2.5. Predicting Apartment Prices in Mexico City MX

In [1]:
```
import warnings

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

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

In [2]:
```
# Import libraries here
from glob import glob
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
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
```

Prepare Data

Import

Task 2.5.1

> **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.

In [3]:
```
# Build your `wrangle` function
def wrangle(file_path):
    df = pd.read_csv(file_path)

    # 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]

    # Trimming the bottom  and top 10% of properties in terms of surface_covered_in
    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 the lat-lon column into 2 and then drop it
    df[["lat", "lon"]] = df["lat-lon"].str.split(",", expand=True).astype(float)
    df.drop(columns="lat-lon", inplace=True)

    # Create a borough feature from the place_with_parent_names column
    df["borough"] = df["place_with_parent_names"].str.split("|", expand=True)[1]
    df.drop(columns="place_with_parent_names", inplace=True)

    #  Drop columns that have more than 50%  null values
    df.drop(columns=["surface_total_in_m2", "price_usd_per_m2", "floor", "rooms", "

    # Drop columns containing low or high cardinality categorical values
    df.drop(columns=[
        'operation',
        'property_type',
        'currency',
        'properati_url'
    ],
        inplace=True)

    # Let's drop Leaky columns
    df.drop(columns=[
        'price',
        'price_aprox_local_currency',
        'price_per_m2'
    ],
        inplace=True)

    return df
```

In [4]:
```
# Use this cell to test your wrangle function on the file `mexico-city-real-estate-
frame1 = wrangle("data/mexico-city-real-estate-1.csv")
```

In [5]:
```
frame1.head()
```

Out[5]:

	price_aprox_usd	surface_covered_in_m2	lat	lon	borough
11	94022.66	57.0	23.634501	-102.552788	Benito Juárez
20	70880.12	56.0	19.402413	-99.095391	Iztacalco
21	68228.99	80.0	19.357820	-99.149406	Benito Juárez
22	24235.78	60.0	19.504985	-99.208557	Azcapotzalco
26	94140.20	50.0	19.354219	-99.126244	Coyoacán

Task 2.5.2

In [6]:
```
files = sorted(glob("data/mexico-city-real-estate-*.csv"))
files
```

Out[6]:
```
['data\\mexico-city-real-estate-1.csv',
 'data\\mexico-city-real-estate-2.csv',
 'data\\mexico-city-real-estate-3.csv',
 'data\\mexico-city-real-estate-4.csv',
 'data\\mexico-city-real-estate-5.csv']
```

Task 2.5.3

In [7]:
```
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
```

Out[7]:

	price_aprox_usd	surface_covered_in_m2	lat	lon	borough
0	94022.66	57.0	23.634501	-102.552788	Benito Juárez
1	70880.12	56.0	19.402413	-99.095391	Iztacalco
2	68228.99	80.0	19.357820	-99.149406	Benito Juárez
3	24235.78	60.0	19.504985	-99.208557	Azcapotzalco
4	94140.20	50.0	19.354219	-99.126244	Coyoacán

Explore

Slight Code Change

In the following task, you'll notice a small change in how plots are created compared to what you saw in the lessons. While the lessons use the global matplotlib method like `plt.plot(...)`, in this task, you are expected to use the object-oriented (OOP) API instead. This means creating your plots using `fig, ax = plt.subplots()` and then calling plotting methods on the `ax` object, such as `ax.plot(...)`, `ax.hist(...)`, or `ax.scatter(...)`.

If you're using pandas' or seaborn's built-in plotting methods (like `df.plot()` or `sns.lineplot()`), make sure to pass the `ax=ax` argument so that the plot is rendered on the correct axes.

This approach is considered best practice and will be used consistently across all graded tasks that involve matplotlib.

Task 2.5.4

In [8]:
```
fig, ax = plt.subplots()

# Plot the histogram on the axes object
ax.hist(df["price_aprox_usd"])

# Label axes using the axes
ax.set_xlabel("Price [$]")
ax.set_ylabel("Count")


# Add title
ax.set_title("Distribution of Apartment Prices")
```

Out[8]:
```
Text(0.5, 1.0, 'Distribution of Apartment Prices')
```

Task 2.5.5

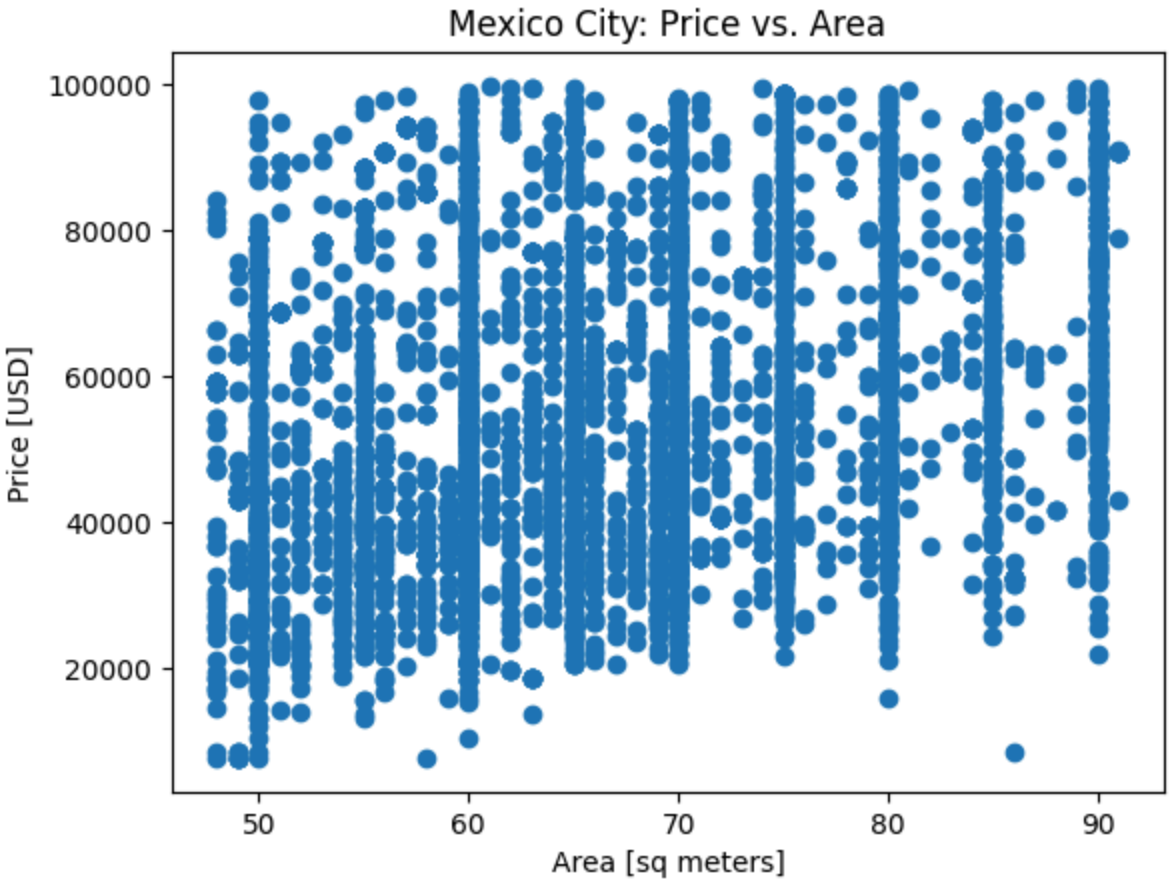
In [9]:
```
fig, ax = plt.subplots()

# Create the scatter plot on the axes object
ax.scatter(
    x = df["surface_covered_in_m2"],
    y = df["price_aprox_usd"]
)

# Label axes
ax.set_xlabel("Area [sq meters]")
ax.set_ylabel("Price [USD]")

#  Add title
ax.set_title("Mexico City: Price vs. Area")
```

Out[9]: Text(0.5, 1.0, '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?

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?

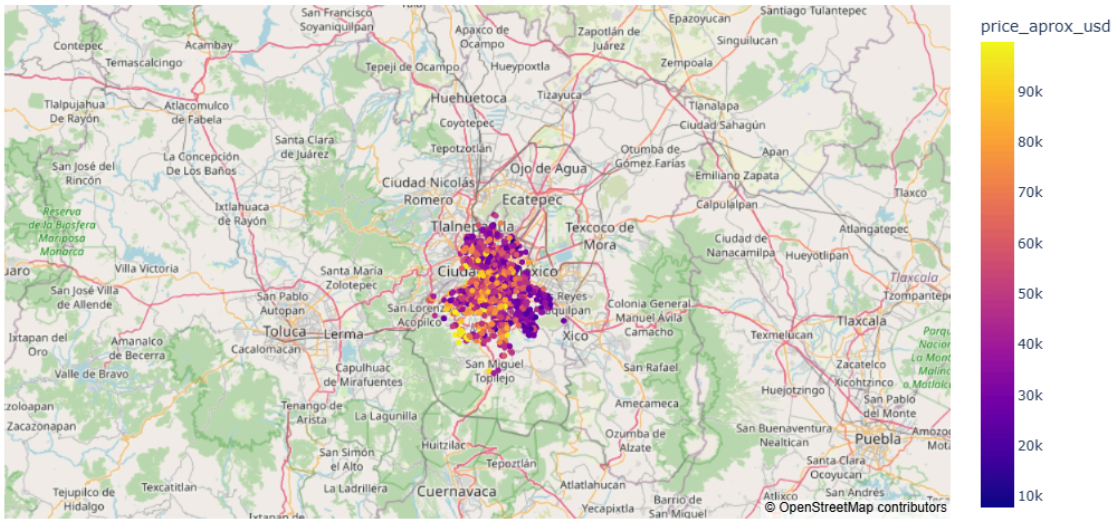
In [10]:
```
# Plot Mapbox location and price
```

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

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

fig.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_14568\2297340783.py:3: DeprecationWarning:
*scatter_mapbox* is deprecated! Use *scatter_map* instead. Learn more at: https://pl
otly.com/python/mapbox-to-maplibre/
  fig = px.scatter_mapbox(
```



Split

Task 2.5.7

In [11]:
```
# Split data into feature matrix `X_train` and target vector `y_train`.
target = "price_aprox_usd"
features = ["surface_covered_in_m2", "lat", "lon", "borough"]
X_train = df[features]
y_train = df[target]
```

Build Model

Baseline

Task 2.5.8

In [13]:
```
y_mean = y_train.mean()
y_pred_baseline = [y_mean] * len(y_train)
baseline_mae = round(mean_absolute_error(y_train, y_pred_baseline), 2)
print("Mean apt price:", y_mean)
print("Baseline MAE:", baseline_mae)
```

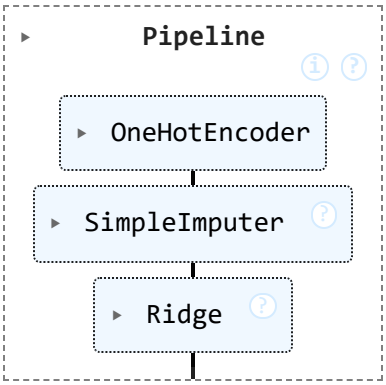
```
Mean apt price: 54246.5314982642
Baseline MAE: 17239.94
```

Iterate

Task 2.5.9

In [14]:
```
# Build Model
model = make_pipeline(
    OneHotEncoder(use_cat_names=True),
    SimpleImputer(),
    Ridge()
)
# Fit model
model.fit(X_train, y_train)
```

Out[14]:



Evaluate

Task 2.5.10

> **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.

In [15]:
```
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
```

Out[15]:

	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

Task 2.5.11

In [16]:
```
y_test_pred = pd.Series(model.predict(X_test))
y_test_pred.head()
```

Out[16]:
```
0    53538.366480
1    53171.988369
2    34263.884179
3    53488.425607
4    68738.924884
dtype: float64
```

Communicate Results

Task 2.5.12

In [17]:
```
coefficients = model.named_steps["ridge"].coef_
features = model.named_steps["onehotencoder"].get_feature_names()
feat_imp = pd.Series(coefficients, features).sort_values(ascending=True, key=abs)
feat_imp
```

Out[17]:
```
surface_covered_in_m2              291.654156
borough_Cuauhtémoc                -350.531990
borough_Iztacalco                  405.403127
lat                                478.901375
borough_Xochimilco                 929.857400
borough_Miguel Hidalgo            1977.314718
borough_Azcapotzalco              2459.288646
lon                              -2492.221814
borough_Álvaro Obregón            3275.121061
borough_Coyoacán                  3737.561001
borough_Venustiano Carranza      -5609.918629
borough_La Magdalena Contreras   -5925.666450
borough_Gustavo A. Madero        -6637.429757
borough_Cuajimalpa de Morelos     9157.269123
borough_Tlalpan                  10319.429804
borough_Iztapalapa              -13349.017448
borough_Benito Juárez            13778.188880
borough_Tláhuac                 -14166.869486
dtype: float64
```

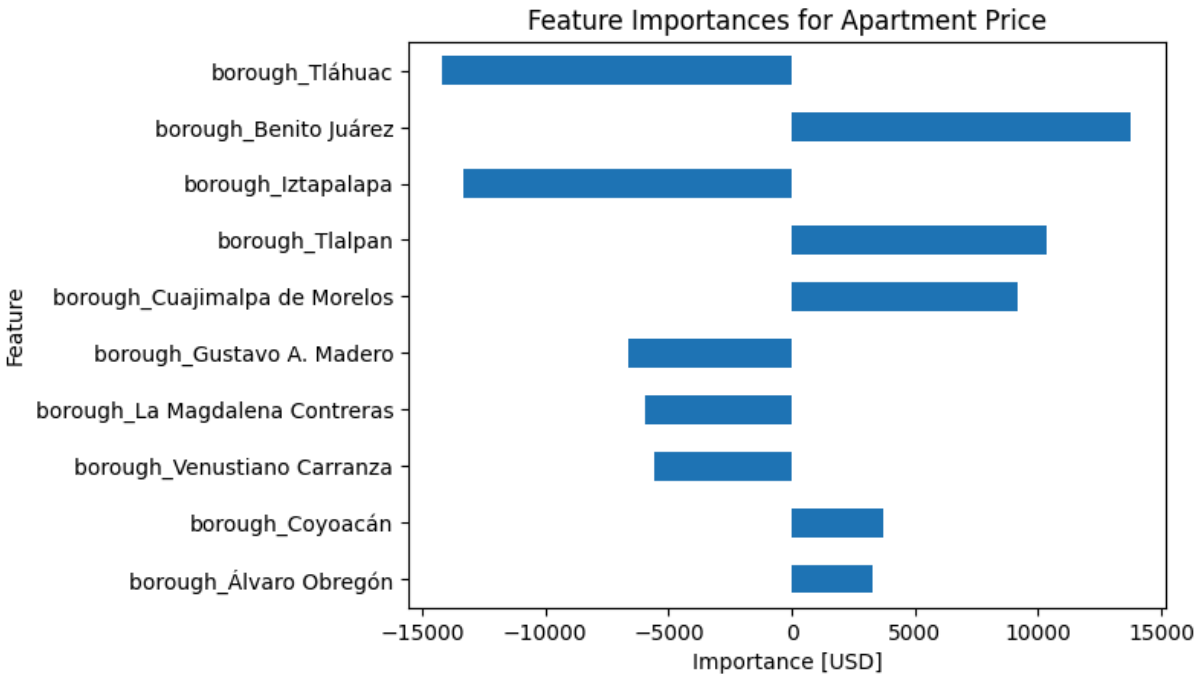
Task 2.5.13

In [18]:
```
fig, ax = plt.subplots()

# Create the horizontal bar plot on the axes object
feat_imp.sort_values(key=abs).tail(10).plot(kind="barh", ax=ax)

#  Label axes
ax.set_xlabel("Importance [USD]")
ax.set_ylabel("Feature")

# Add title
ax.set_title("Feature Importances for Apartment Price");
```



Copyright 2024 WorldQuant University. This content is licensed solely for personal use. Redistribution or publication of this material is strictly prohibited.