025-assignment

May 24, 2022

Assignment: Predicting Apartment Prices in Mexico City

[1]:
```
import warnings

import wqet_grader

warnings.simplefilter(action="ignore", category=FutureWarning)
wqet_grader.init("Project 2 Assessment")
```

```
<IPython.core.display.HTML object>
```

```
<b>Note:</b> In this project there are graded tasks in both the lesson notebooks and in this as
```

In this assignment, you'll decide which libraries you need to complete the tasks. You can import them in the cell below.

[104]:
```
# Import libraries here

import warnings
from glob import glob

import pandas as pd
import seaborn as sns
import wqet_grader
from category_encoders import OneHotEncoder
from IPython.display import VimeoVideo
from ipywidgets import Dropdown, FloatSlider, IntSlider, interact
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression, Ridge  # noqa F401
from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import make_pipeline
from sklearn.utils.validation import check_is_fitted

warnings.simplefilter(action="ignore", category=FutureWarning)
```

1 Prepare Data

1.1 Import

Task 2.5.1: (8 points) Write a `wrangle` function that takes the name of a CSV file as input and returns a DataFrame. The function should do the following steps:

1. Subset the data in the CSV file and return only apartments in Mexico City (`"Distrito Federal"`) that cost less than $100,000.
2. Remove outliers by trimming the bottom and top 10% of properties in terms of `"surface_covered_in_m2"`.
3. Create separate `"lat"` and `"lon"` columns.
4. Mexico City is divided into 16 boroughs. Create a `"borough"` feature from the `"place_with_parent_names"` column.
5. Drop columns that are more than 50% null values.
6. Drop columns containing low- or high-cardinality categorical values.
7. Drop any columns that would constitute leakage for the target `"price_aprox_usd"`.
8. Drop any columns that would create issues of multicollinearity.

Tip: Don't try to satisfy all the criteria in the first version of your wrangle function. Instead, work iteratively. Start with the first criteria, test it out with one of the Mexico CSV files in the data/ directory, and submit it to the grader for feedback. Then add the next criteria.

[131]:
```python
# Build your `wrangle` function
def wrangle(filepath):
    # Read CSV file
    df = pd.read_csv(filepath)

    # Subset data: Apartments in "Capital Federal", less than 400,000
    mask_ba = df["place_with_parent_names"].str.contains("Distrito Federal")
    mask_apt = df["property_type"] == "apartment"
    mask_price = df["price_aprox_usd"] < 100_000
    df = df[mask_ba & mask_apt & mask_price]

    # Subset data: Remove outliers for "surface_covered_in_m2"
    low, high = df["surface_covered_in_m2"].quantile([0.1, 0.9])
    mask_area = df["surface_covered_in_m2"].between(low, high)
    df = df[mask_area]

    # Split "lat-lon" column
    df[["lat", "lon"]] = df["lat-lon"].str.split(",", expand=True).astype(float)
    df.drop(columns="lat-lon", inplace=True)

    # Get place name
    df["borough"] = df["place_with_parent_names"].str.split("|", expand=True)[1]
    df.drop(columns="place_with_parent_names", inplace=True)

    #Drop features with high null values
```

```
    df.
↪drop(columns=["floor","expenses","rooms","surface_total_in_m2","price_usd_per_m2"],inplace=

     #Drop features with high and low cardinality
    df.
↪drop(columns=["operation","property_type","properati_url","currency"],inplace=True)

    #drop leaky columns
    df.
↪drop(columns=['price','price_aprox_local_currency','price_per_m2'],inplace=True)

    #drop columns with multicolinearity
   # df.drop(columns=["surface_total_in_m2","rooms"],inplace=True)

    return df
```

[132]:
```
files =glob("data/mexico-city-real-estate-*.csv")
files
```

[132]:
```
['data/mexico-city-real-estate-2.csv',
 'data/mexico-city-real-estate-5.csv',
 'data/mexico-city-real-estate-3.csv',
 'data/mexico-city-real-estate-1.csv',
 'data/mexico-city-real-estate-4.csv']
```

[133]:
```
#using list comprehension
frames=[wrangle(file) for  file in files]
frames[0].head(10)
```

[133]:
```
    price_aprox_usd  surface_covered_in_m2        lat        lon  \
0          63223.78                   88.0  19.516777 -99.160149
1          25289.51                   48.0  19.466724 -99.131614
17         89250.90                   90.0  19.383327 -99.152712
19         39887.51                   60.0  19.388280 -99.195529
20         42475.37                   80.0  19.454582 -99.145651
23         36890.91                   70.0  19.307417 -99.125191
27         56514.99                   75.0  19.377169 -99.187643
31         92201.34                   72.0  19.427443 -99.096555
34         40199.78                   69.0  19.402197 -99.101441
38         32237.80                   60.0  19.408758 -99.129474

               borough
0     Gustavo A. Madero
1     Gustavo A. Madero
17        Benito Juárez
19       Álvaro Obregón
20           Cuauhtémoc
```

```
23             Coyoacán
27         Benito Juárez
31  Venustiano Carranza
34            Iztacalco
38           Cuauhtémoc
```

[134]:
```
#concatenating dataframes
df =pd.concat(frames,ignore_index=True)
print(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5473 entries, 0 to 5472
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   price_aprox_usd       5473 non-null   float64
 1   surface_covered_in_m2 5473 non-null   float64
 2   lat                   5149 non-null   float64
 3   lon                   5149 non-null   float64
 4   borough               5473 non-null   object
dtypes: float64(4), object(1)
memory usage: 213.9+ KB
None
```

[134]:
```
   price_aprox_usd  surface_covered_in_m2        lat        lon  \
0         63223.78                   88.0  19.516777 -99.160149
1         25289.51                   48.0  19.466724 -99.131614
2         89250.90                   90.0  19.383327 -99.152712
3         39887.51                   60.0  19.388280 -99.195529
4         42475.37                   80.0  19.454582 -99.145651

             borough
0  Gustavo A. Madero
1  Gustavo A. Madero
2      Benito Juárez
3     Álvaro Obregón
4         Cuauhtémoc
```

[135]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5473 entries, 0 to 5472
Data columns (total 5 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   price_aprox_usd       5473 non-null   float64
 1   surface_covered_in_m2 5473 non-null   float64
```

```
 2   lat                         5149 non-null   float64
 3   lon                         5149 non-null   float64
 4   borough                     5473 non-null   object
dtypes: float64(4), object(1)
memory usage: 213.9+ KB
```

[136]:
```
#checking missing values, drop a column with more than half values missing
df.isnull().sum()/len(df)
```

[136]:
```
price_aprox_usd          0.0000
surface_covered_in_m2    0.0000
lat                      0.0592
lon                      0.0592
borough                  0.0000
dtype: float64
```

[137]:
```
df.select_dtypes("object").head()
```

[137]:
```
             borough
0  Gustavo A. Madero
1  Gustavo A. Madero
2      Benito Juárez
3     Álvaro Obregón
4         Cuauhtémoc
```

[48]:
```
#Finding number of unique values in each column
df.select_dtypes("object").nunique()
```

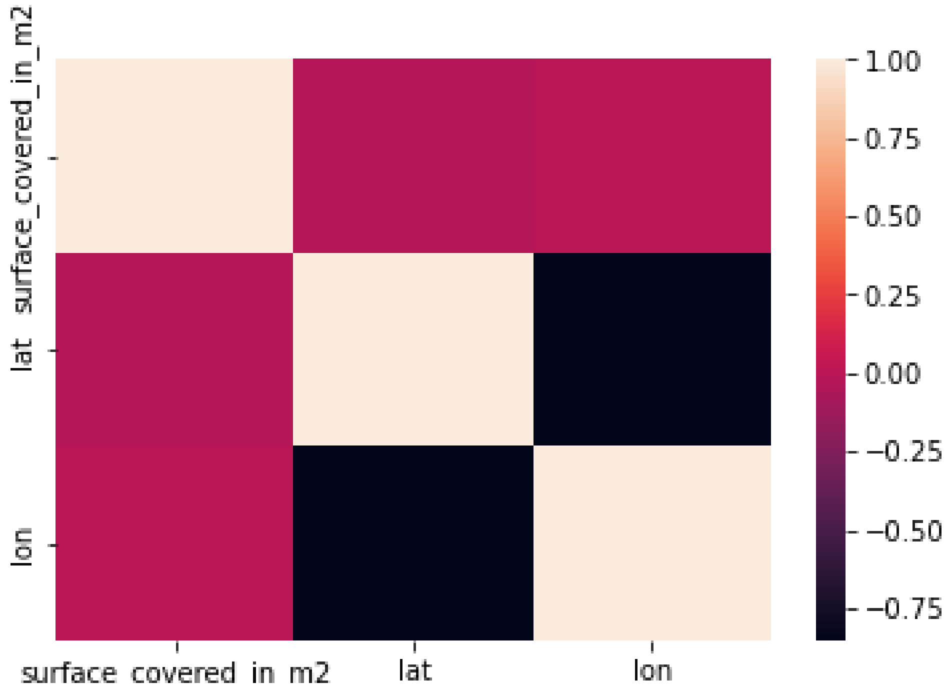
[48]:
```
borough    1
dtype: int64
```

[49]:
```
df.select_dtypes("object").nunique()
```

[49]:
```
borough    1
dtype: int64
```

[57]:
```
corr=df.select_dtypes("number").drop(columns="price_aprox_usd").corr()
sns.heatmap(corr)
```

[57]:
```
<AxesSubplot:>
```



[58]:
```
df.corr()
```

[58]:
```
                       price_aprox_usd  surface_covered_in_m2       lat  \
price_aprox_usd               1.000000               0.276316  0.073034
surface_covered_in_m2         0.276316               1.000000 -0.033695
lat                           0.073034              -0.033695  1.000000
lon                          -0.107600              -0.002994 -0.852599

                            lon
price_aprox_usd       -0.107600
surface_covered_in_m2 -0.002994
lat                   -0.852599
lon                    1.000000
```

[40]:
```
# Use this cell to test your wrangle function and explore the data
```

[138]:
```
wqet_grader.grade(
    "Project 2 Assessment", "Task 2.5.1", wrangle("data/
↪mexico-city-real-estate-1.csv")
)
```

<IPython.core.display.HTML object>

Task 2.5.2: Use glob to create the list `files`. It should contain the filenames of all the Mexico City real estate CSVs in the `./data` directory, except for `mexico-city-test-features.csv`.

[139]:
```
files =glob("data/mexico-city-real-estate-*.csv")
files
```

[139]:
```
['data/mexico-city-real-estate-2.csv',
 'data/mexico-city-real-estate-5.csv',
 'data/mexico-city-real-estate-3.csv',
 'data/mexico-city-real-estate-1.csv',
 'data/mexico-city-real-estate-4.csv']
```

[140]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.2", files)
```

```
<IPython.core.display.HTML object>
```

Task 2.5.3: Combine your `wrangle` function, a list comprehension, and `pd.concat` to create a DataFrame `df`. It should contain all the properties from the five CSVs in `files`.

[ ]:
```
df = ...
print(df.info())
df.head()
```

[141]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.3", df)
```

```
<IPython.core.display.HTML object>
```

1.2 Explore

Task 2.5.4: Create a histogram showing the distribution of apartment prices (`"price_aprox_usd"`) in `df`. Be sure to label the x-axis `"Area [sq meters]"`, the y-axis `"Count"`, and give it the title `"Distribution of Apartment Prices"`.

What does the distribution of price look like? Is the data normal, a little skewed, or very skewed?

[143]:
```
#Import Matplotlib and plotly
import matplotlib.pyplot as plt
import plotly.express as px
```

[144]:
```
# Plot distribution of price
plt.hist(df["price_aprox_usd"])
plt.xlabel("Area [sq meters]")
plt.ylabel("Count")
plt.title("Distribution of Apartment Prices")
# Don't delete the code below
plt.savefig("images/2-5-4.png", dpi=150)
```

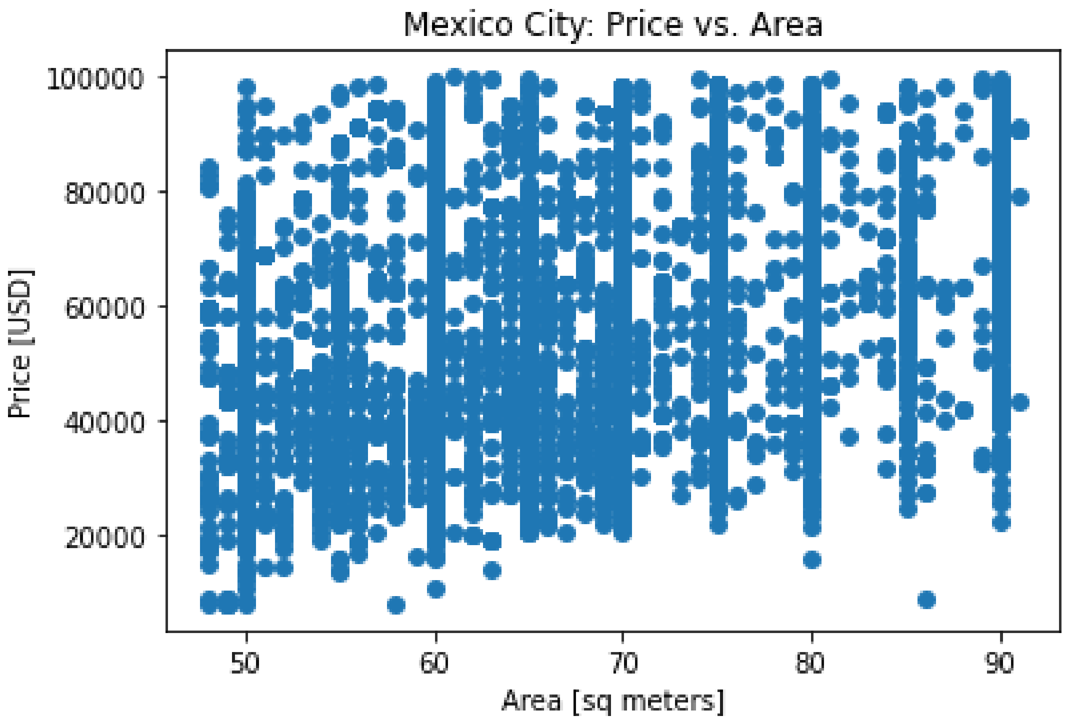
[145]:
```
with open("images/2-5-4.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.4", file)
```

<IPython.core.display.HTML object>

Task 2.5.5: Create a scatter plot that shows apartment price (`"price_aprox_usd"`) as a function of apartment size (`"surface_covered_in_m2"`). Be sure to label your axes `"Price [USD]"` and `"Area [sq meters]"`, respectively. Your plot should have the title `"Mexico City: Price vs. Area"`.

Do you see a relationship between price and area in the data? How is this similar to or different from the Buenos Aires dataset?

[146]:
```
# Plot price vs area
plt.scatter(x=df["surface_covered_in_m2"],y=df["price_aprox_usd"])
plt.xlabel("Area [sq meters]")
plt.ylabel("Price [USD]")
plt.title("Mexico City: Price vs. Area")
# Don't delete the code below
plt.savefig("images/2-5-5.png", dpi=150)
```



[147]:
```
with open("images/2-5-5.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.5", file)
```

```
<IPython.core.display.HTML object>
```

Task 2.5.6: (UNGRADED) Create a Mapbox scatter plot that shows the location of the apartments in your dataset and represent their price using color.

What areas of the city seem to have higher real estate prices?

[ ]:
```
# Plot Mapbox location and price
```

1.3 Split

Task 2.5.7: Create your feature matrix `X_train` and target vector `y_train`. Your target is `"price_aprox_usd"`. Your features should be all the columns that remain in the DataFrame you cleaned above.

[148]:
```
df.columns
```

[148]:
```
Index(['price_aprox_usd', 'surface_covered_in_m2', 'lat', 'lon', 'borough'],
dtype='object')
```

[149]:
```
# Split data into feature matrix `X_train` and target vector `y_train`.
target = "price_aprox_usd"
```

```
X_train =df[["surface_covered_in_m2","lat","lon","borough"]]
y_train=df[target]
```

[150]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.7a", X_train)
```

```
<IPython.core.display.HTML object>
```

[152]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.7b", y_train)
```

```
<IPython.core.display.HTML object>
```

2 Build Model

2.1 Baseline

Task 2.5.8: Calculate the baseline mean absolute error for your model.

[ ]:
```
y_mean=y_train.mean()
y_pred_baseline=[y_mean]*len(y_train)
print("Mean apt price:",round(y_mean,2))

print("Baseline MAE:",mean_absolute_error(y_train,y_pred_baseline))
```

[ ]:
```
y_pred_training =model.predict(X_train)
mae_training =mean_absolute_error(y_train,y_pred_training)
print("Training MAE:", round(mae_training, 2))
```

[173]:
```
y_mean =y_train.mean()
y_pred_baseline =[y_mean]*len(y_train)
baseline_mae =mean_absolute_error(y_train,y_pred_baseline )
print("Mean apt price:", y_mean)
print("Baseline MAE:", baseline_mae )
```

```
Mean apt price: 54246.53149826428
Baseline MAE: 17239.939475888303
```

[174]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.8", [baseline_mae])
```

```
<IPython.core.display.HTML object>
```

2.2 Iterate

Task 2.5.9: Create a pipeline named `model` that contains all the transformers necessary for this dataset and one of the predictors you've used during this project. Then fit your model to the training data.

[175]:
```
#Replacing all missing values with mean
imputer = SimpleImputer()
```

[176]:
```
#instantiate
ohe =OneHotEncoder()
#Fit
ohe.fit(X_train)
#Transform
XT_train =ohe.transform(X_train)
print(XT_train.shape)
XT_train.head()
```

```
(5473, 18)
```

[176]:
```
   surface_covered_in_m2        lat        lon  borough_1  borough_2  \
0                   88.0  19.516777 -99.160149          1          0
1                   48.0  19.466724 -99.131614          1          0
2                   90.0  19.383327 -99.152712          0          1
3                   60.0  19.388280 -99.195529          0          0
4                   80.0  19.454582 -99.145651          0          0

   borough_3  borough_4  borough_5  borough_6  borough_7  borough_8  \
0          0          0          0          0          0          0
1          0          0          0          0          0          0
2          0          0          0          0          0          0
3          1          0          0          0          0          0
4          0          1          0          0          0          0

   borough_9  borough_10  borough_11  borough_12  borough_13  borough_14  \
0          0           0           0           0           0           0
1          0           0           0           0           0           0
2          0           0           0           0           0           0
3          0           0           0           0           0           0
4          0           0           0           0           0           0

   borough_15
0           0
1           0
2           0
3           0
4           0
```

[177]:
```
imputer.fit(XT_train)
```

[177]:
```
SimpleImputer()
```

[178]:
```
# Build Model
model =make_pipeline(  OneHotEncoder(use_cat_names=True),
    SimpleImputer(),
    Ridge()
```

```
)
# Fit model
model.fit(X_train,y_train)
```

```
[178]: Pipeline(steps=[('onehotencoder',
                 OneHotEncoder(cols=['borough'], use_cat_names=True)),
                ('simpleimputer', SimpleImputer()), ('ridge', Ridge())])
```

```
[179]: wqet_grader.grade("Project 2 Assessment", "Task 2.5.9", model)
```

```
<IPython.core.display.HTML object>
```

2.3 Evaluate

Task 2.5.10: Read the CSV file `mexico-city-test-features.csv` into the DataFrame `X_test`.

Tip: Make sure the X_train you used to train your model has the same column order as X_test. Otherwise, it may hurt your model's performance.

```
[181]: X_test =pd.read_csv("data/mexico-city-test-features.csv")
print(X_test.info())
X_test.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1041 entries, 0 to 1040
Data columns (total 4 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   surface_covered_in_m2  1041 non-null   float64
 1   lat                   986 non-null    float64
 2   lon                   986 non-null    float64
 3   borough               1041 non-null   object
dtypes: float64(3), object(1)
memory usage: 32.7+ KB
None
```

```
[181]:    surface_covered_in_m2        lat        lon        borough
0                       60.0  19.493185 -99.205755   Azcapotzalco
1                       55.0  19.307247 -99.166700       Coyoacán
2                       50.0  19.363469 -99.010141     Iztapalapa
3                       60.0  19.474655 -99.189277   Azcapotzalco
4                       74.0  19.394628 -99.143842  Benito Juárez
```

```
[182]: wqet_grader.grade("Project 2 Assessment", "Task 2.5.10", X_test)
```

```
<IPython.core.display.HTML object>
```

Task 2.5.11: Use your model to generate a Series of predictions for `X_test`. When you submit your predictions to the grader, it will calculate the mean absolute error for your model.

[183]:
```
y_test_pred =pd.Series(model.predict(X_test))
y_test_pred.head()
```

[183]:
```
0    53538.366480
1    53171.988369
2    34263.884179
3    53488.425607
4    68738.924884
dtype: float64
```

[184]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.11", y_test_pred)
```

```
<IPython.core.display.HTML object>
```

3 Communicate Results

Task 2.5.12: Create a Series named `feat_imp`. The index should contain the names of all the features your model considers when making predictions; the values should be the coefficient values associated with each feature. The Series should be sorted ascending by absolute value.

[193]:
```
coefficients =model.named_steps["ridge"].coef_
features =model.named_steps["onehotencoder"].get_feature_names()
feat_imp =pd.Series(coefficients,index=features)
feat_imp
```

[193]:
```
borough_Azcapotzalco                 291.654156
borough_Benito Juárez                478.901375
borough_Coyoacán                   -2492.221814
borough_Cuajimalpa de Morelos      -6637.429757
borough_Cuauhtémoc                 13778.188880
borough_Gustavo A. Madero           3275.121061
borough_Iztacalco                   -350.531990
borough_Iztapalapa                  3737.561001
borough_La Magdalena Contreras     -5609.918629
borough_Miguel Hidalgo               405.403127
borough_Tlalpan                    10319.429804
borough_Tláhuac                     2459.288646
borough_Venustiano Carranza       -13349.017448
borough_Xochimilco                  1977.314718
borough_Álvaro Obregón            -14166.869486
lat                                  929.857400
lon                                -5925.666450
surface_covered_in_m2               9157.269123
dtype: float64
```

[ ]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.12", feat_imp)
```

[198]:
```
sorted_coeff=coefficients.sort()
sorted_coeff
```

Task 2.5.13: Create a horizontal bar chart that shows the **10 most influential** coefficients for your model. Be sure to label your x- and y-axis `"Importance [USD]"` and `"Feature"`, respectively, and give your chart the title `"Feature Importances for Apartment Price"`.

[ ]:
```
# Create horizontal bar chart

# Don't delete the code below
plt.savefig("images/2-5-13.png", dpi=150)
```

[ ]:
```
with open("images/2-5-13.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.13", file)
```
