Lab 1 Exploratory Analysis - Problems (Sakshi Suman)

February 8, 2022

1 Exploratory Analysis

1.1 Problems:

Load the NYC AirBnB Truncated Dataset. This dataset is a mirror of the full NYC AirBnB dataset found at Kaggle, but only contains the first 10,000 entries.

https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data

For the numerical features,

1) Display histograms for the numerical features.
2) Construct the scatter plots of price with each of the numerical features.
3) Display the correlation histogram.
4) Using numerical features to predict the renting price.
5) Write down the predict function from (4)
6) Calculate the RSS cost.

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt

airbnb_data = pd.read_csv("https://raw.githubusercontent.com/tipthederiver/
↪Math-7243-2020/master/Datasets/NYCAirBnB/train.csv")
```

[2]:
```
airbnb_data.head()
```

[2]:
```
                                              name  host_id    host_name  \
0                 Clean & quiet apt home by the park     2787         John
1                              Skylit Midtown Castle     2845     Jennifer
2                THE VILLAGE OF HARLEM…NEW YORK !     4632    Elisabeth
3                    Cozy Entire Floor of Brownstone     4869  LisaRoxanne
4   Entire Apt: Spacious Studio/Loft by central park     7192        Laura

  neighbourhood_group neighbourhood  latitude  longitude        room_type  \
0            Brooklyn    Kensington  40.64749  -73.97237     Private room
```

```
1           Manhattan       Midtown  40.75362  -73.98377  Entire home/apt
2           Manhattan        Harlem  40.80902  -73.94190     Private room
3            Brooklyn  Clinton Hill  40.68514  -73.95976  Entire home/apt
4           Manhattan   East Harlem  40.79851  -73.94399  Entire home/apt

   price  minimum_nights  number_of_reviews last_review  reviews_per_month  \
0    149               1                  9  10/19/2018               0.21
1    225               1                 45   5/21/2019               0.38
2    150               3                  0         NaN                NaN
3     89               1                270    7/5/2019               4.64
4     80              10                  9  11/19/2018               0.10

   calculated_host_listings_count  availability_365
0                               6               365
1                               2               355
2                               1               365
3                               1               194
4                               1                 0
```

[3]:
```python
airbnb_data.shape
```

[3]: (9999, 15)

[4]:
```python
airbnb_data.dtypes
```

[4]:
```
name                               object
host_id                             int64
host_name                          object
neighbourhood_group                object
neighbourhood                      object
latitude                          float64
longitude                         float64
room_type                          object
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
last_review                        object
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
dtype: object
```

[5]:
```python
# Dropping non-numeric columns
numeric_data = airbnb_data.drop(columns=['name',
                                  'host_id',
                                  'host_name',
                                  'neighbourhood_group',
```

```
                                  'neighbourhood',
                                  'room_type',
                                  'last_review'])
```

[6]:
```
numeric_data.isnull().sum()
```

[6]:
```
latitude                             0
longitude                            0
price                                0
minimum_nights                       0
number_of_reviews                    0
reviews_per_month                 1322
calculated_host_listings_count       0
availability_365                     0
dtype: int64
```

[7]:
```
numeric_data.dropna(inplace=True)
```

[8]:
```
numeric_data.isnull().sum()
```

[8]:
```
latitude                          0
longitude                         0
price                             0
minimum_nights                    0
number_of_reviews                 0
reviews_per_month                 0
calculated_host_listings_count    0
availability_365                  0
dtype: int64
```

[9]:
```
numeric_data.dtypes
```

[9]:
```
latitude                          float64
longitude                         float64
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
dtype: object
```

[10]:
```
numeric_data.shape
```

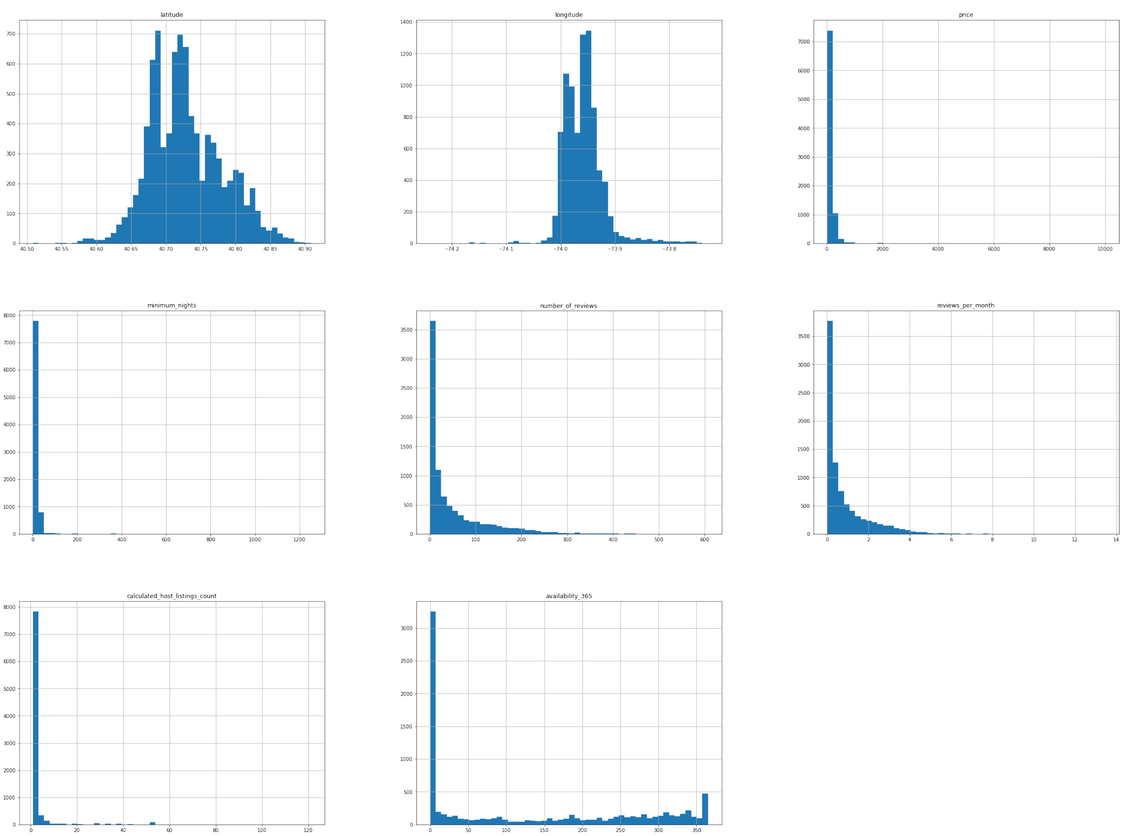
[10]:
```
(8677, 8)
```

[11]:
```
X_df = numeric_data.drop(columns='price')
X = X_df.values
```

```
Y_df = numeric_data['price']
Y = Y_df.values
X = np.concatenate((np.ones(X.shape[0]).reshape((-1, 1)), X), axis=1)
```

2 Histogram

[12]:
```
numeric_data.hist(bins=50, figsize=(40, 30))
plt.show()
```

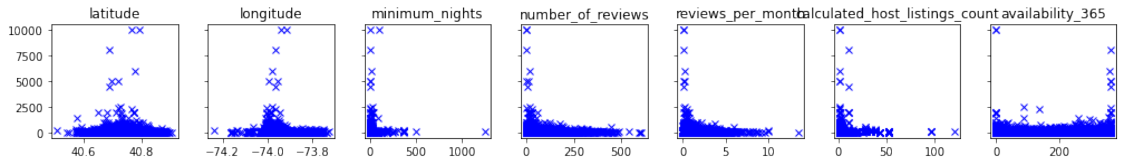


[13]:
```
column_names = list(X_df.columns)
column_names
```

[13]:
```
['latitude',
 'longitude',
 'minimum_nights',
 'number_of_reviews',
 'reviews_per_month',
 'calculated_host_listings_count',
 'availability_365']
```

3 ScatterPlot

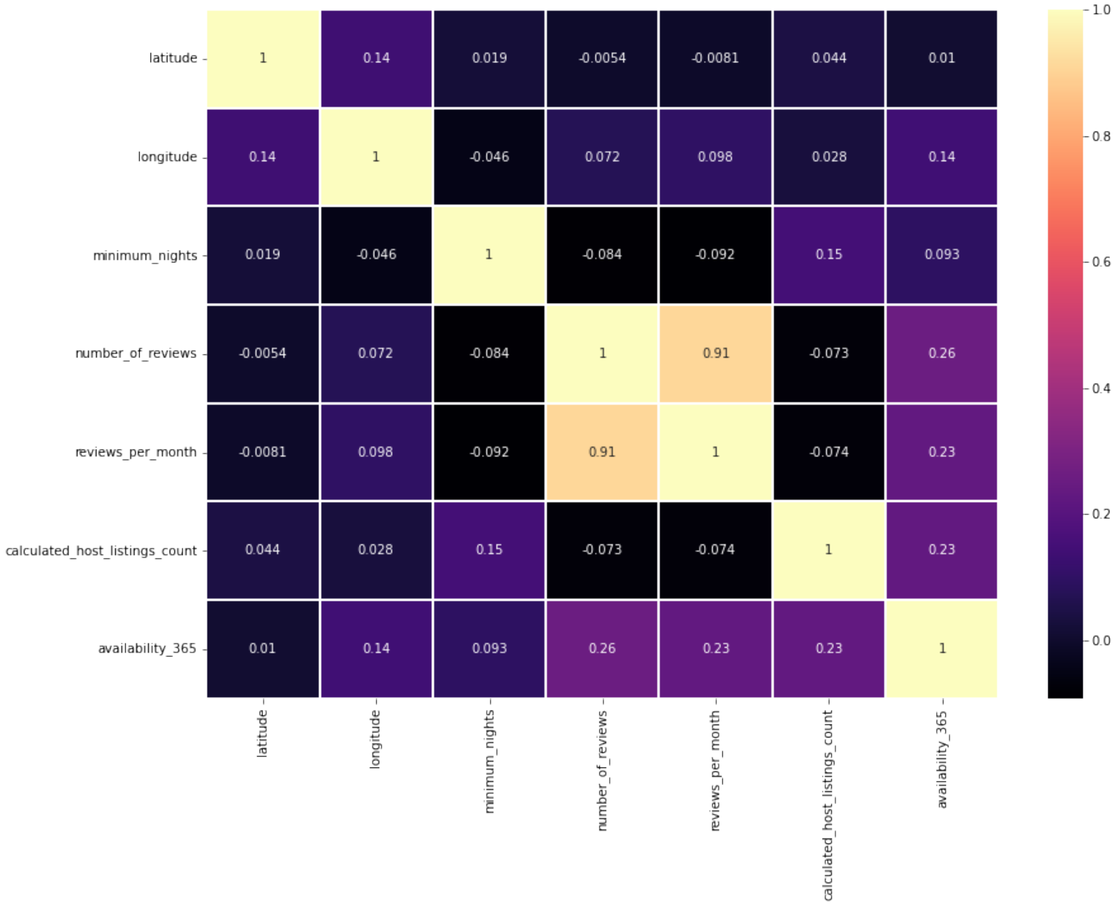
[14]:
```
f, axes = plt.subplots(1, 7, sharey=True)
f.set_size_inches(14, 2)
f.tight_layout()

axes = axes.reshape(7)

for i in range(len(column_names)):
    axes[i].plot(X[:, i + 1], Y, 'x', color='Blue')
    axes[i].set_title(column_names[i], fontsize=12)
```



4 Correlation

[15]:
```
correlation_matrix = X_df.corr()
```

[16]:
```
fig, ax = plt.subplots(figsize=(14,10))
sns.heatmap(correlation_matrix, ax=ax, linewidths=0.05,cmap="magma",annot=True)
plt.show()
```



5 Linear Regression

```
[17]: # Linear regression matrix calculation
def normal_equation(x, y, w=None):
    if w is None:
        return np.linalg.inv(x.T.dot(x)).dot(x.T).dot(y)
    else:
        return np.linalg.inv(x.T.dot(w).dot(x)).dot(x.T).dot(w).dot(y)
```

```
[18]: theta = normal_equation(X, Y)
theta
```

```
[18]: array([-6.82277031e+04,  1.28955835e+02, -8.53509860e+02,  2.45529700e-02,
        -1.34321164e-01, -4.64585556e+00, -8.99817604e-01,  1.32849443e-01])
```

6 Predict function

[19]:
```
def predict(x, theta):
    return np.dot(x, theta)
```

[20]:
```
predict(X[0, :], theta)
```

[20]: 191.10624136681963

7 RSS

[21]:
```
def rss(X, Y):
    theta = normal_equation(X, Y)
    error = X @ theta - Y
    return np.linalg.norm(error) / Y.shape[0]
```

[22]:
```
rss(X, Y)
```

[22]: 2.569281936431202