025-assignment

April 29, 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.

[48]:
```
# Import libraries here
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from glob import glob
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from category_encoders import OneHotEncoder
```

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.

[33]:
```
# Build your `wrangle` function
def wrangle(filepath):
    df = pd.read_csv(filepath)

    # Subset data: Distrito Federal, apartment, cost $100,000
    mask_df = df['place_with_parent_names'].str.contains('Distrito Federal')
    mask_cost = df['price_aprox_usd'] < 100000
    mask_apt = df['property_type'] == 'apartment'

    df = df[mask_df & mask_cost & mask_apt]

    # Outliers
    low, high = df['surface_covered_in_m2'].quantile([0.10, 0.90])
    mask_area = df['surface_covered_in_m2'].between(low, high)

    df = df[mask_area]

    # Split lat and lon
    df[['lat', 'lon']] = df['lat-lon'].str.split(',', expand = True).
↪astype(float)

    # Boroughs
    df['borough'] = df['place_with_parent_names'].str.split('|', expand =␣
↪True)[1]


    df.drop(columns = ['lat-lon', # Split
                       'place_with_parent_names',  #split
                       'surface_total_in_m2', 'price_usd_per_m2', 'floor',␣
↪'rooms', 'expenses', # More than 50% null values
                       'operation', 'property_type', 'currency', # low␣
↪cardinality
                       'properati_url', # high cardinality
                       'price', 'price_aprox_local_currency', 'price_per_m2',␣
↪'price_usd_per_m2', 'properati_url'  # leakage
                      ], inplace = True)
```

```
    return df
```

[34]:
```
# Use this cell to test your wrangle function and explore the data
df = wrangle('data/mexico-city-real-estate-1.csv')
df.isnull().sum() / len(df)
```

[34]:
```
price_aprox_usd          0.000000
surface_covered_in_m2    0.000000
lat                      0.054496
lon                      0.054496
borough                  0.000000
dtype: float64
```

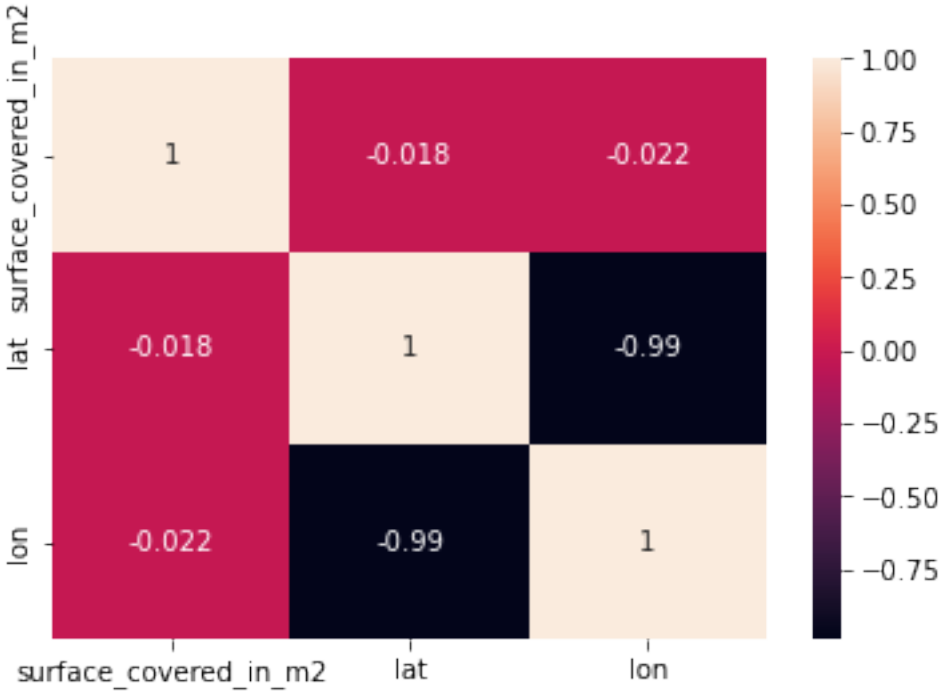
[35]:
```
df.select_dtypes('object').nunique()
```

[35]:
```
borough    14
dtype: int64
```

[36]:
```
sorted(df.columns)
```

[36]:
```
['borough', 'lat', 'lon', 'price_aprox_usd', 'surface_covered_in_m2']
```

[37]:
```
corr = df.select_dtypes('number').drop(columns = 'price_aprox_usd').corr()
sns.heatmap(corr, annot = True);
```



[38]:
```
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`.

[39]:
```
files = sorted(glob('data/mexico-city-real-estate-*.csv'))
files
```

[39]:
```
['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']
```

[40]:
```
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`.

[41]:
```
df = pd.concat([wrangle(file) for file in files])
print(df.info())
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5473 entries, 11 to 4618
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: 256.5+ KB
None
```

[41]:

	price_aprox_usd	surface_covered_in_m2	lat	lon	\
11	94022.66	57.0	23.634501	-102.552788	
20	70880.12	56.0	19.402413	-99.095391	
21	68228.99	80.0	19.357820	-99.149406	
22	24235.78	60.0	19.504985	-99.208557	

```
26         94140.20                  50.0  19.354219  -99.126244

          borough
11  Benito Juárez
20      Iztacalco
21  Benito Juárez
22   Azcapotzalco
26       Coyoacán
```

[42]:
```
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?

[44]:
```
# 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)
```

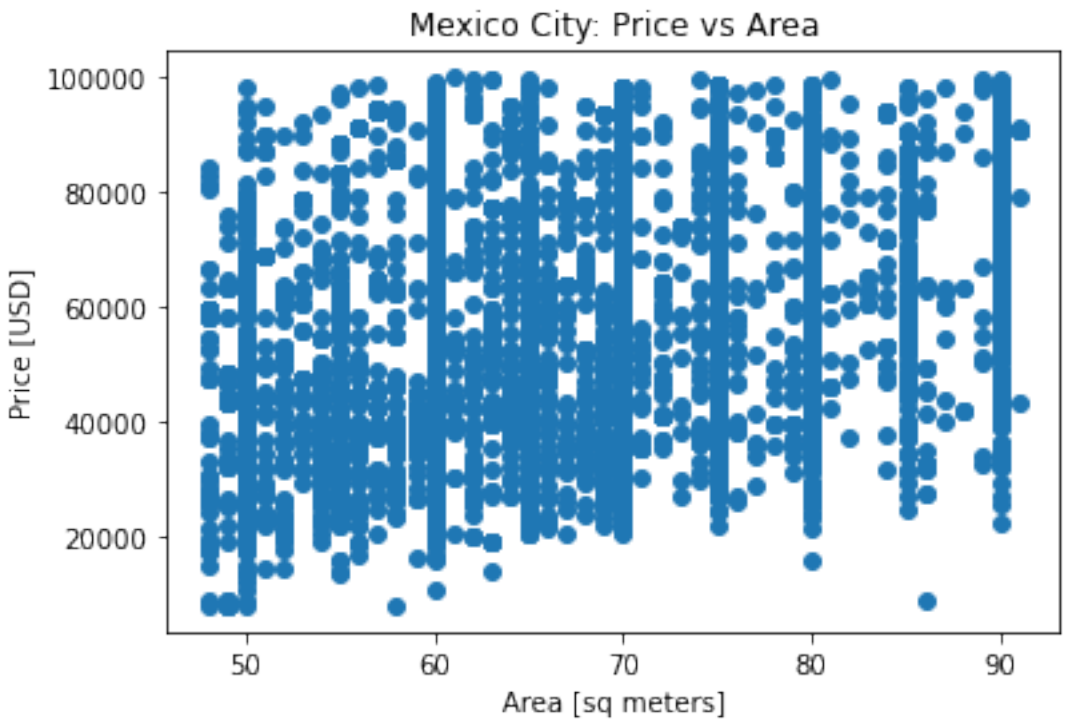
[45]:
```
with open("images/2-5-4.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.4", file)
```

```
<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?

[46]:
```
# 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)
```



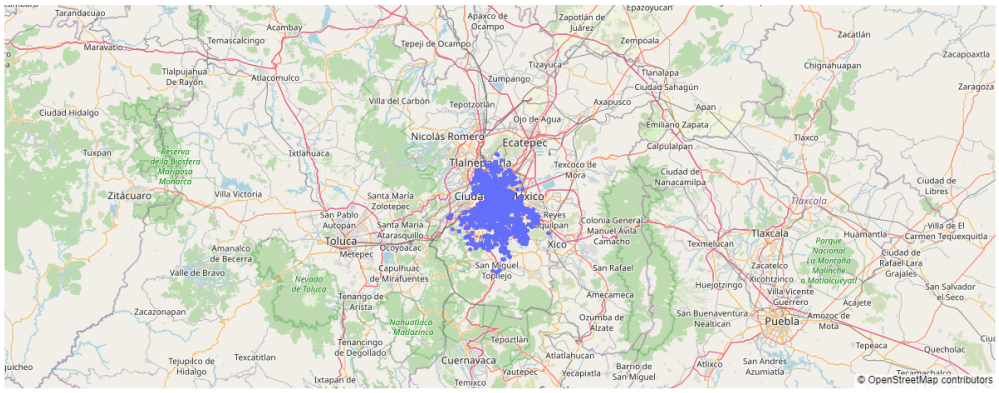
[47]:
```
with open("images/2-5-5.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.5", file)
```

```
<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?

[50]:
```
# Plot Mapbox location and price
fig = px.scatter_mapbox(
    df,  # Our DataFrame
    lat= 'lat',
    lon= 'lon',
    center={"lat": 19.43, "lon": -99.13},  # Map will be centered on Mexico City
    width=600,  # Width of map
    height=600,  # Height of map
    hover_data=["price_aprox_usd"],  # Display price when hovering mouse over␣
 ↪house
)

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

fig.show()
```



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.

[52]:
```
# Split data into feature matrix `X_train` and target vector `y_train`.
features = df.drop(columns = 'price_aprox_usd').columns
target = 'price_aprox_usd'
X_train = df[features]
y_train = df[target]
```

[53]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.7a", X_train)
```

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

[55]:
```
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.

[56]:
```
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: 54246.53149826422
Baseline MAE: 17239.939475888295
```

[57]:
```
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.

[58]:
```
# Build Model
model = make_pipeline(
    OneHotEncoder(use_cat_names = True),
    SimpleImputer(),
    Ridge()
)
# Fit model
model.fit(X_train, y_train)
```

[58]:
```
Pipeline(steps=[('onehotencoder',
                 OneHotEncoder(cols=['borough'], use_cat_names=True)),
                ('simpleimputer', SimpleImputer()), ('ridge', Ridge())])
```

[59]:
```
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.

[60]:
```
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
```

[60]:

	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

[61]:
```
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.

[63]:
```
y_test_pred = pd.Series(model.predict(X_test))
y_test_pred.head()
```

[63]:
```
0    53538.366480
1    53171.988369
2    34263.884179
3    53488.425607
4    68738.924884
dtype: float64
```

[64]:
```
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.

[65]:
```
coefficients = model.named_steps['ridge'].coef_
features = model.named_steps['onehotencoder'].get_feature_names()
feat_imp = pd.Series(coefficients, index = features)
feat_imp
```

[65]:
```
surface_covered_in_m2              291.654156
lat                                478.901375
lon                              -2492.221814
borough_Benito Juárez            13778.188880
borough_Iztacalco                  405.403127
```

```
borough_Azcapotzalco                2459.288646
borough_Coyoacán                    3737.561001
borough_Álvaro Obregón              3275.121061
borough_Iztapalapa                -13349.017448
borough_Cuauhtémoc                   -350.531990
borough_Tláhuac                   -14166.869486
borough_Miguel Hidalgo              1977.314718
borough_Venustiano Carranza        -5609.918629
borough_Tlalpan                    10319.429804
borough_Gustavo A. Madero          -6637.429757
borough_Xochimilco                   929.857400
borough_La Magdalena Contreras     -5925.666450
borough_Cuajimalpa de Morelos       9157.269123
dtype: float64
```

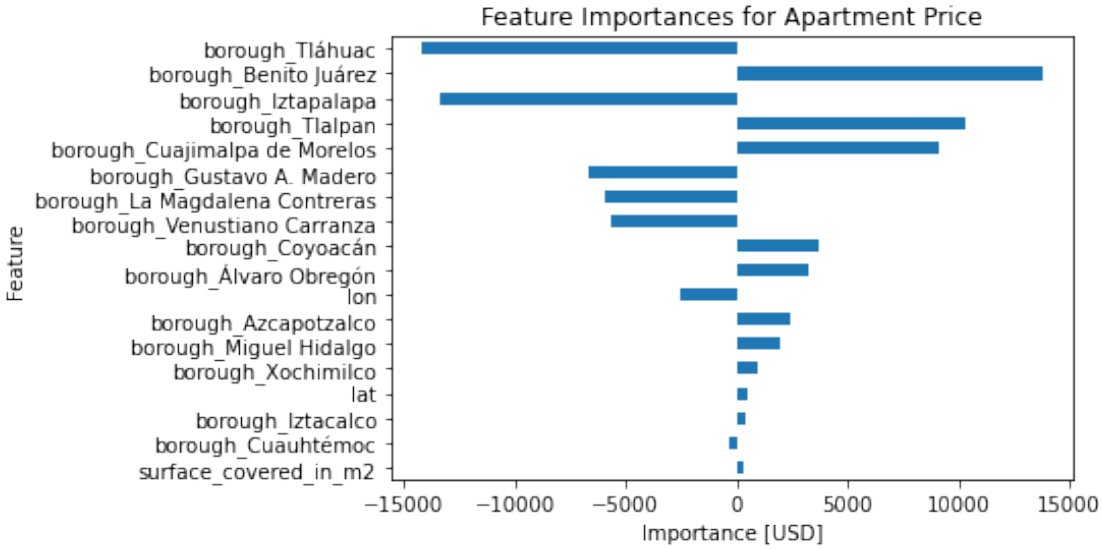
[67]:
```
wqet_grader.grade("Project 2 Assessment", "Task 2.5.12", feat_imp)
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Input In [67], in <cell line: 1>()
----> 1 wqet_grader.grade("Project 2 Assessment", "Task 2.5.12", feat_imp)

File /opt/conda/lib/python3.9/site-packages/wqet_grader/__init__.py:180, in␣
 ↪grade(assessment_id, question_id, submission)
    175 def grade(assessment_id, question_id, submission):
    176   submission_object = {
    177     'type': 'simple',
    178     'argument': [submission]
    179   }
--> 180   return␣
 ↪show_score(grade_submission(assessment_id, question_id, submission_object))

File /opt/conda/lib/python3.9/site-packages/wqet_grader/transport.py:145, in␣
 ↪grade_submission(assessment_id, question_id, submission_object)
    143     raise Exception('Grader raised error: {}'.format(error['message']))
    144   else:
--> 145     raise Exception('Could not grade submission: {}'.
 ↪format(error['message']))
    146 result = envelope['data']['result']
    148 # Used only in testing

Exception: Could not grade submission: Could not verify access to this␣
 ↪assessment: Received error from WQET submission API: You have already passed␣
 ↪this course!
```

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".

[68]:
```
# Create horizontal bar chart
feat_imp.sort_values(key = abs).plot(kind = 'barh')
plt.xlabel('Importance [USD]')
plt.ylabel('Feature')
plt.title('Feature Importances for Apartment Price')
# Don't delete the code below
plt.savefig("images/2-5-13.png", dpi=150)
```



[69]:
```
with open("images/2-5-13.png", "rb") as file:
    wqet_grader.grade("Project 2 Assessment", "Task 2.5.13", file)
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)
Input In [69], in <cell line: 1>()
      1 with open("images/2-5-13.png", "rb") as file:
----> 2     wqet_grader.grade("Project 2 Assessment", "Task 2.5.13", file)

File /opt/conda/lib/python3.9/site-packages/wqet_grader/__init__.py:180, in␣
 ↪grade(assessment_id, question_id, submission)
    175 def grade(assessment_id, question_id, submission):
    176   submission_object = {
    177     'type': 'simple',
    178     'argument': [submission]
    179   }
--> 180   return␣
 ↪show_score(grade_submission(assessment_id, question_id, submission_object))
```

```
File /opt/conda/lib/python3.9/site-packages/wqet_grader/transport.py:145, in␣
 ↪grade_submission(assessment_id, question_id, submission_object)
    143     raise Exception('Grader raised error: {}'.format(error['message']))
    144   else:
--> 145     raise Exception('Could not grade submission: {}'.
 ↪format(error['message']))
    146 result = envelope['data']['result']
    148 # Used only in testing

Exception: Could not grade submission: Could not verify access to this␣
 ↪assessment: Received error from WQET submission API: You have already passed␣
 ↪this course!
```

Copyright © 2022 WorldQuant University. This content is licensed solely for personal use. Redistribution or publication of this material is strictly prohibited.