Can Airbnb data predict the fee of the property?

March 30, 2020

1 Introduction

Sydney is one of the most famous tourist attraction in the world. With a breath-taking coastline, along with stunning skyline, and various impressive landmarks, this city has everything to offer. A common way that tourist choose accomodation in this city is using Airbnb, which allow them to avoid costly fees from hotels, and still have the comfort of staying at home. However, Airbnb price in Sydney has gained an anecdotal reputation to have a large discrepancy between different areas.

A part of this problem was attributed to the location of the Airbnb, as well as the high living cost of the city overall. Sydney is one of the most expensive city to live in, which means accomodation for traveller will need to consider this factor (after all, the host needs to get a stable income to maintain the property). Location wise, it largely depends on where the property is located from the city central. Generally, a property that is within walking distance to the city central will attract a higher fee for stay, as it is more convenient for tourist to access attractions. A cheaper option is usually farther from the city central, and many tourists might not want to spend more time in traffic, considering that their time in a new city is already limited.

As a common knowledge, a property can be expensive, but that does not guarantee that the property will be satisfactory. Which is why the need to find an Airbnb within an acceptable distance to the city central, and have resonable fee, has arised.

1.1 Describing the data

The data was obtained from insideairbnb.com, with some addtional data obtained from webscraping. The data contains listing from different surburbs. The price and location of the property, as well as other useful metadata are also included.

This data will solve the problem by identifying any potential relationship between price and location of a property using statistical tests. Also, this data will combine with visualisation techniques to identify any possible correlation between the property and relevant factors. The data recognises trends that include but not limited to lication and price. The data will also offer insights on the overall Airbnb scene of Sydney.

2 Methodology

2.1 Cleaning the data

As the data was obtained from webscrapping, it includes a wide range of metadata such as info of the host, link to the listing (which might have expired at the time this report was written), and description of the property. Though this data may be necessary for an user to decide confidently which property is worth their money, it is not necessary for prediction and statistical tests. The following tasks were performed: * Dropping cells with missing data, non-descriptive data (such as the information was overly generalised, or using the default "Sydney", and data that are not useful to an average Airbnb user (such as data not in English) * Fixing data types: Stripping off characters that interfere with intepretation of numerical data (for example: commas, currency symbols) * Sorting the data accordingly to the analysis being performed * Coercing data into appropriate data types

2.2 Obtaining general statistical analyses of data

Numerical data are subjected to statistical analysis to get a quick insight of what the data contain, and how the data can be used for further analysis

3 Analyses and Results

3.1 Importing libraries

[1]:
```python
# Data Handling libraries
import pandas as pd
import numpy as np
import collections

# System libraries
import os

# Machine Learning libraries
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, normalize, scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error, r2_score

# Data Visualisation libraries
import matplotlib.pyplot as plt
import plotly
import plotly.express as px
import seaborn as sns

# Mapping Libraries
import folium
```

```
from geopy.geocoders import Nominatim
from urllib.request import urlopen
import json
```

3.2 Loading and cleaning the data

```
[2]: data = pd.read_csv('listings.csv') # Importing data
```

```
/usr/local/anaconda3/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3063: DtypeWarning:

Columns (43,61,62,94) have mixed types.Specify dtype option on import or set
low_memory=False.
```

```
[3]: data.columns # Checking the list of columns.
```

```
[3]: Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'name', 'summary',
           'space', 'description', 'experiences_offered', 'neighborhood_overview',
           ...
           'instant_bookable', 'is_business_travel_ready', 'cancellation_policy',
           'require_guest_profile_picture', 'require_guest_phone_verification',
           'calculated_host_listings_count',
           'calculated_host_listings_count_entire_homes',
           'calculated_host_listings_count_private_rooms',
           'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
          dtype='object', length=106)
```

```
[4]: df = data[[
                'city',
         'zipcode',
                'latitude',
                'longitude',
                'room_type',
                'price',
              ]] # Creating dataframe based on columns that are necessary to predict␣
      ↪price
```

```
[5]: df.sort_values('city') # Sorting data based on Surburbs
     df.rename(columns={'city':'surburb'}, inplace = True) # Changing name of column␣
      ↪to surburb
```

```
/usr/local/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py:4133:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

[6]:
```
df.dropna(subset=['surburb','zipcode'], inplace = True) # Dropping NA rows in␣
 ↪Surburb column
df = df.sort_values('surburb')
```

/usr/local/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1:
SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

[7]:
```
df = df[:-42] # Dropping values in foreign languages. This is the last 42 rows␣
 ↪in the dataset, after sorting by surburb
df.drop_duplicates() # Remove duplicate listings
```

[7]:

	surburb	zipcode	latitude	longitude	room_type	price
806	Bondi Beach	2026	-33.88298	151.27204	Entire home/apt	$220.00
2208	Coogee ,Sydney	2034	-33.91721	151.25792	Entire home/apt	$165.00
26868	Neutral Bay	2089	-33.83155	151.21473	Entire home/apt	$179.00
26869	Neutral Bay	2089	-33.83207	151.21468	Entire home/apt	$161.00
33296	Ryde	2112	-33.82046	151.09741	Private room	$79.00
...	...	...	...	...	...	...
21231	Zetland	2017	-33.91056	151.20470	Private room	$42.00
21276	Zetland	2017	-33.90374	151.20891	Private room	$79.00
14947	Zetland	2017	-33.90953	151.20324	Entire home/apt	$121.00
25541	Zetland	2017	-33.90794	151.20989	Entire home/apt	$199.00
28465	Zetland	2017	-33.90391	151.20657	Entire home/apt	$131.00

[40117 rows x 6 columns]

[8]:
```
# Converting price to numerical data by stripping excess characters and changing␣
 ↪data type
df['price'] = df['price'].str.replace('$','')
df['price'] = df['price'].str.replace(',','')
df['price'] = df.price.astype('float')
```

[9]:
```
# Converting zipcode to numerical data
df['zipcode'] = pd.to_numeric(df['zipcode'], errors='coerce')
```

3.3 Exploratory analysis

[10]:
```
df.describe()
```

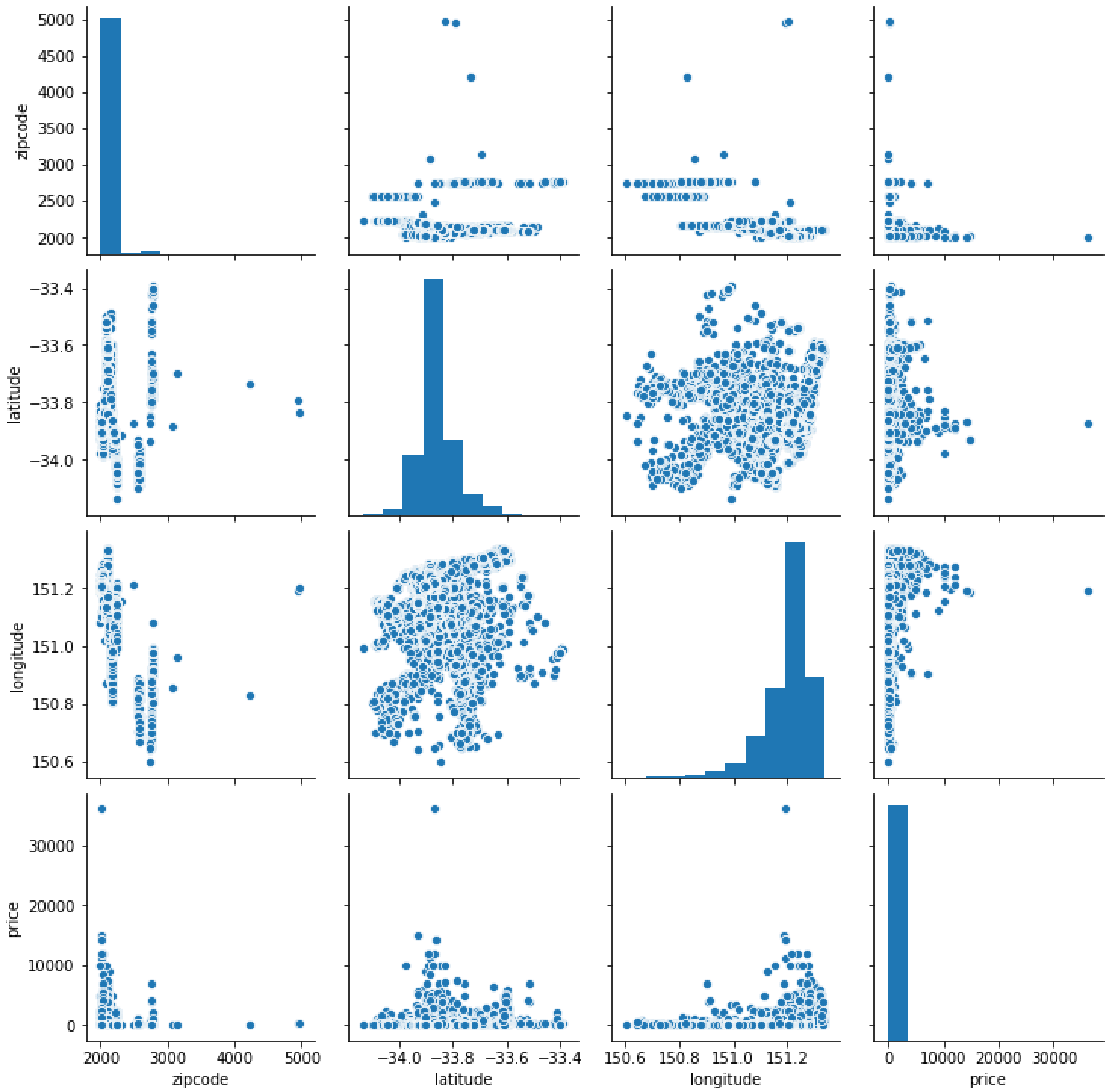
[10]:

	zipcode	latitude	longitude	price
count	40250.000000	40254.000000	40254.000000	40254.000000
mean	2072.786484	-33.863237	151.197834	218.409823
std	99.983822	0.072408	0.088337	449.336866
min	2000.000000	-34.135590	150.601470	0.000000
25%	2021.000000	-33.898840	151.174608	80.000000
50%	2035.000000	-33.881560	151.212250	131.000000
75%	2099.000000	-33.830883	151.258078	219.000000
max	4971.000000	-33.390750	151.339870	36128.000000

The average Airbnb price of Sydney is 218 AUD. Interestingly, this does not differ from an average rental price, provided that you rent a room in Sydney.

[11]:
```
sns.pairplot(df)
```

[11]:
```
<seaborn.axisgrid.PairGrid at 0x1a1ff19250>
```



So far, there is no obvious correlation between price and any other parameters. Also, from this exploratory analysis, it was found that some properties are listed as 0. This data is invalid, as a fee of 0.00 indicates that the host has likely has their account being suspended, and their active listings defaulted to hidden. When scrapping Airbnb, hidden fees were displayed as 0.

[12]:
```
# Drop price = 0 as this is invalid. Airbnb listing with price = 0 indicate the␣
↪host violates T&C of Airbnb, rendering the property unrentable
indexNames = df[ df['price'] == 0 ].index
df.drop(indexNames, inplace = True)
```

[13]:
```
df = df.sort_values('price')
df
```

```
[13]:                  surburb  zipcode  latitude  longitude        room_type  \
      39580           Stanmore   2048.0 -33.89545  151.17129     Private room
      9896           Woollahra   2025.0 -33.92764  151.23620     Private room
      16520              Bondi   2026.0 -33.89685  151.26079      Shared room
      16078           Balgowlah  2093.0 -33.79835  151.26332  Entire home/apt
      26795    North Parramatta  2151.0 -33.79950  151.00480  Entire home/apt
      ...                  ...      ...       ...        ...              ...
      37181             Mascot   2020.0 -33.93051  151.18839  Entire home/apt
      37178             Mascot   2020.0 -33.93203  151.18834  Entire home/apt
      37179             Mascot   2020.0 -33.93089  151.18828  Entire home/apt
      37180             Mascot   2020.0 -33.93185  151.18736  Entire home/apt
      30076            Pyrmont   2009.0 -33.86908  151.19359  Entire home/apt

               price
      39580      4.0
      9896       4.0
      16520      4.0
      16078     13.0
      26795     13.0
      ...        ...
      37181  14885.0
      37178  14885.0
      37179  14885.0
      37180  14885.0
      30076  36128.0

      [40250 rows x 6 columns]
```

3.4 Encoding categorical data

```
[14]: collections.Counter(df['room_type']) # Getting numbers of each category for␣
       ↪reference
```

```
[14]: Counter({'Private room': 14013,
               'Shared room': 767,
               'Entire home/apt': 25200,
               'Hotel room': 270})
```

```
[15]: df["room_type"] = df["room_type"].astype('category')
      df["room_type"] = df["room_type"].cat.codes
      df.sort_values('room_type')
      collections.Counter(df.room_type)
```

```
[15]: Counter({2: 14013, 3: 767, 0: 25200, 1: 270})
```

After encoding, the types of properties are listed as follow

Symbol	Room category
0	`Entire property`
1	`Hotel room`
2	`Private room`
3	`Shared room`

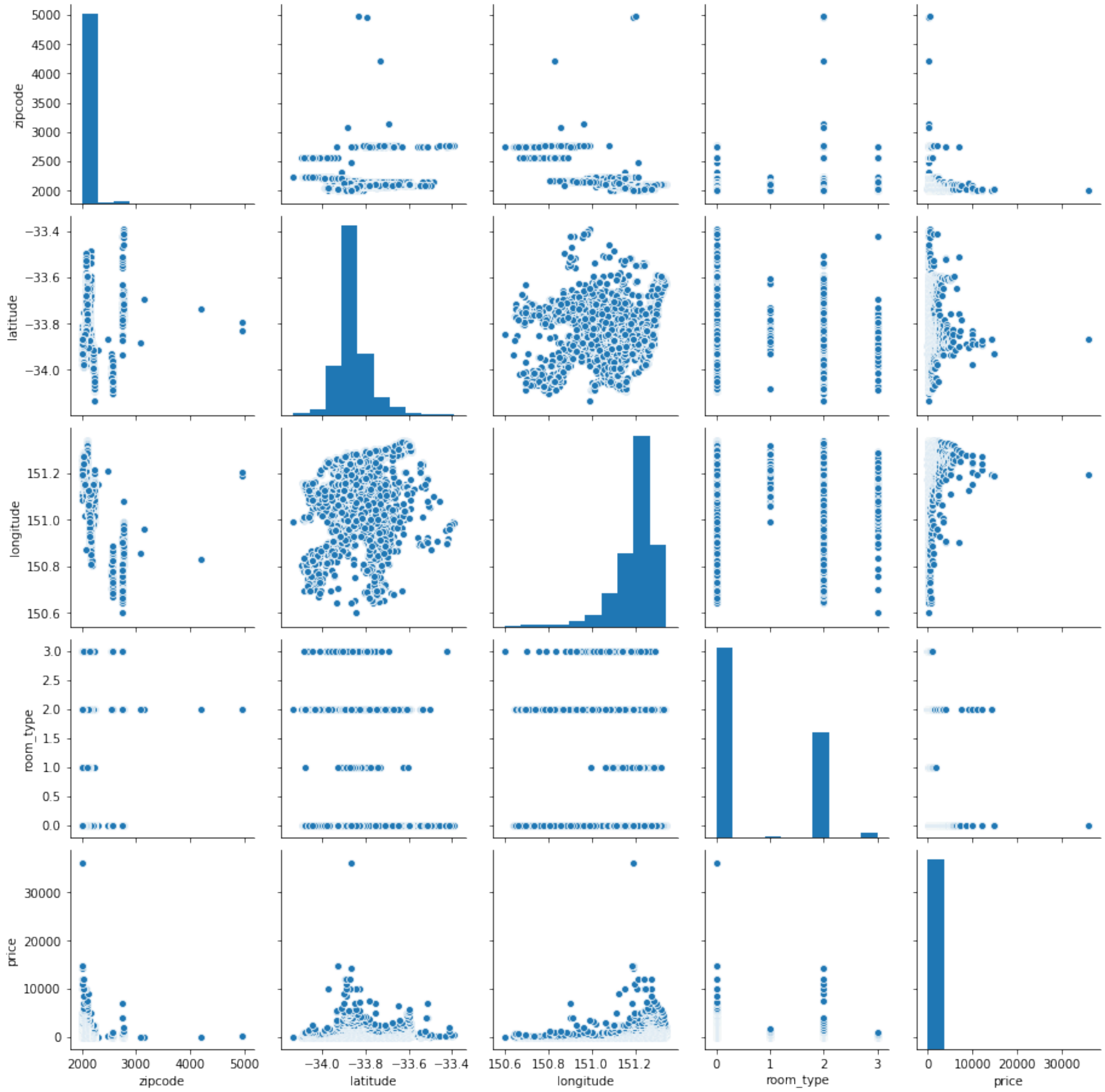
3.5 Repeating exploratory analysis

[16]:
```
df.describe()
```

[16]:
```
            zipcode      latitude     longitude     room_type         price
count  40246.000000  40250.000000  40250.000000  40250.000000  40250.000000
mean    2072.782090    -33.863231    151.197835      0.760174    218.431528
std       99.982776      0.072408      0.088339      0.996478    449.353917
min     2000.000000    -34.135590    150.601470      0.000000      4.000000
25%     2021.000000    -33.898840    151.174633      0.000000     80.000000
50%     2035.000000    -33.881555    151.212250      0.000000    131.000000
75%     2099.000000    -33.830880    151.258078      2.000000    219.000000
max     4971.000000    -33.390750    151.339870      3.000000  36128.000000
```

[17]:
```
sns.pairplot(df)
```

[17]:
```
<seaborn.axisgrid.PairGrid at 0x1a2130a190>
```



This repeat produced a more comprehensive analysis for the data. Though there were not any strong correlation, as indicated by the plot above, further exploring of this data could unveal some hidden insights.

As there is a distinction between room type and price, with hotels staying quite consistent, a linear regression was used to test this relationship.

3.6 Linear regression test

[18]:
```
lreg = LinearRegression()

X = df['room_type']
X = X.values.reshape(-1,1) #1D value needs to be reshaped for LinearRegression␣
↪function.
```

```
y = df['price']
y = y.values.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)

model = lreg.fit(X_train, y_train)
```

[19]:
```
y_pred = lreg.predict(X_test)

r2s = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
lreg_coeff_pos = lreg.coef_[np.argsort(-lreg.coef_)[:10]]
lreg_coeff_neg = lreg.coef_[np.argsort(lreg.coef_)[:10]]
lreg_coeff_abs = abs(lreg.coef_)
lreg_coeff_max = "";#lreg.coef_[np.argsort(lreg_coeff_abs)[:10]]
lreg_coeff_min = lreg.coef_[np.argsort(lreg_coeff_abs)[:10]]

print('R2-score           :', r2s )
print('MEAN Squared Error :', mse )
print('ABS coefs          :', lreg_coeff_abs)
print('Coefficients:')
print('\tPOSITIVE = ',lreg_coeff_pos)
print('\tNEGATIVE = ',lreg_coeff_neg)
```

```
R2-score           : 0.033313968037018915
MEAN Squared Error : 271242.8757931954
ABS coefs          : [[96.7521553]]
Coefficients:
        POSITIVE =  [[[-96.7521553]]]
        NEGATIVE =  [[[-96.7521553]]]
```

As R2-score is low, and mean square error is high, the relationship between room type and price is not statistically significant. Principal Component Regression test will further prove that.

3.7 Principal Component Regression test

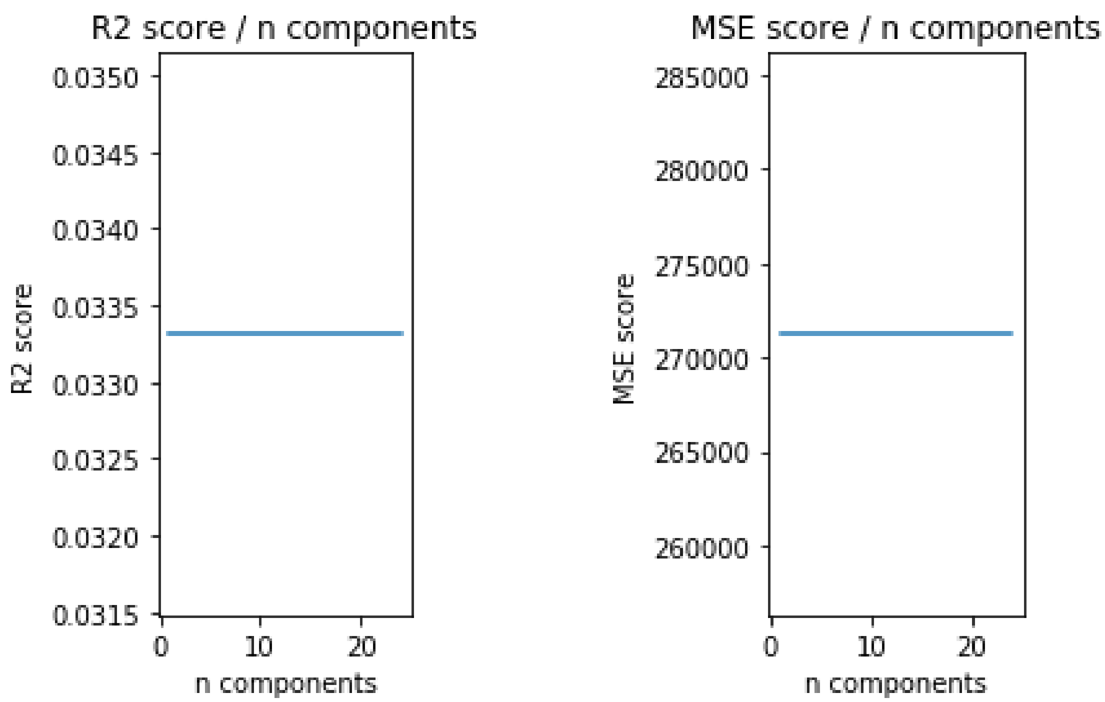
[20]:
```
X = df['room_type']
X = X.values.reshape(-1,1)
y = df['price']
y = y.values.reshape(-1,1)

pca   = PCA(svd_solver='auto', random_state=0)
X_pca = pca.fit_transform(scale(X))
```

[21]:
```
n_component_list = range(1, 25)
r2_list = []
mse_list = []
```

```
# Second linear regression. This is necessary for the graph below
for i in n_component_list:
    lreg = LinearRegression()
    X_train, X_test, y_train, y_test = train_test_split(X_pca[:,:i], y,␣
 ↪test_size=0.2, random_state=0)
    model = lreg.fit(X_train, y_train)
    # check the result
    y_pred = lreg.predict(X_test)
    r2 = r2_score(y_test, y_pred) # r2 score
    mse = mean_squared_error(y_test, y_pred) # mse
    r2_list.append(r2)
    mse_list.append(mse)

scores_df = pd.DataFrame.from_dict(dict([('NComponents', n_component_list),
                                         ('R2', r2_list),
                                         ('MSE', mse_list)]))
scores_df.set_index('NComponents', inplace=True)
```

[22]:

```
# Plotting the scores
plt.subplot(1, 3, 1)
scores_df['R2'].plot(kind='line')
plt.title('R2 score / n components')
plt.ylabel('R2 score')
plt.xlabel('n components')

plt.subplot(1, 3, 3)
scores_df['MSE'].plot(kind='line')
plt.title('MSE score / n components')
plt.ylabel('MSE score')
plt.xlabel('n components')

plt.show()
```



The result from this test further confirms that there is no relationship between room type and price.

3.8 Mapping the most expensive Airbnb property in Sydney

Latitude and longitute of the most expensive Airbnb in Sydney was obtained by sorting the data by price (see above). The library **folium** was employed to locate this property on a map.

[23]:
```
geo = Nominatim(user_agent='Coursera-Capstone')
address = 'Sydney'
location = geo.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(f'The geographical coordinates of Sydney are {latitude}, {longitude}.')

# create map of Sydney using latitude and longitude values
map_sydney = folium.Map(location=[latitude, longitude],tiles="OpenStreetMap",␣
 ↪zoom_start=12)


# add markers to map

print('The most expensive Airbnb Property in Sydney is in Pyrmont')
folium.CircleMarker(
```

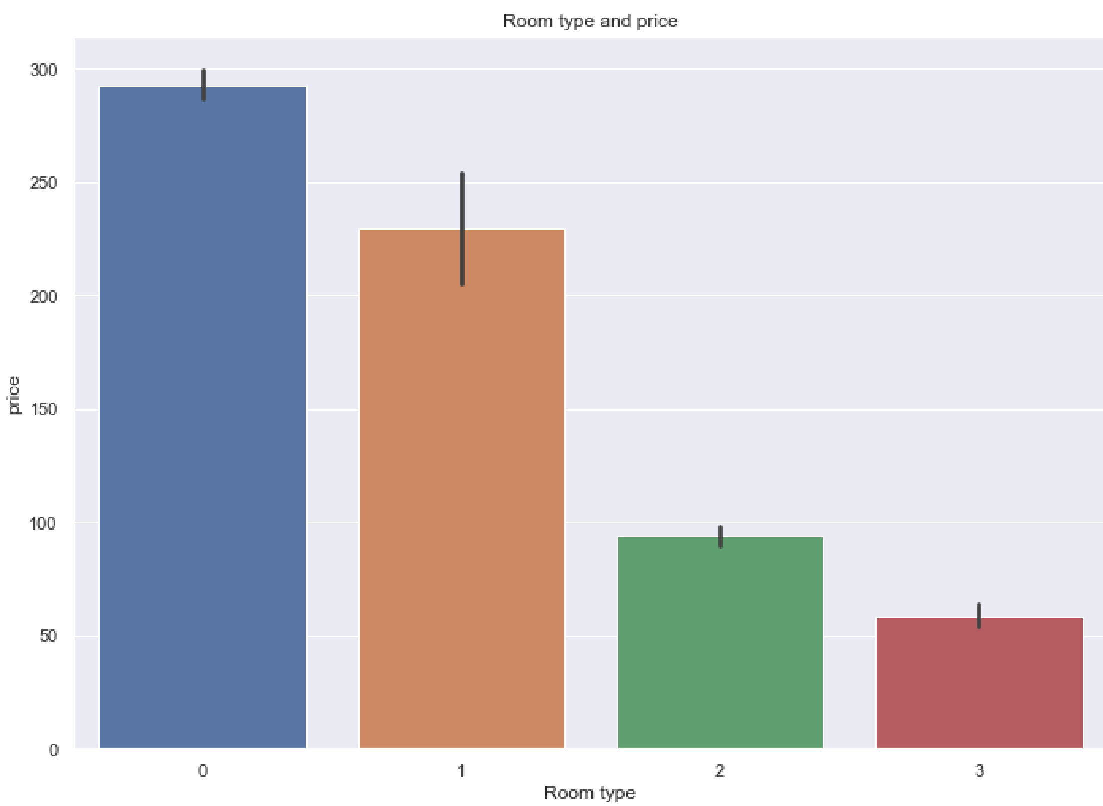
```
    [-33.86908,151.19359],
    radius=6,
    color='red',
    fill=True,
    fill_color='#f2f6f5',
    fill_opacity=0.5,
    parse_html=False).add_to(map_sydney)
map_sydney
```

```
The geograpical coordinates of Sydney are -33.8548157, 151.2164539.
The most expensive Airbnb Property in Sydney is in Pyrmont
```

[23]:
```
<folium.folium.Map at 0x1a214d12d0>
```

As the data from Airbnb did not indicate a relationship between room type and price, or location and price, a comparison between each room type was made. This was visualised using a simple bar graph of the average price based on room type

[24]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax0 = sns.barplot(data = df.reset_index(), x = 'room_type', y = 'price')
ax0.set_title("Room type and price")
ax0.set_xlabel('Room type')
plt.show()
```



The price of room types differs significantly between four room types. Notably, whole property (category 0) was the most expensive. The least expensive type, as expected, is shared room (category 3).

4 Conclusion

It appears that there is no correlation between the price of a property and its distance to the city central. Scatter plot of the location of the property does not show any significant relationship between price and longitude/latitude. However, this data might not been intepreted sufficiently to come to that conclusion, as there might be different statistical tests or visualisation tools that can intepret this data more succintly. Ideally, if a tool can cluster the data on a map, with different color codes represent the average price of the property, could prove to be helpful in this further analysis.

Unsurprisingly, whole property has the most expensive fees, followed by hotel rooms. With that being articulated, a private room in a occupied property might be the most suitable for tourists. The price for this type of property is reasonable and half as much comparing to a hotel room. With tourists being parsimonious, a shared room could be an option, as it is the least expensive choice and it is half as expensive as a private room. Though, if this option is considered, the user might need to sacrifice some privacy.

The most expensive property is in Pyrmont. On a map, it can be seen that Pyrmont is right in the middle of city central, and has a view towards some tourist attractions such as the Darling Habour and Sydney Opera House. Also, it is within walking distance to many tourists attractions. There might be more reasons that makes this property so expensive, but if you crave an "once-of-a-lifetime" experience, it might worth a try.

For the purpose of this tasks, this notebook has completed the analysis required for the Data Science Professional Certificate by IBM. If you have any questions, feel free to reach out.

All the best

Hieu Nguyen

LinkedIn: www.linkedin.com/in/hieungx

```
[ ]:
```