2.5. Predicting Apartment Prices in Mexico City MX

```
import warnings

import wqet_grader

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

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

```
# Import libraries here
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import wqet_grader
import plotly.express as px
import plotly.graph_objects as go
from ipywidgets import Dropdown, FloatSlider, IntSlider, interact
from category_encoders import OneHotEncoder
from sklearn.linear_model import LinearRegression, Ridge  # noqa F401
from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.utils.validation import check_is_fitted
from glob import glob
```

Prepare Data

Import

Task 2.5.1: 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 15 boroughs. Create a `"borough"` feature from the `"place_with_parent_names"` column.

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.

```
# Build your `wrangle` function
def wrangle(csv_file):
    # Read the CSV file into a DataFrame
    df = pd.read_csv(csv_file)

    ab = df["place_with_parent_names"].str.contains("Distrito Federal")
    apt = df["property_type"]=='apartment'
    price = df["price_aprox_usd"] < 100_000
    df = df[ab & apt & price]
    low, high = df["surface_covered_in_m2"].quantile([0.1 , 0.9])
    make_surface_covered_in_m2 = df["surface_covered_in_m2"].between(low, high)
    df = df[make_surface_covered_in_m2]
    df[["lat" , "lon"]] = df["lat-lon"].str.split("," , expand=True).astype(float)
    df.drop(columns="lat-lon", inplace=True)
    df["borough"] = df["place_with_parent_names"].str.split("|" , expand=True)[1]
    df.drop(columns="place_with_parent_names", inplace=True)

    df.drop(columns=["operation" , "currency" , "property_type" , "properati_url"], inplace=True)
    df.drop(columns=["price_aprox_local_currency" , "price_usd_per_m2" , "price_per_m2" , "price"], inplace=True)
    df.drop(columns=["floor" , "expenses"], inplace=True)
    df.drop(columns=["rooms" , "surface_total_in_m2"], inplace=True)

    return df
```

```
# Use this cell to test your wrangle function and explore the data
```

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

Boom! You got it.

Score: 1

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`.

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

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

```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.2", files)
```

You got it. Dance party time! 🕺💃🕺💃

Score: 1

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 = pd.concat([wrangle(file) for file in files] , 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
```

	price_aprox_usd	surface_covered_in_m2	lat	lon	borough
0	82737.39	75.0	19.362690	-99.150565	Benito Juárez
1	72197.60	62.0	19.291345	-99.124312	Tlalpan

1	72197.60	62.0	19.291345	-99.124312	Tlalpan
2	44277.72	85.0	19.354987	-99.061709	Iztapalapa
3	60589.45	52.0	19.469681	-99.086136	Gustavo A. Madero
4	47429.08	53.0	19.443592	-99.121407	Venustiano Carranza

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

Yup. You got it.

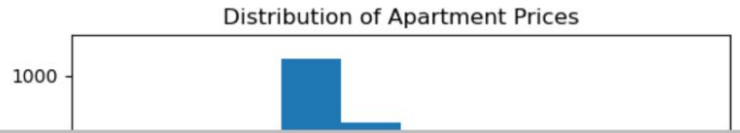
Score: 1

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 `"Price [$]"` , the y-axis `"Count"` , and give it the title `"Distribution of Apartment Prices"` . Use Matplotlib (`plt`).

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

]:
```
# Build histogram
plt.hist(df["price_aprox_usd"])

# Label axes
plt.xlabel("Price [$]")
plt.ylabel("Count")

# Add title
plt.title("Distribution of Apartment Prices")

# Don't delete the code below 👇
plt.savefig("images/2-5-4.png", dpi=150)
```




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

Party time! 🎉 🎉 🎉

Score: 1

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 x-axis `"Area [sq meters]"` and y-axis `"Price [USD]"` . Your plot should have the title `"Mexico City: Price vs. Area"` . Use Matplotlib (`plt`).

```
# Build scatter plot
plt.scatter(df["surface_covered_in_m2"] , df["price_aprox_usd"])

# Label axes
plt.xlabel("Area [sq meters]")
```

```
# Build scatter plot
plt.scatter(df["surface_covered_in_m2"] , df["price_aprox_usd"])
# Label axes
plt.xlabel("Area [sq meters]")
plt.ylabel("Price [USD]")
# Add title
plt.title("Mexico City: Price vs. Area")
# Don't delete the code below 👇
plt.savefig("images/2-5-5.png", dpi=150)
```

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

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

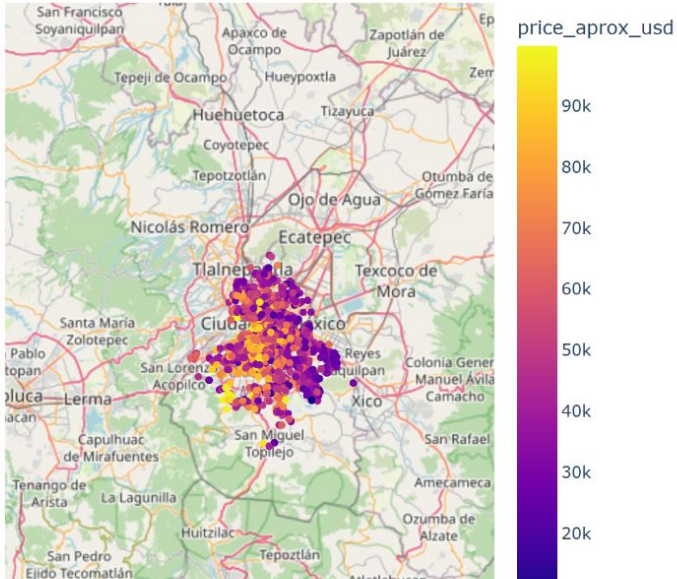
You're making this look easy. 😊

Score: 1

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
fig = px.scatter_mapbox(
    df,
    lat = "lat",
    lon = "lon",
    width = 600,
    height = 600,
    color = "price_aprox_usd",
    hover_data = ["price_aprox_usd"]
)

fig.update_layout(mapbox_style="open-street-map")
fig.show()
```



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.

]:
```
# Split data into feature matrix `X_train` and target vector `y_train`.
X = ["surface_covered_in_m2" , "lat" , "lon" , "borough"]
Y = "price_aprox_usd"
X_train = df[X]
y_train = df[Y]
```

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

Boom! You got it.

Score: 1

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

Very impressive.

Score: 1

Build Model

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)
baseline_mae = mean_absolute_error(y_train, y_pred_baseline)
print("Mean apt price:", y_mean)
print("Baseline MAE:", baseline_mae)
```

]:
```
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: 54246.5314982642
Baseline MAE: 17239.939475888295
```

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

Boom! You got it.

Score: 1

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.

]:
```
# Build Model
model = make_pipeline(
    OneHotEncoder(use_cat_names=True),
    SimpleImputer(),
    Ridge()
)
# Fit model
model.fit(X_train , y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Score: 1

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.

```
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
```

	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

```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.10", X_test)
```

Excellent! Keep going.

Score: 1

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.

```
y_test_pred = pd.Series(model.predict(X_test))
y_test_pred.head()
```

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

```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.11", y_test_pred)
```

Your model's mean absolute error is `14901.618`. That's the right answer. Keep it up!

Score: 1

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.

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

```
surface_covered_in_m2              291.654156
lat                                478.901375
lon                              -2492.221814
borough_Benito Juárez            13778.188880
borough_Tlalpan                  10319.429804
borough_Iztapalapa              -13349.017448
borough_Gustavo A. Madero        -6637.429757
borough_Venustiano Carranza      -5609.918629
borough_Iztacalco                  405.403127
borough_Coyoacán                  3737.561001
borough_Cuauhtémoc                -350.531990
borough_Miguel Hidalgo            1977.314718
```

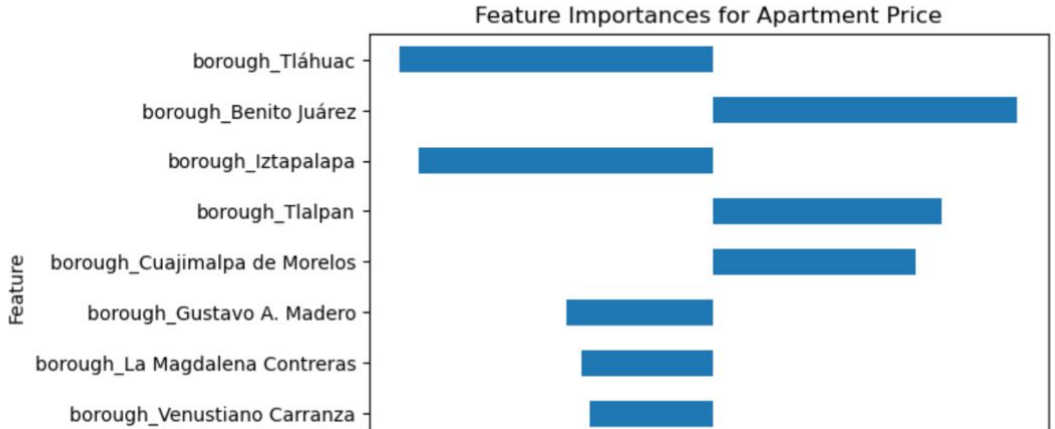
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.12", feat_imp)
```

Party time! 🎉 🎉 🎉

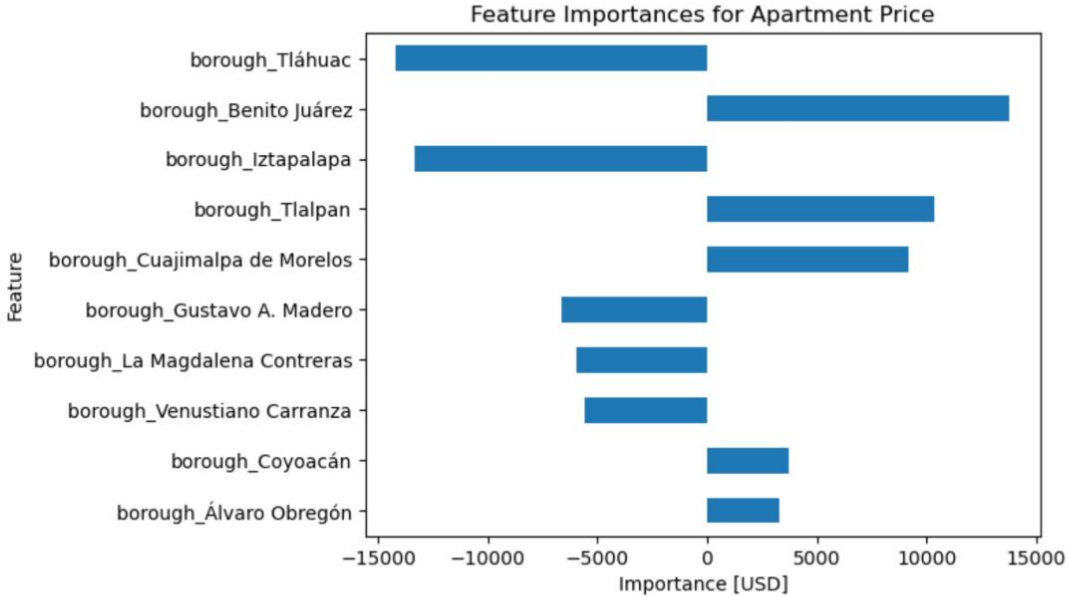
Score: 1

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"`. Use pandas.

```
# Build bar chart
feat_imp.sort_values(key=abs).tail(10).plot(kind="barh")


# Label axes
plt.xlabel("Importance [USD]")
plt.ylabel("Feature")

# Add title
plt.title("Feature Importances for Apartment Price")

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



```
# Add title
plt.title("Feature Importances for Apartment Price")

# 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)
```

You got it. Dance party time! 🕺 💃 🕺 💃

Score: 1