EXPERIMENT NO 5

Data Preprocessing and Random Forest Regression Using GridSearchCV

1.Importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.preprocessing as
import sklearn.model_selection as
import sklearn.ensemble
```

2. Importing the dataset

```
dataset = pd.read_csv('
/kaggle/input/listings-csv/AirbnbNewYork_NY_2018_NYC_2019.csv')
```

3. Exploring the dataset

```
dataset.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elizabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3
3	3831	Cozy Entire Floor of Brownstone	4869	LeRoyce	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1
4	5099	Large Superb Loft in central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94339	Entire home/apt	80	10

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48895 entries, 0 to 48894
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      48895 non-null  int64
 1   name                    48879 non-null  object
 2   host_id                 48895 non-null  int64
 3   host_name               48874 non-null  object
 4   neighbourhood_group     48895 non-null  object
 5   neighbourhood           48895 non-null  object
 6   latitude                48895 non-null  float64
 7   longitude               48895 non-null  float64
 8   room_type               48895 non-null  object
 9   price                   48895 non-null  int64
 10  minimum_nights          48895 non-null  int64
 11  number_of_reviews       48895 non-null  int64
 12  last_review             38843 non-null  object
 13  reviews_per_month       38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365        48895 non-null  int64
dtypes: float64(5), int64(9), object(2)
memory usage: 6.0+ MB
```

4.Checking for null values and duplicate data

```
dataset.isnull().sum()
```

```
id                              0
name                           16
host_id                         0
host_name                      21
neighbourhood_group             0
neighbourhood                   0
latitude                        0
longitude                       0
room_type                       0
price                           0
minimum_nights                  0
number_of_reviews               0
last_review                 10052
reviews_per_month           10052
calculated_host_listings_count  0
availability_365                0
dtype: int64
```

```
dataset.duplicated().sum()
```

```
0
```

5.Removing unneeded attributes and null values

```
dataset.drop(["name", "host_name", "last_review"],"", axis=1,inplace=True)
```

```
dataset.fillna("reviews_per_month", 0, inplace=True)
```

```
# Checking for nulls
dataset.isnull().sum()
```

```
host_id                         0
neighbourhood_group             0
neighbourhood                   0
latitude                        0
longitude                       0
room_type                       0
price                           0
minimum_nights                  0
number_of_reviews               0
last_review                 10052
reviews_per_month           10052
calculated_host_listings_count  0
availability_365                0
dtype: int64
```

6.Viewing data statistics

```
dataset.describe()
```

	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listing
count	4.889500e+04	48895.000000	48895.000000	48895.000000	48895.000000	48895.000000	38843.000000	48895.000000
mean	6.762001e+07	40.728949	-73.952169	152.720687	7.029962	23.274466	1.090910	7.143982
std	7.861097e+07	0.054530	0.046157	240.415436	20.510550	44.549998	1.597283	32.952519
min	2.438900e+03	40.499790	-74.244420	0.000000	1.000000	0.000000	0.000000	1.000000
25%	7.822033e+06	40.690089	-73.983070	69.000000	1.000000	1.000000	0.040000	1.000000
50%	3.679509e+07	40.723070	-73.955681	106.000000	3.000000	5.000000	0.370000	1.000000
75%	1.074344e+08	40.763115	-73.936275	175.000000	5.000000	24.000000	1.580000	2.000000
max	2.743213e+08	40.913060	-73.712990	10000.000000	1250.000000	629.000000	58.180000	327.000000

7.Finding correlations between variables

```
corr_matrix = dataset.corr()
```

```
# Visualizing all correlations
fig, ax = plt.figure(figsize=(8, 8))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt= '.2f',
                 ax=ax,
                 cmap="Blues",
                 cbar=False)
plt.set_title('Correlation Matrix of Attributes')
```

```
Text(0.5, 1.0, 'Correlation Matrix of Attributes')
```



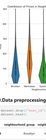
```
# Finding correlation between the price and neighbourhood variables
corr_matrix["price"].sort_values(ascending=False)
```

```
price                           1.000000
availability_365                0.083883
minimum_nights                  0.042799
calculated_host_listings_count  0.030608
host_id                         0.003192
latitude                        0.033939
longitude                       -0.150019
reviews_per_month               -0.030360
number_of_reviews               -0.046954
dtype: float64
```

8.Visualizing the data

```
# Bar chart of listings in each neighbourhood group
fig,figsize=(12,6)
ax = sns.countplot(x="neighbourhood_group",
ax = ax.set_xlabel("Neighbourhood group",
                   size=16,weight="bold",
                   labelpad=20)
ax.set_ylabel("Number of listings",
              size=16,weight="bold",
              labelpad=20)
ax.set_title("Number of Listings in Each Neighbourhood Group",
             size=20,weight="bold",
             pad=20)
ax.set_xticklabels(ax.get_xticklabels(),
                   size=12,weight="bold")
ax.set_yticklabels(ax.get_yticklabels(),
                   size=12,weight="bold")
```

```
[Text(0, 0.0, '0.0'),
 Text(0, 0.2, '0.2'),
 Text(0, 0.4, '0.4'),
 Text(0, 0.6, '0.6'),
 Text(0, 0.8, '0.8'),
 Text(0, 1.0, '1.0')]
```




```
dataset["neighbourhood_group".value_counts()
```

```
Manhattan     21661
Brooklyn      20104
Queens         5666
Bronx          1091
Staten Island   373
Name: Neighbourhood_group, dtype: int64
```

```
# Bar chart of listings of room types
fig,figsize=(12,6)
ax= sns.stripplot("neighbourhood_group",
ax = ax.set_xlabel("Neighbourhood Group",
                   size=16,weight="bold",
                   labelpad=20)
ax.set_ylabel("Room Type",
              size=16,weight="bold",
              labelpad=20)
ax.set_title("Room Types In Each Neighbourhood Group")
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arg
uments without explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
[Text(0.0, 0.0, 'Room Types In Each Neighbourhood Group')]
```

```
# Distribution of price in neighbourhood groups
fig,figsize=(12,6)
ax= sns.stripplot("neighbourhood_group",
                  "price",
                  data=dataset,
                  jitter=True,
ax.set_xlabel("Neighbourhood Group",
              size=16,weight="bold",
              labelpad=20)
ax.set_ylabel("Price",
              size=16,weight="bold",
              labelpad=20)
ax.set_title("Distribution of Prices In Neighbourhood Groups (< 2500)")
plt.ylim(0, 2500)
```

```
(0, 2500.0)
```

9.Data preprocessing

```
dataset.drop(["id", "host_id"], axis=1, inplace=True)
dataset.head()
```

	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calc
0	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9	0.21	
1	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45	0.38	
2	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0	0.00	
3	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270	4.64	
4	Manhattan	East Harlem	40.79851	-73.94339	Entire home/apt	80	10	9	0.10	

10.Encoding categorical data

```
# Encoding categorical variables
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

label_encoder = LabelEncoder()
dataset['neighbourhood_group'] = label_encoder.fit_transform(dataset['neighbourhood_group'])
dataset['neighbourhood'] = label_encoder.fit_transform(dataset['neighbourhood'])
dataset['room_type'] = label_encoder.fit_transform(dataset['room_type'])
dataset.head()
```

	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calc
0	1	110	40.64749	-73.97237	1	149	1	9	0.21	
1	2	127	40.75362	-73.98377	0	225	1	45	0.38	
2	2	74	40.80902	-73.94190	1	150	3	0	0.00	
3	1	47	40.68514	-73.95976	0	89	1	270	4.64	
4	2	62	40.79851	-73.94339	0	80	10	9	0.10	

```
# Dummy variables
dummy = pd.get_dummies(dataset)
print("Dummy Data Frame: ", dummy.shape)
print(dummy.sample(5))
```

```
Dummy Data Frame:  (48895, 226)
       neighbourhood_group  neighbourhood  latitude  longitude  room_type  price  minimum_nights  number_of_reviews  reviews_per_month  calc
16                       1             16  40.71344  -73.94039          1    150               3                  0               0.00     1
10                       1             61  40.67402  -73.94976          1    199               4                  3               0.67     1
9                        1             60  40.67853  -73.94735          1    105               1                 74               0.59     1
7                        1             58  40.67220  -73.94610          1     65               7                 49               0.40     1
11                       1             67  40.68088  -73.94755          1     65               2                  2               0.14     1

       host_id_2539  host_id_2595  host_id_3647  host_id_3831  host_id_5099  ...  last_review_2018-05-27  last_review_2018-05-28  last_review_2018-05-29  last_review_2018-06-19  last_review_2018-06-20
16             0             0             0             0             0  ...                       0                       0                       0                       0                       0
10             0             0             0             0             0  ...                       0                       0                       0                       0                       0
9              0             0             0             0             0  ...                       0                       0                       0                       0                       0
7              0             0             0             0             0  ...                       0                       0                       0                       0                       0
11             0             0             0             0             0  ...                       0                       0                       0                       0                       0

       last_review_2019-03-04  last_review_2019-03-05  last_review_2019-03-06  last_review_2019-03-07  last_review_2019-03-08
16                          0                       0                       0                       0                       0
10                          0                       0                       0                       0                       0
9                           0                       0                       0                       0                       0
7                           0                       0                       0                       0                       0
11                          0                       0                       0                       0                       0

[5 rows x 226 columns]
```

11.Splitting into Test and Training Set

```
df.size
```

```
2204020
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df,
                                                    df,
                                                    test_size=0.3,
                                                    random_state=1)
```

12.Feature Scaling

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

13.Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn.model_selection import cross_val_score

forest_reg = RandomForestRegressor()
forest_reg.fit(X_train, y_train)
y_pred = forest_reg.predict(X_test)
forest_mse = metrics.mse(y_test, y_pred)
forest_mse
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_regression.py:251:FutureWarning:Mean Absolute Error, Mean Squared Error, and Median Absolute Error metrics are deprecated in 1.0 and will be removed in 1.2.
Use mean_absolute_error, mean_squared_error, and median_absolute_error instead.
  FutureWarning:
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_regression.py:1100:FutureWarning:Function mean_squared_error is deprecated; Use error=mean_squared_error(*y_true, *y_pred, sample_weight=sample_weight, multioutput=multioutput) instead.
  FutureWarning:
```

```
0.0
```

```
# Splitting the data into independent and dependent variables
X = dataset.iloc[:, dataset.columns != 'price'].values
y = dataset.iloc[:, 3].values
from sklearn.ml.ts ...
```

```
X_labelencoder = LabelEncoder()
X[:, 1] = X_labelencoder_X[:, 1]
X[:, 2] = X_labelencoder_X[:, 2]
X[:, 3] = X_labelencoder_X[:, 3]
X[:, 4] = X_labelencoder_X[:, 4]
X[:, 5] = X_labelencoder_X[:, 5]
X_labelencoder = pd.DataFrame(X)
```

```
X_labelencoder.head()
```

	0	1	2	3	4	5	6	7	8
0	1	110	40.64749	-73.97237	1	149	1	9	0.21
1	2	127	40.75362	-73.98377	0	225	1	45	0.38
2	2	74	40.80902	-73.94190	1	150	3	0	0.00
3	1	47	40.68514	-73.95976	0	89	1	270	4.64
4	2	62	40.79851	-73.94339	0	80	10	9	0.10

```
# Dummy variables
dummy = pd.get_dummies(X_labelencoder)
print("Dummy Data Frame: ", dummy.shape)
print(dummy.sample(5))
```

```
Dummy Data Frame:  (48895, 226)
       0  1  2  3  4  5  6  7  8  9  ...  216  217  218  219  220  221  222  223  224  225
16     1  3  0  1  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0    0    0
10     1  4  0  1  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0    0    0
9      1  4  0  1  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0    0    0
7      1  4  0  1  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0    0    0
11     1  4  0  1  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0    0    0

[5 rows x 226 columns]
```

14.Splitting into Test and Training Set

```
df.size
```

```
2204020
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df,
                                                    df,
                                                    test_size=0.3,
                                                    random_state=1)
```

15.Feature Scaling

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

16.Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn.model_selection import cross_val_score

forest_reg = RandomForestRegressor()
forest_reg.fit(X_train, y_train)
y_pred = forest_reg.predict(X_test)
forest_mse = metrics.mse(y_test, y_pred)
forest_mse
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_regression.py:251:FutureWarning:Mean Absolute Error, Mean Squared Error, and Median Absolute Error metrics are deprecated in 1.0 and will be removed in 1.2.
Use mean_absolute_error, mean_squared_error, and median_absolute_error instead.
  FutureWarning:
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_regression.py:1100:FutureWarning:Function mean_squared_error is deprecated; Use error=mean_squared_error(*y_true, *y_pred, sample_weight=sample_weight, multioutput=multioutput) instead.
  FutureWarning:
```

```
0.0
```

```
# Splitting the data into independent and dependent variables
X = dataset.iloc[:, dataset.columns != 'price'].values
y = dataset.iloc[:, 3].values
# X.shape, y.shape
```

17.Encoding categorical data

```
# Encoding categorical variables
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.compose import ColumnTransformer

label_encoder = LabelEncoder()
dataset['neighbourhood_group'] = label_encoder.fit_transform(dataset['neighbourhood_group'])
dataset['neighbourhood'] = label_encoder.fit_transform(dataset['neighbourhood'])
dataset['room_type'] = label_encoder.fit_transform(dataset['room_type'])
dataset.head()
```

	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calc
0	1	110	40.64749	-73.97237	1	149	1	9	0.21	
1	2	127	40.75362	-73.98377	0	225	1	45	0.38	
2	2	74	40.80902	-73.94190	1	150	3	0	0.00	
3	1	47	40.68514	-73.95976	0	89	1	270	4.64	
4	2	62	40.79851	-73.94339	0	80	10	9	0.10	

```
# Dummy variables
dummy = pd.get_dummies(dataset)
print("Dummy Data Frame: ", dummy.shape)
print(dummy.sample(5))
```

```
Dummy Data Frame:  (48895, 226)
       neighbourhood_group  neighbourhood  latitude  longitude  room_type  price  minimum_nights  number_of_reviews  reviews_per_month  calc
16                       1             16  40.71344  -73.94039          1    150               3                  0               0.00     1
10                       1             61  40.67402  -73.94976          1    199               4                  3               0.67     1
9                        1             60  40.67853  -73.94735          1    105               1                 74               0.59     1
7                        1             58  40.67220  -73.94610          1     65               7                 49               0.40     1
11                       1             67  40.68088  -73.94755          1     65               2                  2               0.14     1

       host_id_2539  host_id_2595  host_id_3647  host_id_3831  host_id_5099  ...  last_review_2018-05-27  last_review_2018-05-28  last_review_2018-05-29  last_review_2018-06-19  last_review_2018-06-20
16             0             0             0             0             0  ...                       0                       0                       0                       0                       0
10             0             0             0             0             0  ...                       0                       0                       0                       0                       0
9              0             0             0             0             0  ...                       0                       0                       0                       0                       0
7              0             0             0             0             0  ...                       0                       0                       0                       0                       0
11             0             0             0             0             0  ...                       0                       0                       0                       0                       0

       last_review_2019-03-04  last_review_2019-03-05  last_review_2019-03-06  last_review_2019-03-07  last_review_2019-03-08
16                          0                       0                       0                       0                       0
10                          0                       0                       0                       0                       0
9                           0                       0                       0                       0                       0
7                           0                       0                       0                       0                       0
11                          0                       0                       0                       0                       0

[5 rows x 226 columns]
```

18.Grid Search

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [3, 10, 30],
    'max_features': [2, 4, 6, 8]
},
    'bootstrap': [False],
    'n_estimators': [3, 10],
    'max_features': [2, 3, 4]
}

forest_reg = RandomForestRegressor()

grid_search = GridSearchCV(forest_reg,
                           param_grid,
                           cv=5,
                           scoring='neg_mean_squared_error')

grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestRegressor(bootstrap=True,
                                             ccp_alpha=0.0,
                                             criterion='mse',
                                             max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             max_samples=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators=100,
                                             n_jobs=None,
                                             oob_score=False,
                                             random_state=None,
                                             verbose=0,
                                             warm_start=False),
             iid=False, n_jobs=None,
             param_grid=[{'bootstrap': [False],
                          'max_features': [2, 3, 4],
                          'n_estimators': [3, 10]},
                         {'max_features': [2, 4, 6, 8],
                          'n_estimators': [3, 10, 30]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='neg_mean_squared_error', verbose=0)
```

```
grid_search.cv_results_
```

```
{'mean_fit_time': array([0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 
```