2440016804 - Rio Pramana - LA01 - Assignment 6

Import libraries and dataset

In [1]:
```
import pandas as pd
import numpy as np
from numpy import cov
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
import scipy.stats as ss
from scipy.stats import spearmanr
from scipy.stats import pearsonr
```

In [2]:
```
# Importing the dataset, downloaded file is in the same folder
csv_path = "listings.csv"
listings_df = pd.read_csv(csv_path)
```

Run a quick check on the dataset

In [3]:
```
listings_df.shape
```

Out[3]:
```
(7907, 16)
```

In [4]:
```
listings_df.head(5)
```

Out[4]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_
0	49091	COZICOMFORT LONG TERM STAY ROOM 2	266763	Francesca	North Region	Woodlands	1.44255	103.79580	Private room	83	180	
1	50646	Pleasant Room along Bukit Timah	227796	Sujatha	Central Region	Bukit Timah	1.33235	103.78521	Private room	81	90	

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_
2	56334	COZICOMFORT	266763	Francesca	North Region	Woodlands	1.44246	103.79667	Private room	69	6	
3	71609	Ensuite Room (Room 1 & 2) near EXPO	367042	Belinda	East Region	Tampines	1.34541	103.95712	Private room	206	1	
4	71896	B&B Room 1 near Airport & EXPO	367042	Belinda	East Region	Tampines	1.34567	103.95963	Private room	94	1	

In [5]:
```
listings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7907 entries, 0 to 7906
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              7907 non-null   int64  
 1   name                            7905 non-null   object 
 2   host_id                         7907 non-null   int64  
 3   host_name                       7907 non-null   object 
 4   neighbourhood_group             7907 non-null   object 
 5   neighbourhood                   7907 non-null   object 
 6   latitude                        7907 non-null   float64
 7   longitude                       7907 non-null   float64
 8   room_type                       7907 non-null   object 
 9   price                           7907 non-null   int64  
 10  minimum_nights                  7907 non-null   int64  
 11  number_of_reviews               7907 non-null   int64  
 12  last_review                     5149 non-null   object 
 13  reviews_per_month               5149 non-null   float64
 14  calculated_host_listings_count  7907 non-null   int64  
 15  availability_365                7907 non-null   int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 988.5+ KB
```

Agar lebih mudah, dependent variable (yaitu price) akan dipindahkan menjadi kolom paling terakhir

In [6]:
```
listings_new = listings_df.copy()
cols_at_end = ['price']
listings_new = listings_new[[c for c in listings_new if c not in cols_at_end]
        + [c for c in cols_at_end if c in listings_new]]
listings_new.head(5)
```

Out[6]:

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	minimum_nights	number_of_reviews
0	49091	COZICOMFORT LONG TERM STAY ROOM 2	266763	Francesca	North Region	Woodlands	1.44255	103.79580	Private room	180	1
1	50646	Pleasant Room along Bukit Timah	227796	Sujatha	Central Region	Bukit Timah	1.33235	103.78521	Private room	90	18
2	56334	COZICOMFORT	266763	Francesca	North Region	Woodlands	1.44246	103.79667	Private room	6	20
3	71609	Ensuite Room (Room 1 & 2) near EXPO	367042	Belinda	East Region	Tampines	1.34541	103.95712	Private room	1	14
4	71896	B&B Room 1 near Airport & EXPO	367042	Belinda	East Region	Tampines	1.34567	103.95963	Private room	1	22

Handling missing data

In [7]:
```
listings_new.isnull().sum()
```

Out[7]:
```
id                              0
name                            2
host_id                         0
host_name                       0
neighbourhood_group             0
neighbourhood                   0
latitude                        0
longitude                       0
room_type                       0
minimum_nights                  0
```

```
number_of_reviews                   0
last_review                      2758
reviews_per_month                2758
calculated_host_listings_count      0
availability_365                    0
price                               0
dtype: int64
```

Pada dataset, terdapat 3 kolom yang memiliki missing data, yaitu kolom name, neighbourhood_group, dan room_type.

Handle missing data on 'name' column

Untuk kolom name, kita menghandle missing data dengan mereplace missing data tersebut menggunakan mode dari kolom name karena kolom name berisi categorical data

In [8]:
```
listings_new.name.value_counts()
```

Out[8]:
```
Luxury hostel with in-cabin locker - Single mixed     13
Studio Apartment - Oakwood Premier                     9
Inviting & Cozy 1BR APT 3 mins from Tg Pagar MRT       9
Stylish 1BR Located 7 mins from Tg Pagar MRT           8
City-located 1BR loft apartment *BRAND NEW*            8
                                                      ..
Boonlay 16sqm Cosy Master Room for Rent                1
Tanjong Pagar Pristine Studio Apartment                1
lavLoftbed *RmT, no-sharing, wifi, mrt                 1
Newly furnished spacious room                          1
Amazing room with private bathroom walk to Orchard     1
Name: name, Length: 7457, dtype: int64
```

In [9]:
```
listings_new.name.mode()
```

Out[9]:
```
0    Luxury hostel with in-cabin locker - Single mixed
dtype: object
```

Mode dari kolom name adalah 'Luxury hostel with in-cabin locker - Single mixed', maka missing value pada kolom ini akan direplace dengan value tersebut. Untuk mengaksesnya, menggunakan [0] dibelakang mode

In [10]:
```
listings_new.name.mode()[0]
```

Out[10]:
```
'Luxury hostel with in-cabin locker - Single mixed'
```

In [11]:
```
listings_new['name'].fillna(listings_new['name'].mode()[0], inplace = True)
```

In [12]:
```
listings_new.name.value_counts()
```

Out[12]:
```
Luxury hostel with in-cabin locker - Single mixed     15
Inviting & Cozy 1BR APT 3 mins from Tg Pagar MRT       9
Studio Apartment - Oakwood Premier                     9
Superhost 1BR APT in the heart of Tg Pagar             8
Stylish 1BR Located 7 mins from Tg Pagar MRT           8
                                                      ..
Boonlay 16sqm Cosy Master Room for Rent                1
Tanjong Pagar Pristine Studio Apartment                1
lavLoftbed *RmT, no-sharing, wifi, mrt                 1
Newly furnished spacious room                          1
Amazing room with private bathroom walk to Orchard     1
Name: name, Length: 7457, dtype: int64
```

In [13]:
```
listings_new.isnull().sum()
```

Out[13]:
```
id                                  0
name                                0
host_id                             0
host_name                           0
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
minimum_nights                      0
number_of_reviews                   0
last_review                      2758
reviews_per_month                2758
calculated_host_listings_count      0
availability_365                    0
price                               0
dtype: int64
```

Kolom name sudah tidak ada missing value lagi dan direplace dengan value modenya

Handle missing data on 'last_review' column

Untuk kolom last_review, kita menghandle missing data dengan mereplace missing data tersebut menggunakan mode dari kolom last_review karena kolom last_review berisi categorical data

In [14]:
```
listings_new.last_review.value_counts()
```

Out[14]:
```
2019-08-12    152
2019-08-11    128
2019-08-13    110
2019-08-10     87
2019-08-08     78
             ...
2016-12-03      1
2016-01-18      1
2016-07-27      1
2017-08-19      1
2019-03-22      1
Name: last_review, Length: 1001, dtype: int64
```

Mode dari kolom last_review:

In [15]:
```
listings_new.last_review.mode()
```

Out[15]:
```
0    2019-08-12
dtype: object
```

Replace missing values:

In [16]:
```
listings_new['last_review'].fillna(listings_new['last_review'].mode()[0], inplace = True)
```

In [17]:
```
listings_new.last_review.value_counts()
```

Out[17]:
```
2019-08-12    2910
2019-08-11     128
2019-08-13     110
2019-08-10      87
2019-08-08      78
              ...
2016-12-03       1
2016-01-18       1
2016-07-27       1
```

```
2017-08-19       1
2019-03-22       1
Name: last_review, Length: 1001, dtype: int64
```

In [18]:
```
listings_new.isnull().sum()
```

Out[18]:
```
id                                    0
name                                  0
host_id                               0
host_name                             0
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
minimum_nights                        0
number_of_reviews                     0
last_review                           0
reviews_per_month                  2758
calculated_host_listings_count        0
availability_365                      0
price                                 0
dtype: int64
```

Kolom last_review sudah tidak ada missing value lagi dan direplace dengan value modenya

Handle missing data on 'reviews_per_month' column

Untuk kolom reviews_per_month, kita menghandle missing data dengan mereplace missing data tersebut menggunakan mode dari kolom reviews_per_month karena lebih optimal jika kita menggunakan reviews_per_month yang paling sering muncul untuk menghindari kemungkinan penurunan akurasi dalam jumlah yang besar

In [19]:
```
listings_new.reviews_per_month.value_counts()
```

Out[19]:
```
1.00    172
0.04    104
0.08     96
0.05     93
0.12     92
       ...
4.02      1
3.92      1
```

```
3.52      1
3.57      1
8.00      1
Name: reviews_per_month, Length: 527, dtype: int64
```

In [20]:
```
listings_new.reviews_per_month.mode()
```

Out[20]:
```
0    1.0
dtype: float64
```

In [21]:
```
listings_new['reviews_per_month'].fillna(listings_new['reviews_per_month'].mode()[0], inplace = True)
```

In [22]:
```
listings_new.reviews_per_month.value_counts()
```

Out[22]:
```
1.00    2930
0.04     104
0.08      96
0.05      93
0.10      92
        ...
4.02       1
3.92       1
3.52       1
3.57       1
8.00       1
Name: reviews_per_month, Length: 527, dtype: int64
```

In [23]:
```
listings_new.isnull().sum()
```

Out[23]:
```
id                              0
name                            0
host_id                         0
host_name                       0
neighbourhood_group             0
neighbourhood                   0
latitude                        0
longitude                       0
room_type                       0
minimum_nights                  0
number_of_reviews               0
```

```
last_review                     0
reviews_per_month               0
calculated_host_listings_count  0
availability_365                0
price                           0
dtype: int64
```

Kolom reviews_per_month sudah tidak ada missing value lagi dan direplace dengan value modenya

1. Discuss to find the correlation and covariance of the data (the variables used can be categorical vs categorical, categorical vs numeric, or numeric vs numeric variables).

Correlation

Untuk menemukan correlation antar variabel, saya akan membuat satu contoh untuk masing-masing kategori (categorical vs categorical 1 contoh, categorical vs numeric 1 contoh, dan numeric vs numeric 1 contoh)

Categorical vs Categorical

Untuk menghitung correlation antar categorical data tanpa encoding, kita bisa menggunakan Cramers V statistic

In [24]:
```
def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x,y)
    chi2 = ss.chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2/n
    r,k = confusion_matrix.shape
    phi2corr = max(0, phi2-((k-1)*(r-1))/(n-1))
    rcorr = r-((r-1)**2)/(n-1)
    kcorr = k-((k-1)**2)/(n-1)
    return np.sqrt(phi2corr/min((kcorr-1),(rcorr-1)))
```

In [25]:
```
## Menghitung correlation name dengan room_type
cramers_v(listings_new["name"], listings_new["room_type"])
```

Out[25]:
```
0.22506158567250478
```

Dapat dilihat bahwa name dengan room_type memiliki correlation 0,225 Berarti bisa dikatakan bahwa mereka memiliki positive correlation tetapi

correlationnya sangat lemah

Categorical vs Numerical

Untuk mengecek correlation antara categorical dan numerical data, kita bisa menggunakan **ANOVA**. Menghitung ANOVA bisa menggunakan library scipy.stats menggunakan method f_oneway dan outputnya adalah F dan p. Jika F semakin mendekati 0, maka correlation kedua variable tersebut semakin lemah. Disini akan digunakan **room_type (categorical independent) dan price (numerical dependent variable)**

In [26]:
```
F, p = ss.f_oneway(listings_new[listings_new.room_type=="Entire home/apt"].price,
                   listings_new[listings_new.room_type=="Private room"].price,
                   listings_new[listings_new.room_type=="Shared room"].price)
print("Statistics Values: ",np.round(F,2), "\n","P _Value        :",np.round(p,2))
```

```
Statistics Values:  131.68
 P _Value        : 0.0
```

Dari hasil tersebut didapat bahwa F nya adalah 131,86 yang menandakan bahwa terdapat correlation antara room_type dengan price

Numerical vs Numerical

Untuk menghitung correlation antar numerical data bisa digunakan Pearson dan Spearman correlation, misalnya antara variable number_of_reviews dengan price

In [27]:
```
pearsonr_corr, _ = pearsonr(listings_new["number_of_reviews"], listings_new["price"])
print("Pearson Correlation: ", np.round(pearsonr_corr,2))
spearmanr_corr, _ = spearmanr(listings_new["number_of_reviews"], listings_new["price"])
print("Spearman Correlation: ", np.round(spearmanr_corr,2))
```

```
Pearson Correlation:  -0.04
Spearman Correlation:  -0.08
```

Karena Pearson dan Spearman correlationnya sangat mendekati 0, dapat dikatakan bahwa hampir tidak ada correlation sama sekali antara number_of_reviews dengan price

Covariance

Untuk menghitung covariance diperlukan data yang numerical, berarti untuk categorical data perlu dilakukan encoding terlebih dahulu sebelum melakukan perhitungan covariance. Untuk contoh ini saya hanya akan menunjukkan perhitungan covariance untuk variables yang sudah numerical, yaitu variable number_of_reviews dengan price

In [28]:
```
covariance = cov(listings_new["number_of_reviews"], listings_new["price"])
print(covariance)
```

```
[[   882.55017059   -424.59206261]
 [  -424.59206261 115727.60260831]]
```

Dari covariance matrix diatas, dapat dilihat bahwa number_of_reviews dengan price memiliki negative relation satu sama lain karena mereka memiliki nilai covariance negatif, yaitu -424

2. Normalized the data using Standardization or Range Scaling method.

Untuk categorical data pada umumnya tidak perlu dilakukan normalization karena jika dilakukan encoding, biasanya scalenya adalah 0-1 sehingga scale differencenya sangat kecil, tidak seperti values yang memiliki range seperti 1-100000. Maka, normalization biasanya dilakukan/ditujukan untuk numerical variables.

Untuk normalization dan feature selection & extraction, saya akan menggunakan numerical data dari dataset ini

In [29]:
```
listings_numeric = listings_new.select_dtypes(include='number')
listings_numeric.head(5)
```

Out[29]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	price
0	49091	266763	1.44255	103.79580	180	1	0.01	2	365	83
1	50646	227796	1.33235	103.78521	90	18	0.28	1	365	81
2	56334	266763	1.44246	103.79667	6	20	0.20	2	365	69
3	71609	367042	1.34541	103.95712	1	14	0.15	9	353	206
4	71896	367042	1.34567	103.95963	1	22	0.22	9	355	94

Normalization using Standardization

In [30]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
listings_standardized = pd.DataFrame(scaler.fit_transform(listings_numeric), columns = listings_numeric.columns)
listings_standardized.head(5)
```

Out[30]:

	id	host_id	latitude	longitude	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count	availability_365	p
0	-2.296403	-1.109569	4.198050	-1.213308	3.860356	-0.397477	-0.981392	-0.592769	1.069556	-0.25
1	-2.296250	-1.110045	0.593859	-1.455798	1.722181	0.174801	-0.721213	-0.608122	1.069556	-0.25
2	-2.295690	-1.109569	4.195107	-1.193387	-0.273450	0.242128	-0.798303	-0.592769	1.069556	-0.29
3	-2.294187	-1.108345	1.020998	2.480602	-0.392238	0.040147	-0.846485	-0.485293	0.987427	0.10
4	-2.294159	-1.108345	1.029502	2.538076	-0.392238	0.309454	-0.779031	-0.485293	1.001115	-0.22

3. Do feature selection and extraction

Untuk melakukan feature selection and extraction, saya menggunakan SelectKBest dari scikit-learn untuk mendapatkan top features

SelectKBest memilih feature-feature sesuai dengan k highest score (misalnya ingin memilih 4 top features, maka features yang dipilih adalah yang scorenya berada di top 4 highest scores)

In [31]:
```
#Import SelectKBest dan f_regression
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
```

In [32]:
```
#Extracting independent variables:
x = listings_standardized.iloc[:,:-1].values #Extract semua kolom kecuali kolom terakhir
print(x)
```

```
[[-2.29640274 -1.10956898  4.19805041 ... -0.98139223 -0.59276857
   1.06955603]
 [-2.29624974 -1.11004475  0.59385877 ... -0.7212132  -0.60812221
   1.06955603]
 [-2.2956901  -1.10956898  4.19510688 ... -0.79830328 -0.59276857
   1.06955603]
 ...
 [ 1.44838721  2.32350012 -0.04357948 ... -0.02740247 -0.57741493
  -0.24451526]
 [ 1.44850105  1.86426245 -0.61364355 ... -0.02740247 -0.59276857
  -1.2232246 ]
```

```
[ 1.4487243  -0.76133463 -0.57145292 ... -0.02740247 -0.51600037
  1.06955603]]
```

In [33]:

```
#Extracting dependent variable:
y = listings_standardized.iloc[:,9].values #Extract kolom terakhir
print(y)
```

```
[-0.2537966  -0.25967608 -0.29495297 ... -0.32729011 -0.33316959
 -0.30671193]
```

In [34]:

```
# Define feature selection
fs = SelectKBest(score_func=f_regression, k=4) #Misalnya memilih 4 top features
# Apply feature selection
x_selected = fs.fit_transform(x, y)
print("Shape 4 top features: ", x_selected.shape)
print(x_selected)
```

```
Shape 4 top features:  (7907, 4)
[[-2.29640274 -1.10956898  4.19805041 -0.39747656]
 [-2.29624974 -1.11004475  0.59385877  0.17480096]
 [-2.2956901  -1.10956898  4.19510688  0.24212773]
 ...
 [ 1.44838721  2.32350012 -0.04357948 -0.43113994]
 [ 1.44850105  1.86426245 -0.61364355 -0.43113994]
 [ 1.4487243  -0.76133463 -0.57145292 -0.43113994]]
```

Dari hasil x_selected, dapat dilihat bahwa 4 top features untuk memprediksi dependent variable price adalah **id, host_id, latitude, number_of_reviews**