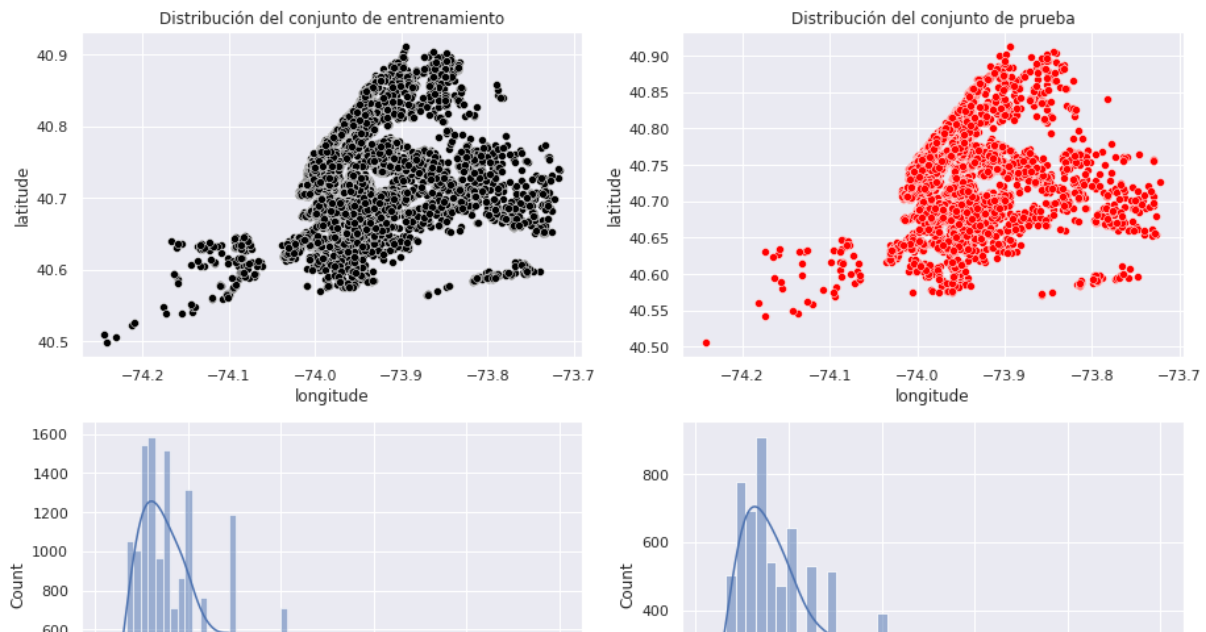
Observemos cómo están distribuidas las muestras en el conjunto de prueba y entrenamiento.

```
1 # Respuesta
2 #dividir el dataset airbnb en X y y siendo y el campo de price
3 X = airbnb.drop(['price'], axis=1)
4 y = airbnb[['price']]
5 #ahora se divide en entrenamiento y test
6
7
8 from sklearn.model_selection import train_test_split
9 X_train, X_test, y_train, y_test = train_test_split(X, y,
10                                                    test_size=0.3,
11                                                    random_state=0)
```

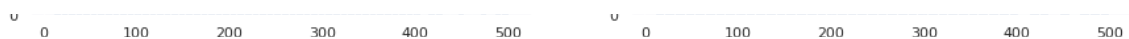
```
1 # No modifique esta celda
2 # Esta celda le permite graficar los resultados del Ejercicio anterior
3
4 fig, ax = plt.subplots(2,2,figsize = (15,10))
5
6 plt.suptitle('Comparación de los conjuntos de prueba y entrenamiento',
7
8 ax[0,0].set_title('Distribución del conjunto de entrenamiento')
9 sns.scatterplot(data = X_train, x = 'longitude',y = 'latitude', color =
10
11 ax[0,1].set_title('Distribución del conjunto de prueba')
12 sns.scatterplot(data = X_test, x = 'longitude',y = 'latitude', color =
13
14 sns.histplot( x = y_train, kde = True, ax = ax[1,0])
15 sns.histplot( x = y_test, kde = True, ax = ax[1,1])
16
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe4109b5290>

Comparación de los conjuntos de prueba y entrenamiento



▼ Vecinos más cercanos - KNN



▼ Ejercicio 2

Investigue cuales son las ventajas y desventajas del algoritmo de vecinos más cercanos. No olvide añadir las fuentes consultadas.

Respuesta Ventaja

1. KNN puede manejar problemas de clasificación y, naturalmente, puede manejar problemas de clasificación múltiple, como la clasificación del iris.
2. Sencillo, fácil de entender y muy potente al mismo tiempo Para el reconocimiento de dígitos escritos a mano, la precisión del iris es muy alta.
3. KNN también puede manejar problemas de regresión, es decir, predicción

Desventaja

1. Baja eficiencia, porque cada vez que se tienen que contar la clasificación o regresión, los datos de entrenamiento y los datos de prueba. Si la cantidad de datos es grande, la potencia informática requerida será asombrosa, pero en el aprendizaje automático, el procesamiento de big data también es Algo muy común
2. La dependencia de los datos de entrenamiento es particularmente grande. Aunque todos los algoritmos de aprendizaje automático dependen mucho de los datos, KNN es particularmente serio, porque si nuestro conjunto de datos de entrenamiento, uno o dos datos son incorrectos, simplemente correctos en nuestro Además del valor que debe clasificarse, esto conducirá directamente a la inexactitud de los datos


predichos, y la tolerancia a fallas de los datos de entrenamiento es demasiado pobre

3. El desastre de la dimensionalidad, KNN no es muy bueno para el procesamiento de datos multidimensionales, como se muestra a continuación

Entrenemos ahora nuestro primero modelo usando el regresor de vecinos más cercanos.

[Bibliografía utilizada para responder](#)

```
1 from sklearn.neighbors import KNeighborsRegressor
2
3 knn = KNeighborsRegressor()
4 knn.fit(X_train,y_train)
5
6 y_train
```

	price	
26748	149	
1714	80	
9186	81	
22951	55	
12182	60	
...	...	
22401	35	
17093	41	
27063	20	
8366	80	
17530	75	

21121 rows × 1 columns

Note que no hemos pasado ningún parámetro al algoritmo. Sin embargo, se ha entrenado exitosamente un modelo.

▼ Ejercicio 3


Investigue cuales fueron los parámetros por defecto usados para entrenar el modelo `knn`, preste atención especial al número de vecinos usados y la métrica para medir distancia

que se empleó. ¿Se usó la métrica euclidiana, Manhattan o coseno?

► **Pistas**

Respuesta

```
1 #extraigo los parametros de knn
2 #extraer los parametros del modelo y los convierto en dataframe
3 parametros=knn.get_params()
4 #se debe transformar el diccionario en dataframe
5 parametros=pd.DataFrame.from_dict(parametros, orient='index')
6 parametros
7
```

	0 
algorithm	auto
leaf_size	30
metric	minkowski
metric_params	None
n_jobs	None
n_neighbors	5
p	2
weights	uniform

▼ **Ejercicio 4**

Evalué el desempeño del modelo knn con el conjunto de prueba. Calcule el error cuadrático medio (MSE) y el coeficiente de determinación (R^2). ¿Qué opina acerca del desempeño de este primer modelo?

► **Pistas**

```
1 #calcular el coeficientes de determinacion de los modelos
2 y_predict = knn.predict(X_test)
3 y_pred2 = knn.predict(X_train)

1 # Evaluación del desempeño del modelo con R^2
2
3 from sklearn.metrics import r2_score
4
5 r2_prueba = r2_score(y_test, y_predict)
6 print('Exactitud en partición de prueba: {:.3f}'.format(r2_prueba))
```

```

7
8
9 r2_entren = r2_score(y_train, y_pred2)
10 print('Exactitud en partición de entrenamiento: {:.3f}'.format(r2_entren))
11
12 # Para modelos de clasificación score es el coeficiente de determinación
13
14 print('\nOtra forma de calcular el R2: {}'.format(knn.score(X_test, y_test)))

Exactitud en partición de prueba: 0.639595
Exactitud en partición de entrenamiento: 0.641688

Otra forma de calcular el R2: 0.6395945246584944


1 #calcular el error cuadrado medio
2 from sklearn.metrics import mean_squared_error
3 mse = mean_squared_error(y_test, y_predict)
4 mse

3145.752041537781

```

▼ Experimentación

Vamos a modificar la complejidad del modelo con el fin de optimizar la salida. La complejidad puede ser modificada cambiando el parámetro K, número de vecinos. Vamos a realizar experimentos para determinar cuál es el parámetro que minimiza el error de predicción en el conjunto de prueba.

▼ Ejercicio 5

Complete el siguiente código con el fin de encontrar el valor de error cuadrático medio para 40 modelos con valores impares de K entre 1 y 80.

► Pistas

```

1 # Respuesta
2
3 # Lista vacía para almacenar los errores
4 mse_knn = []
5 # Lista con los valores de K, valores impares de 1 a 80
6 valores_vecinos = range(1,80,2)
7
8 for i in range(1,80,2):
9     knn = KNeighborsRegressor(n_neighbors=i)
10    knn.fit(X_train,y_train)
11    y_pred = knn.predict(X_test)

```

```

12     y_pred2 = knn.predict(X_train)
13     error=mse_knn.append(mean_squared_error(y_test, y_pred))
14

1 #transformar error en dataframe
2 mse_knn = pd.DataFrame(mse_knn)
3 mse_knn.columns = ['mse']
4 mse_knn.index = valores_vecinos
5
6 #ordenamos de menor error a mayor y solo mostramos los 5 primeros
7 mse_knn = mse_knn.sort_values(by='mse')
8
9 print(mse_knn.head(3))

```

```

      mse
1  3960.036328
3  3972.884649
5  3987.898164

```

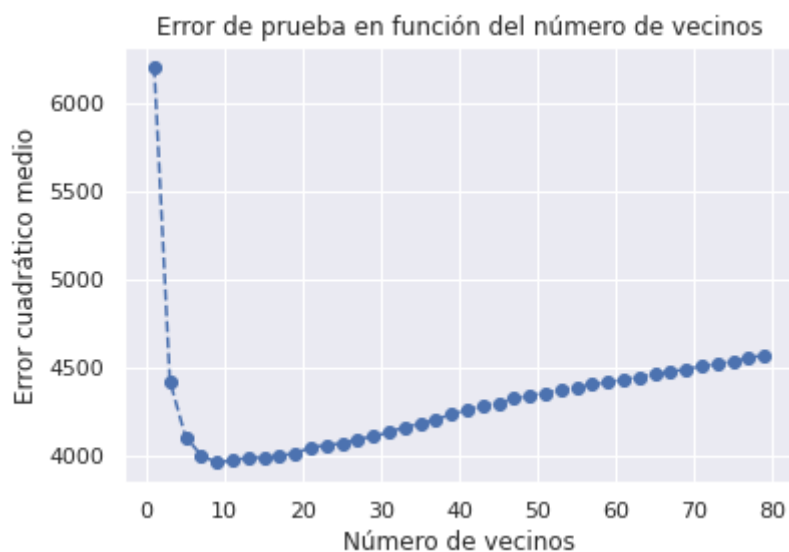
1

```

1 # No modifique esta celda
2 # Esta celda le permite graficar los resultados del Ejercicio anterior
3
4 plt.plot(valores_vecinos, mse_knn,'o',linestyle='dashed')
5
6 plt.title('Error de prueba en función del número de vecinos')
7 plt.xlabel('Número de vecinos')
8 plt.ylabel('Error cuadrático medio')

```

```
Text(0, 0.5, 'Error cuadrático medio')
```



▼ Ejercicio 6

Basado en los experimentos realizados en el ejercicio 4, ¿cuál es el número de vecinos óptimo para el modelo de regresión?

Respuesta En el dataframe que se creo mse_knn se determina que el índice que hace menor el error mse es el de indice 1 y mientras mas valores el error se incrementa

▼ Ejercicio 7

Entrene un modelo con el nombre `knn_final` con el número de vecinos óptimo que encontró en el Ejercicio 5.

Evalúe el desempeño del modelo `knn_final` con las siguientes métricas, recuerde hacer el cálculo con el conjunto de prueba.

- Error cuadrático medio (MSE)
- Raíz del error cuadrático medio (RMSE)
- Error absoluto medio (MAE)
- Coeficiente de determinación (R^2)

Interprete los coeficientes obtenidos, ¿cree que el modelo obtenido cumple con los requisitos del negocio?

```
1 # Respuesta
2
3
4 #calcular el root mean squared error
5 from sklearn.metrics import mean_squared_error
6 mse=mean_squared_error(y_test, y_predict)
7 rmse = mean_squared_error(y_test, y_predict)**0.5
8
9 #calcular el error absolute error
10 from sklearn.metrics import mean_absolute_error
11 mae = mean_absolute_error(y_test, y_predict)
12
13 #calcular coeficiente r2 para el modelo
14 from sklearn.metrics import r2_score
15 r2 = r2_score(y_test, y_predict)
16
17
18 print('El MSE es:' ,mse )
19 print('Raíz del error cuadrático medio (RMSE):' ,rmse )
20 print('Error absoluto medio (MAE):' ,mae )
21 print('Coeficiente de determinación R2:' ,r2 )
22
23
```

El MSE es: 3145.752041537781
Raíz del error cuadrático medio (RMSE): 56.08700421254269
Error absoluto medio (MAE): 37.4214538223597
Coeficiente de determinación R2: 0.6395945246584944

▼ Árboles de Decisión

▼ Ejercicio 8

Investigue cuales son las ventajas y desventajas del algoritmo árbol de decisión. No añadir colocar las fuentes consultadas.

Respuesta **Algunas ventajas de los árboles de decisión son:** Son simples de entender y de interpretar

- Si el árbol no es excesivamente grande, puede visualizarse No requiere una preparación de los datos demasiado exigente (aunque la implementación de Scikit-Learn no soporta valores nulos)
- Se puede trabajar tanto con variables cuantitativas como cualitativas Utiliza un modelo de caja blanca: la respuesta del algoritmo es fácilmente justificable a partir de la lógica booleana implementada en él

Por otro lado, entre las desventajas podemos destacar las siguientes: Los aprendices de árbol de decisión tienden al sobreentrenamiento,

- Son inestables: cualquier pequeño cambio en los datos de entrada puede suponer un árbol de decisión completamente diferente No se puede garantizar que el árbol generado sea el óptimo Hay conceptos que no son fácilmente aprendibles pues los árboles de decisión no son capaces de expresarlos con facilidad (como el operador XOR)
- Los aprendices crean árboles sesgados si hay clases dominantes, por lo que se recomienda balancear el conjunto de datos antes de entrenar el aprendiz

[Referencia de consulta](#)

Entrenemos ahora nuestro primero modelo usando el regresor de vecinos más cercanos.

```
1 from sklearn.tree import DecisionTreeRegressor
2
3 tree = DecisionTreeRegressor(random_state= 5)
4 tree.fit(X_train,y_train)
5
```

```
DecisionTreeRegressor(random_state=5)
```

Ya hemos entrenado un modelo preliminar usando un árbol de decisión sin control de hiperparámetros.

▼ Ejercicio 9

Investigue y responda las siguientes preguntas.

¿Cuál fue el parámetro por defecto empleado para limitar la profundidad el árbol?

¿Cuál fue la profundidad del árbol?

¿Qué opina de la complejidad del modelo obtenido?

```
1 #Se determina cuales son todos los parametros del árbol
2 tgp=tree.get_params()
3 tgp=pd.DataFrame(tgp,index=['parametros'])
4 tgp
```

	ccp_alpha	criterion	max_depth	max_features	max_leaf_nodes	m
parametros	0.0	squared_error	None	None	None	



▼ RESPUESTA

- ¿Cuál fue el parámetro por defecto empleado para limitar la profundidad el árbol?
R: max_depth
- ¿Cuál fue la profundidad del árbol? **R: 39**
- ¿Qué opina de la complejidad del modelo obtenido? **R: No se puede determinar una complejidad alta ya que el modelo utiliza sus parámetros por default**

► Pistas

▼ Ejercicio 10

Evalué el desempeño del modelo `tree` con el conjunto de prueba. Calcule el error cuadrático medio (MSE) y el coeficiente de determinación (R^2). ¿Qué opina del desempeño de este modelo?

```
1 # Respuesta
```

```
Resultados del modelo tree
```

El coeficiente de determinación es: 0.29023252254017684

El error cuadrático medio es: 6195.112571807335

- Se pudo determinar que el modelo DecisionTreeRegressor no se ajusta de buena manera al conjunto de los datos ya que tanto el coeficiente de determinación R^2 no refleja que las variables independientes afecten a el precio final del hospedaje.
- En este caso sería importante determinar cuáles son las variables que influyen directamente en el comportamiento del modelo.

▼ Experimentación

Vamos a modificar la complejidad del modelo basado en árbol de decisión con el fin de optimizar la salida. La complejidad puede ser modificada cambiando el parámetro de profundidad máxima, la pureza de las hojas o el ccp_alpha. Vamos a realizar experimentos para determinar cuál es el parámetro de profundidad que minimiza el error de predicción en el conjunto de prueba.

▼ Ejercicio 11

Complete el siguiente código con el fin de encontrar el valor de error cuadrático medio para 40 modelos con diferente profundidad máxima que va desde 1 hasta 40.

► Pistas

```
1 from sklearn.metrics import mean_squared_error
2 # Lista vacía para almacenar los errores
3 mse_tree = []
4 # Lista con los valores de profundidad, valores de 1 a 40
5 valores_profundidad = range(1,40,1)
6
7 for profundidad in valores_profundidad:
8     tree = DecisionTreeRegressor(max_depth=profundidad, random_state=5)
9     tree.fit(X_train,y_train)
10    y_pred = tree.predict(X_test)
11    #calcular el mse_tree
12    error = mean_squared_error(y_test, y_pred)
13    mse_tree.append(error)
14
15
16 #transformar la lista en un dataframe
17 mse_tree = pd.DataFrame(mse_tree, valores_profundidad, columns=['MSE'])
18
19 #ordenar el dataframe de menor a mayor de los mse
20 mse_tree = mse_tree.sort_values(by='MSE')
```

```

21
22 #imprimir el dataframe head(3)
23 mse_tree.head(3)
24

```

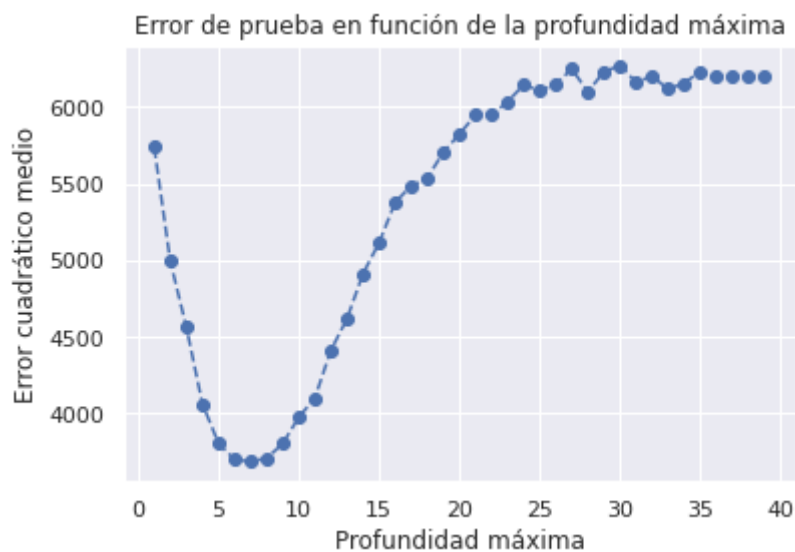
	MSE	
7	3687.158574	
6	3701.821355	
8	3709.578587	

```

1 # No modifique esta celda
2 # Esta celda le permite graficar los resultados del Ejercicio anterior
3
4 plt.plot(valores_profundidad, mse_tree, 'o', linestyle='dashed')
5
6 plt.title('Error de prueba en función de la profundidad máxima')
7 plt.xlabel('Profundidad máxima')
8 plt.ylabel('Error cuadrático medio')

```

Text(0, 0.5, 'Error cuadrático medio')



▼ Ejercicio 12

Basado en los experimentos realizados en el ejercicio 11, ¿cuál es la profundidad óptima para el modelo de regresión?

```

1 #imprimir el dataframe head(1)
2 mse_tree.head(1)

```

	MSE	
7	3687.158574	

- Para determinar el óptimo valor de profundidad se determina en base al primer registro ordenado del dataframe que el mejor valor es el de profundidad 7

Respuesta

▼ Ejercicio 13

Entrene un modelo con el nombre `tree_final` con la profundidad óptima que encontró en el Ejercicio 11.

Evalúe el desempeño del modelo `tree_final` con las siguientes métricas, recuerde hacer el cálculo con el conjunto de prueba.

- Error cuadrático medio (MSE)
- Raíz del error cuadrático medio (RMSE)
- Error absoluto medio (MAE)
- Coeficiente de determinación (R^2)

Interprete los coeficientes obtenidos, ¿cree que el modelo obtenido cumple con los requisitos del negocio?

```

1 # Respuesta
2 from sklearn.metrics import mean_squared_error
3 # Lista vacía para almacenar los errores
4 mse_tree = []
5 # Lista con los valores de profundidad, valores de 1 a 40
6 valores_profundidad = range(7,7,1)
7
8
9 tree_final = DecisionTreeRegressor(max_depth=7, random_state=5)
10 tree_final .fit(X_train,y_train)
11 y_pred = tree.predict(X_test)
12 #calcular el mse_tree
13 error = mean_squared_error(y_test, y_pred)
14 print(error)
15
16
17 y_predict = tree_final .predict(X_test)
18 y_pred2 = tree_final .predict(X_train)
19
20
21 #calcular el root mean squared error
22 from sklearn.metrics import mean_squared_error
23 mse=mean_squared_error(y_test, y_predict)
24 rmse = mean_squared_error(y_test, y_predict)**0.5
25

```

```

26 #calcular el error absolute error
27 from sklearn.metrics import mean_absolute_error
28 mae = mean_absolute_error(y_test, y_predict)
29
30 #calcular coeficiente r2 para el modelo
31 from sklearn.metrics import r2_score
32 r2 = r2_score(y_test, y_predict)
33
34
35 print('El MSE es:' ,mse )
36 print('Raíz del error cuadrático medio (RMSE):' ,rmse )
37 print('Error absoluto medio (MAE):' ,mae )
38 print('Coeficiente de determinación R2:' ,r2 )
39
40

```

```

3687.1585740937485
El MSE es: 3687.1585740937485
Raíz del error cuadrático medio (RMSE): 60.721977685956084
Error absoluto medio (MAE): 40.82541142369468
Coeficiente de determinación R2: 0.5775661523830231

```

▼ Extracción de valor

▼ Ejercicio 14

Los árboles de decisión se caracterizan por su alta interpretabilidad, cree una lista llamada `importancia` con la importancia de las características del modelo `tree_final`. Analice la visualización obtenida, de acuerdo con el modelo entrenado. ¿Cuáles son las variables más importantes a la hora de predecir el precio de un hospedaje?

```

1 # Respuesta
2 #determinar cuales son las variables mas importantes del modelo
3 importances = tree.feature_importances_
4 indices = pd.DataFrame(X_train.columns)
5
6 print('Las variables mas importantes son:', indices[importances>0.01])

```

```

Las variables mas importantes son: 0
1          latitude
2          longitude
3      property_type
4          bathrooms
5          bedrooms
6              beds
9  room_type_Entire home/apt

```

```

1 # No modifique esta celda
2 # Esta celda le permite graficar los resultados del Ejercicio anterior

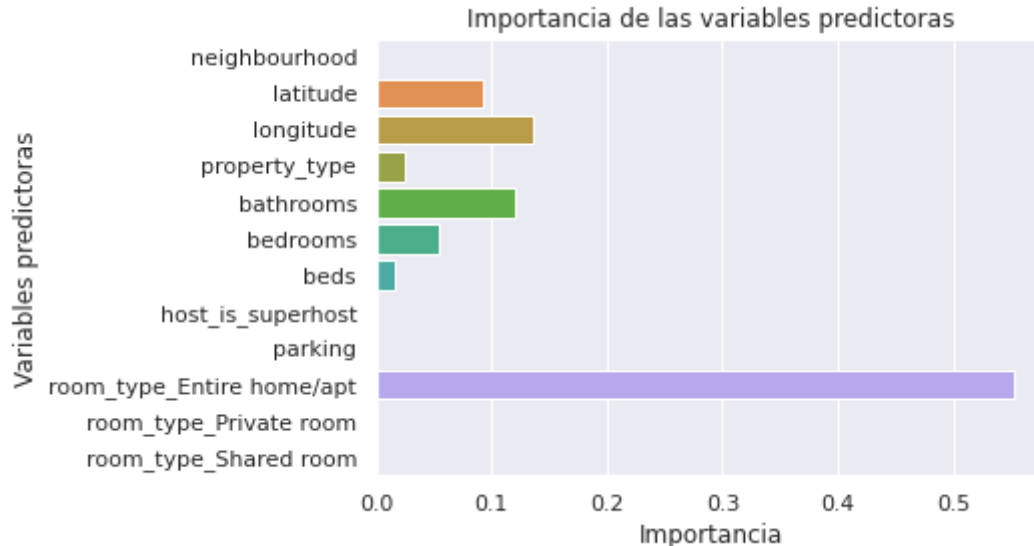
```

```

3
4 sns.barplot(x = importancia,y = list(X_train.columns))
5
6 plt.title('Importancia de las variables predictoras')
7 plt.xlabel('Importancia')
8 plt.ylabel('Variables predictoras')

```

```
Text(0, 0.5, 'Variables predictoras')
```



▼ Ejercicio 15 (Opcional)

Dibuje el árbol de decisión `tree_final`.

Hay muchas formas de dibujar árboles de decisión, en el Notebook M5 - Introducción al problema de clasificación se ilustra una manera. Aquí les dejo algunos artículos que muestran otras formas.

- <https://towardsdatascience.com/visualizing-decision-trees-with-python-scikit-learn-graphviz-matplotlib-1c50b4aa68dc>
- <https://towardsdatascience.com/interactive-visualization-of-decision-trees-with-jupyter-widgets-ca15dd312084>

[Aquí](#) les dejo el resultado de una de esas opciones. Si encuentran otras formas por favor no duden en compartirlas en el Foro.

```
1 !pip install pydotplus
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-w
Requirement already satisfied: pydotplus in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyparsing>=2.0.1 in /usr/local/lib/python3.7/dis
```



```
1 # Respuesta
```

```
2 #haga un grafico del arbol de decision que se vea mas claro
```

```
3 import matplotlib.pyplot as plt
```

```

4 import six
5 import sys
6 sys.modules['sklearn.externals.six'] = six
7 from sklearn.tree import export_graphviz
8 from six import StringIO
9 import pydotplus
10 dot_data = StringIO()
11 export_graphviz(tree, out_file=dot_data,
12                 filled=True, rounded=True,
13                 special_characters=True)
14 graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
15 graph.write_png('tree.png')

True

```

▼ Conclusiones

▼ Ejercicio 16

Para finalizar, compare los modelos obtenidos. A continuación se dejan algunas preguntas guía para que enfoque sus conclusiones.

¿Cuál es el mejor modelo? ¿Cuál modelo utilizaría en producción? Recuerde que las métricas no lo son todo, considere también la interpretabilidad del modelo y el tiempo de entrenamiento. Justifique su respuesta. ¿Qué hallazgos encontró con la fase de modelamiento? ¿Qué recomendaciones haría para mejorar el desempeño de los modelos? ¿Los modelos creados cumplieron con los objetivos demarcados? ¿qué falta para que su modelo tenga el desempeño requerido? ¿qué otro algoritmo podría emplear para lograr mejores resultados?

Métrica	Vecinos más cercanos	Árbol de Decisión
Error cuadrático medio	3145.75	3687.16
Error absoluto medio	56.09	60.72
Coefficiente de determinación	0.64	0.58

Aquí termina la tercera fase del proyecto, ya hemos entrenado y optimizado dos modelos, en la siguiente y última fase se van a presentar 2 algoritmos más avanzados y algunos métodos optimizados para la búsqueda de hiperparámetros.

- ¿Cuál es el mejor modelo? *Entre los dos modelos particularmente por la facilidad de interpretación mi opción es el manejo de árbol de decisiones, ya que a pesar de que la validación del mismo da valores más altos de los errores manejados RSME y Coeficiente de Determinación, pero los valores están muy cercanos al otro modelo.*

- ¿Cuál modelo utilizaría en producción? *Por lo manifestado en la pregunta anterior mi decisión sería utilizar árboles de decisión.*
- ¿Qué hallazgos encontró con la fase de modelamiento? *En mi caso yo baje los datos a mi equipo y probe eliminando algunas variables como latitud, longitud entre otras e intenté ver como el modelo mejoraba pero me di cuenta que el RSME y R2 variaban mucho.*
- ¿Qué recomendaciones haría para mejorar el desempeño de los modelos? *Se debe tratar de determinar la importancia de cada una de las columnas que intervienen en el modelo y se podría utilizar mecanismos de reducción de variables para obtener un modelo más robusto.*
- ¿Los modelos creados cumplieron con los objetivos demarcados? *Si cumplieron con los objetivos planteados ya que permitieron conocer los pasos para poder utilizar técnicas de minería de datos como clasificación y regresión en el ejemplo realizado en este notebook.*
- ¿qué falta para que su modelo tenga el desempeño requerido? *Para mejorar el modelo se requeriría de utilización de un mayor número de GPUS de procesamiento en Colab para poder hacer un mayor número de iteracciones con el fin de optimizar de mejor manera cualquier algoritmo.*
- ¿qué otro algoritmo podría emplear para lograr mejores resultados? *Creo yo que el bosque aleatorio o el manejo de multilayer perceptron permitiría mejorar conjuntamente con mayor número de GPUS de procesamiento y búsqueda de un modelo optimizado.*

Créditos

Profesor: Harry Vargas Rodríguez

Corporación Universitaria de Cataluña - Diplomado en Big Data y Data Science

[Colab paid products](#) - [Cancel contracts here](#)

✓ 5s completed at 2:45 PM

