ImplementMLProjectPlan

August 11, 2023

1 Lab 8: Implement Your Machine Learning Project Plan

In this lab assignment, you will implement the machine learning project plan you created in the written assignment. You will:

1. Load your data set and save it to a Pandas DataFrame.
2. Perform exploratory data analysis on your data to determine which feature engineering and data preparation techniques you will use.
3. Prepare your data for your model and create features and a label.
4. Fit your model to the training data and evaluate your model.
5. Improve your model by performing model selection and/or feature selection techniques to find best model for your problem.

1.0.1 Import Packages

Before you get started, import a few packages.

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

Task: In the code cell below, import additional packages that you have used in this course that you will need for this task.

[2]:
```
import scipy.stats as stats
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
```

1.1 Part 1: Load the Data Set

You have chosen to work with one of four data sets. The data sets are located in a folder named "data." The file names of the three data sets are as follows:

- The "adult" data set that contains Census information from 1994 is located in file `adultData.csv`
- The airbnb NYC "listings" data set is located in file `airbnbListingsData.csv`
- The World Happiness Report (WHR) data set is located in file `WHR2018Chapter2OnlineData.csv`
- The book review data set is located in file `bookReviewsData.csv`

Task: In the code cell below, use the same method you have been using to load your data using `pd.read_csv()` and save it to DataFrame `df`.

[3]:
```
filename = os.path.join(os.getcwd(), "data", "airbnbListingsData.csv")
df = pd.read_csv(filename, header=0)
df.head()
```

[3]:
```
                                                name  \
0                                Skylit Midtown Castle
1  Whole flr w/private bdrm, bath & kitchen(pls r...
2           Spacious Brooklyn Duplex, Patio + Garden
3                   Large Furnished Room Near B'way
4                 Cozy Clean Guest Room - Family Apt

                                         description  \
0  Beautiful, spacious skylit studio in the heart...
1  Enjoy 500 s.f. top floor in 1899 brownstone, w...
2  We welcome you to stay in our lovely 2 br dupl...
3  Please dont expect the luxury here just a bas...
4  Our best guests are seeking a safe, clean, spa...

                              neighborhood_overview    host_name  \
0  Centrally located in the heart of Manhattan ju...    Jennifer
1  Just the right mix of urban center and local n...  LisaRoxanne
2                                                NaN      Rebecca
3    Theater district, many restaurants around here.    Shunichi
4  Our neighborhood is full of restaurants and ca...    MaryEllen

                     host_location  \
0  New York, New York, United States
1  New York, New York, United States
2  Brooklyn, New York, United States
3  New York, New York, United States
4  New York, New York, United States

                                         host_about  host_response_rate  \
0  A New Yorker since 2000! My passion is creatin...                0.80
1  Laid-back Native New Yorker (formerly bi-coast...                0.09
2  Rebecca is an artist/designer, and Henoch is i...                1.00
3  I used to work for a financial industry but no...                1.00
4  Welcome to family life with my oldest two away...                 NaN
```

```
   host_acceptance_rate  host_is_superhost  host_listings_count  ...  \
0                  0.17               True                  8.0  ...
1                  0.69               True                  1.0  ...
2                  0.25               True                  1.0  ...
3                  1.00               True                  1.0  ...
4                   NaN               True                  1.0  ...

   review_scores_communication  review_scores_location  review_scores_value  \
0                         4.79                    4.86                 4.41
1                         4.80                    4.71                 4.64
2                         5.00                    4.50                 5.00
3                         4.42                    4.87                 4.36
4                         4.95                    4.94                 4.92

  instant_bookable calculated_host_listings_count  \
0            False                              3
1            False                              1
2            False                              1
3            False                              1
4            False                              1

   calculated_host_listings_count_entire_homes  \
0                                            3
1                                            1
2                                            1
3                                            0
4                                            0

   calculated_host_listings_count_private_rooms  \
0                                             0
1                                             0
2                                             0
3                                             1
4                                             1

   calculated_host_listings_count_shared_rooms  reviews_per_month  \
0                                            0               0.33
1                                            0               4.86
2                                            0               0.02
3                                            0               3.68
4                                            0               0.87

  n_host_verifications
0                    9
1                    6
2                    3
3                    4
```

```
4                    7

[5 rows x 50 columns]
```

[4]:
```
df.shape
```

[4]:
```
(28022, 50)
```

1.2 Part 2: Exploratory Data Analysis

The next step is to inspect and analyze your data set with your machine learning problem and project plan in mind.

This step will help you determine data preparation and feature engineering techniques you will need to apply to your data to build a balanced modeling data set for your problem and model. These data preparation techniques may include: * addressing missingness, such as replacing missing values with means * renaming features and labels * finding and replacing outliers * performing winsorization if needed * performing one-hot encoding on categorical features * performing vectorization for an NLP problem * addressing class imbalance in your data sample to promote fair AI

Think of the different techniques you have used to inspect and analyze your data in this course. These include using Pandas to apply data filters, using the Pandas `describe()` method to get insight into key statistics for each column, using the Pandas `dtypes` property to inspect the data type of each column, and using Matplotlib and Seaborn to detect outliers and visualize relationships between features and labels. If you are working on a classification problem, use techniques you have learned to determine if there is class imbalance.

Task: Use the techniques you have learned in this course to inspect and analyze your data.

Note: You can add code cells if needed by going to the Insert menu and clicking on Insert Cell Below in the drop-drown menu.

[5]:
```
df.dtypes
```

[5]:
```
name                                              object
description                                       object
neighborhood_overview                             object
host_name                                         object
host_location                                     object
host_about                                        object
host_response_rate                               float64
host_acceptance_rate                             float64
host_is_superhost                                   bool
host_listings_count                              float64
host_total_listings_count                        float64
host_has_profile_pic                                bool
host_identity_verified                              bool
neighbourhood_group_cleansed                      object
room_type                                         object
accommodates                                       int64
bathrooms                                        float64
bedrooms                                         float64
```

```
beds                                            float64
amenities                                        object
price                                           float64
minimum_nights                                    int64
maximum_nights                                    int64
minimum_minimum_nights                          float64
maximum_minimum_nights                          float64
minimum_maximum_nights                          float64
maximum_maximum_nights                          float64
minimum_nights_avg_ntm                          float64
maximum_nights_avg_ntm                          float64
has_availability                                   bool
availability_30                                   int64
availability_60                                   int64
availability_90                                   int64
availability_365                                  int64
number_of_reviews                                 int64
number_of_reviews_ltm                             int64
number_of_reviews_l30d                            int64
review_scores_rating                            float64
review_scores_cleanliness                       float64
review_scores_checkin                           float64
review_scores_communication                     float64
review_scores_location                          float64
review_scores_value                             float64
instant_bookable                                   bool
calculated_host_listings_count                    int64
calculated_host_listings_count_entire_homes       int64
calculated_host_listings_count_private_rooms      int64
calculated_host_listings_count_shared_rooms       int64
reviews_per_month                               float64
n_host_verifications                              int64
dtype: object
```

[6]:
```
df.describe()
```

[6]:
```
       host_response_rate  host_acceptance_rate  host_listings_count  \
count        16179.000000          16909.000000         28022.000000
mean             0.906901              0.791953            14.554778
std              0.227282              0.276732           120.721287
min              0.000000              0.000000             0.000000
25%              0.940000              0.680000             1.000000
50%              1.000000              0.910000             1.000000
75%              1.000000              1.000000             3.000000
max              1.000000              1.000000          3387.000000

       host_total_listings_count  accommodates     bathrooms      bedrooms  \
count               28022.000000  28022.000000  28022.000000  25104.000000
```

```
mean                  14.554778      2.874491      1.142174      1.329708
std                  120.721287      1.860251      0.421132      0.700726
min                    0.000000      1.000000      0.000000      1.000000
25%                    1.000000      2.000000      1.000000      1.000000
50%                    1.000000      2.000000      1.000000      1.000000
75%                    3.000000      4.000000      1.000000      1.000000
max                 3387.000000     16.000000      8.000000     12.000000

               beds         price  minimum_nights  ...  review_scores_checkin  \
count  26668.000000  28022.000000    28022.000000  ...           28022.000000
mean       1.629556    154.228749       18.689387  ...               4.814300
std        1.097104    140.816605       25.569151  ...               0.438603
min        1.000000     29.000000        1.000000  ...               0.000000
25%        1.000000     70.000000        2.000000  ...               4.810000
50%        1.000000    115.000000       30.000000  ...               4.960000
75%        2.000000    180.000000       30.000000  ...               5.000000
max       21.000000   1000.000000     1250.000000  ...               5.000000

       review_scores_communication  review_scores_location  \
count                 28022.000000            28022.000000
mean                      4.808041                4.750393
std                       0.464585                0.415717
min                       0.000000                0.000000
25%                       4.810000                4.670000
50%                       4.970000                4.880000
75%                       5.000000                5.000000
max                       5.000000                5.000000

       review_scores_value  calculated_host_listings_count  \
count         28022.000000                    28022.000000
mean              4.647670                        9.581900
std               0.518023                       32.227523
min               0.000000                        1.000000
25%               4.550000                        1.000000
50%               4.780000                        1.000000
75%               5.000000                        3.000000
max               5.000000                      421.000000

       calculated_host_listings_count_entire_homes  \
count                                 28022.000000
mean                                      5.562986
std                                      26.121426
min                                       0.000000
25%                                       0.000000
50%                                       1.000000
75%                                       1.000000
max                                     308.000000
```

```
       calculated_host_listings_count_private_rooms  \
count                                      28022.000000
mean                                           3.902077
std                                           17.972386
min                                            0.000000
25%                                            0.000000
50%                                            0.000000
75%                                            1.000000
max                                          359.000000

       calculated_host_listings_count_shared_rooms  reviews_per_month  \
count                                     28022.000000       28022.000000
mean                                          0.048283           1.758325
std                                           0.442459           4.446143
min                                           0.000000           0.010000
25%                                           0.000000           0.130000
50%                                           0.000000           0.510000
75%                                           0.000000           1.830000
max                                           8.000000         141.000000

       n_host_verifications
count          28022.000000
mean               5.169510
std                2.028497
min                1.000000
25%                4.000000
50%                5.000000
75%                7.000000
max               13.000000

[8 rows x 36 columns]
```

[7]:
```
# Transforming the 'object' categorical features into numerical boolean values␣
 ↪using one-hot encoding:

to_encode = list(df.select_dtypes(include=['object']).columns)
df[to_encode].nunique()
```

[7]:
```
name                            27386
description                     25952
neighborhood_overview           15800
host_name                        7566
host_location                    1364
host_about                      11962
neighbourhood_group_cleansed        5
room_type                           4
amenities                       25020
```

```
dtype: int64
```

[8]:
```
# Taking a look at the unique values, I can see that all columns with the␣
↪exception of 'host_location', 'neighbourhood_group_cleansed',
# and 'room_type' are descriptive and therefore require NLP, those columns will␣
↪be dropped:

to_drop = ['name', 'description', 'neighborhood_overview', 'host_name',␣
↪'host_about', 'amenities']
df.drop(columns = to_drop, inplace = True)
```

[9]:
```
# Performing one-hot-encoding on 'host_location':

top_50_host_location = list(df['host_location'].value_counts().head(50).index)
```

[10]:
```
# Using a for loop that loops through every value in top_50_host_location and␣
↪creates one-hot encoded columns,
# titled 'host_location + '_' +  < host location value > '.

for value in top_50_host_location:
    df['host_location'+ '_'+ value] = np.where(df['host_location']==value,1,0)
```

[11]:
```
# Dropping the original, multi-valued host_location column from the DataFrame␣
↪df.
# Removing 'host_location' from the to_encode list.

df.drop(columns = 'host_location', inplace = True)
to_encode.remove('host_location')
```

[12]:
```
to_drop = ['name', 'description', 'neighborhood_overview', 'host_name',␣
↪'host_about', 'amenities']
for item in to_drop:
    to_encode.remove(item)

to_encode
```

[12]:
```
['neighbourhood_group_cleansed', 'room_type']
```

[13]:
```
# Performing one-hot-encoding the rest of the columns:

for value in to_encode:
    temp_df = pd.get_dummies(df[value], prefix = value + '_')
    df = df.join(temp_df)
```

[14]:
```
df.drop(columns = to_encode, inplace = True)
df.head()
```

[14]:
```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0                0.80                  0.17               True
1                0.09                  0.69               True
2                1.00                  0.25               True
```

```
3              1.00                  1.00               True
4               NaN                   NaN               True

   host_listings_count  host_total_listings_count  host_has_profile_pic  \
0                  8.0                        8.0                  True
1                  1.0                        1.0                  True
2                  1.0                        1.0                  True
3                  1.0                        1.0                  True
4                  1.0                        1.0                  True

   host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
0                    True             1        1.0       NaN  ...
1                    True             3        1.0       1.0  ...
2                    True             4        1.5       2.0  ...
3                    True             2        1.0       1.0  ...
4                    True             1        1.0       1.0  ...

   host_location_Princeton, New Jersey, United States  \
0                                                  0
1                                                  0
2                                                  0
3                                                  0
4                                                  0

   neighbourhood_group_cleansed__Bronx  \
0                                    0
1                                    0
2                                    0
3                                    0
4                                    0

   neighbourhood_group_cleansed__Brooklyn  \
0                                       0
1                                       1
2                                       1
3                                       0
4                                       0

   neighbourhood_group_cleansed__Manhattan  \
0                                        1
1                                        0
2                                        0
3                                        1
4                                        1

   neighbourhood_group_cleansed__Queens  \
0                                     0
```

```
1                                        0
2                                        0
3                                        0
4                                        0

   neighbourhood_group_cleansed__Staten Island  room_type__Entire home/apt  \
0                                            0                           1
1                                            0                           1
2                                            0                           1
3                                            0                           0
4                                            0                           0

   room_type__Hotel room  room_type__Private room  room_type__Shared room
0                      0                        0                       0
1                      0                        0                       0
2                      0                        0                       0
3                      0                        1                       0
4                      0                        1                       0

[5 rows x 100 columns]
```

[15]:
```
# I now need to see if I have any missing data:

df.isnull().values.any()
```

[15]:
```
True
```

[16]:
```
df.isnull().head()
```

[16]:
```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0               False                 False              False
1               False                 False              False
2               False                 False              False
3               False                 False              False
4                True                  True              False

   host_listings_count  host_total_listings_count  host_has_profile_pic  \
0                False                      False                 False
1                False                      False                 False
2                False                      False                 False
3                False                      False                 False
4                False                      False                 False

   host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
0                   False         False      False      True  ...
1                   False         False      False     False  ...
2                   False         False      False     False  ...
3                   False         False      False     False  ...
4                   False         False      False     False  ...
```

```
   host_location_Princeton, New Jersey, United States  \
0                                              False
1                                              False
2                                              False
3                                              False
4                                              False

   neighbourhood_group_cleansed__Bronx  \
0                               False
1                               False
2                               False
3                               False
4                               False

   neighbourhood_group_cleansed__Brooklyn  \
0                                  False
1                                  False
2                                  False
3                                  False
4                                  False

   neighbourhood_group_cleansed__Manhattan  \
0                                   False
1                                   False
2                                   False
3                                   False
4                                   False

   neighbourhood_group_cleansed__Queens  \
0                                False
1                                False
2                                False
3                                False
4                                False

   neighbourhood_group_cleansed__Staten Island  room_type__Entire home/apt  \
0                                       False                       False
1                                       False                       False
2                                       False                       False
3                                       False                       False
4                                       False                       False

   room_type__Hotel room  room_type__Private room  room_type__Shared room
0                 False                    False                   False
1                 False                    False                   False
2                 False                    False                   False
```

```
3                  False                    False                   False
4                  False                    False                   False

[5 rows x 100 columns]
```

[17]:
```
nan_count = np.sum(df.isnull(), axis = 0)
nan_count
```

[17]:
```
host_response_rate                               11843
host_acceptance_rate                             11113
host_is_superhost                                    0
host_listings_count                                  0
host_total_listings_count                            0
                                                 ...
neighbourhood_group_cleansed__Staten Island          0
room_type__Entire home/apt                           0
room_type__Hotel room                                0
room_type__Private room                              0
room_type__Shared room                               0
Length: 100, dtype: int64
```

[18]:
```
# Since not all of the columns have missing data, I will create a condition for␣
 ↪them:

condition = nan_count != 0
nan_col_names = nan_count[condition].index # to get the column names
nan_cols = list(nan_col_names) # convert column names to python list
nan_cols
```

[18]:
```
['host_response_rate', 'host_acceptance_rate', 'bedrooms', 'beds']
```

[19]:
```
# I want to fill the columns with a dtype of float64:

nan_col_types = df[nan_cols].dtypes
nan_col_types
```

[19]:
```
host_response_rate      float64
host_acceptance_rate    float64
bedrooms                float64
beds                    float64
dtype: object
```

[20]:
```
# Creating dummy variables for missing values:

df['host_response_rate_na'] = df['host_response_rate'].isnull()
df['host_acceptance_rate_na'] = df['host_acceptance_rate'].isnull()
df['bedrooms_na'] = df['bedrooms'].isnull()
df['beds_na'] = df['beds'].isnull()
```

[21]:
```
df.head()
```

[21]:

```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0                0.80                  0.17               True
1                0.09                  0.69               True
2                1.00                  0.25               True
3                1.00                  1.00               True
4                 NaN                   NaN               True

   host_listings_count  host_total_listings_count  host_has_profile_pic  \
0                  8.0                        8.0                  True
1                  1.0                        1.0                  True
2                  1.0                        1.0                  True
3                  1.0                        1.0                  True
4                  1.0                        1.0                  True

   host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
0                    True             1        1.0       NaN  ...
1                    True             3        1.0       1.0  ...
2                    True             4        1.5       2.0  ...
3                    True             2        1.0       1.0  ...
4                    True             1        1.0       1.0  ...

   neighbourhood_group_cleansed__Queens  \
0                                     0
1                                     0
2                                     0
3                                     0
4                                     0

   neighbourhood_group_cleansed__Staten Island  room_type__Entire home/apt  \
0                                            0                           1
1                                            0                           1
2                                            0                           1
3                                            0                           0
4                                            0                           0

   room_type__Hotel room  room_type__Private room  room_type__Shared room  \
0                      0                        0                       0
1                      0                        0                       0
2                      0                        0                       0
3                      0                        1                       0
4                      0                        1                       0

   host_response_rate_na  host_acceptance_rate_na  bedrooms_na  beds_na
0                  False                    False         True    False
1                  False                    False        False    False
2                  False                    False        False    False
3                  False                    False        False    False
```

```
4                   True                        True          False     False

[5 rows x 104 columns]
```

[22]:
```
# Filling the values for missing 'host_response_rate' column:

mean_host_response_rate = df['host_response_rate'].mean()
df['host_response_rate'].fillna(value=mean_host_response_rate, inplace = True)
```

[23]:
```
# Filling the values for missing 'host_acceptance_rate' column:

mean_host_acceptance_rate = df['host_acceptance_rate'].mean()
df['host_acceptance_rate'].fillna(value=mean_host_acceptance_rate, inplace =␣
 ↪True)
```

[24]:
```
# Filling the values for missing 'bedrooms' column:

mean_bedrooms = df['bedrooms'].mean()
df['bedrooms'].fillna(value=mean_bedrooms, inplace = True)
```

[25]:
```
# Filling the values for missing 'beds' column:

mean_beds = df['beds'].mean()
df['beds'].fillna(value=mean_beds, inplace = True)
```

[26]:
```
# Checking that I successfully replaced all null values

print(np.sum(df['host_response_rate'].isnull(), axis = 0))
print(np.sum(df['host_acceptance_rate'].isnull(), axis = 0))
print(np.sum(df['bedrooms'].isnull(), axis = 0))
print(np.sum(df['beds'].isnull(), axis = 0))
```

```
0
0
0
0
```

[27]:
```
# Making sure I have no missing data:

df.isnull().values.any()
```

[27]: False

1.3 Part 3: Implement Your Project Plan

Task: Use the rest of this notebook to carry out your project plan. You will:

1. Prepare your data for your model and create features and a label.
2. Fit your model to the training data and evaluate your model.

3. Improve your model by performing model selection and/or feature selection techniques to find best model for your problem.

Add code cells below and populate the notebook with commentary, code, analyses, results, and figures as you see fit.

[28]:
```
# I now want to implement my poject plan, and predict the␣
↪'review_scores_location' by trainning various regression
# models and comparing their performances
```

[29]:
```
# to_drop are the columns I am not including in my label and the dummy␣
↪variables:

to_drop =['host_response_rate_na', 'host_acceptance_rate_na',␣
↪'bedrooms_na','beds_na']
df = df.drop(columns=to_drop)
df.head()
```

[29]:
```
   host_response_rate  host_acceptance_rate  host_is_superhost  \
0            0.800000              0.170000               True
1            0.090000              0.690000               True
2            1.000000              0.250000               True
3            1.000000              1.000000               True
4            0.906901              0.791953               True

   host_listings_count  host_total_listings_count  host_has_profile_pic  \
0                  8.0                        8.0                  True
1                  1.0                        1.0                  True
2                  1.0                        1.0                  True
3                  1.0                        1.0                  True
4                  1.0                        1.0                  True

   host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
0                    True             1        1.0  1.329708  ...
1                    True             3        1.0  1.000000  ...
2                    True             4        1.5  2.000000  ...
3                    True             2        1.0  1.000000  ...
4                    True             1        1.0  1.000000  ...

   host_location_Princeton, New Jersey, United States  \
0                                                  0
1                                                  0
2                                                  0
3                                                  0
4                                                  0

   neighbourhood_group_cleansed__Bronx  \
0                                    0
1                                    0
```

```
2                                    0
3                                    0
4                                    0

   neighbourhood_group_cleansed__Brooklyn  \
0                                       0
1                                       1
2                                       1
3                                       0
4                                       0

   neighbourhood_group_cleansed__Manhattan  \
0                                        1
1                                        0
2                                        0
3                                        1
4                                        1

   neighbourhood_group_cleansed__Queens  \
0                                     0
1                                     0
2                                     0
3                                     0
4                                     0

   neighbourhood_group_cleansed__Staten Island  room_type__Entire home/apt  \
0                                            0                           1
1                                            0                           1
2                                            0                           1
3                                            0                           0
4                                            0                           0

   room_type__Hotel room  room_type__Private room  room_type__Shared room
0                      0                        0                       0
1                      0                        0                       0
2                      0                        0                       0
3                      0                        1                       0
4                      0                        1                       0

[5 rows x 100 columns]
```

[30]:
```
# Creating my labeled examples:

y = df['review_scores_location']
X = df.drop(columns = 'review_scores_location', axis = 1)
X
```

[30]:

```
       host_response_rate  host_acceptance_rate  host_is_superhost  \
0                0.800000              0.170000               True
1                0.090000              0.690000               True
2                1.000000              0.250000               True
3                1.000000              1.000000               True
4                0.906901              0.791953               True
...                   ...                   ...                ...
28017            1.000000              1.000000               True
28018            0.910000              0.890000               True
28019            0.990000              0.990000               True
28020            0.900000              1.000000               True
28021            0.906901              0.791953               True

       host_listings_count  host_total_listings_count  host_has_profile_pic  \
0                      8.0                        8.0                  True
1                      1.0                        1.0                  True
2                      1.0                        1.0                  True
3                      1.0                        1.0                  True
4                      1.0                        1.0                  True
...                    ...                        ...                   ...
28017                  8.0                        8.0                  True
28018                  0.0                        0.0                  True
28019                  6.0                        6.0                  True
28020                  3.0                        3.0                  True
28021                  0.0                        0.0                  True

       host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
0                        True             1        1.0  1.329708  ...
1                        True             3        1.0  1.000000  ...
2                        True             4        1.5  2.000000  ...
3                        True             2        1.0  1.000000  ...
4                        True             1        1.0  1.000000  ...
...                       ...           ...        ...       ...  ...
28017                    True             2        1.0  1.000000  ...
28018                    True             6        1.0  2.000000  ...
28019                    True             2        2.0  1.000000  ...
28020                    True             3        1.0  1.000000  ...
28021                    True             1        1.0  1.000000  ...

       host_location_Princeton, New Jersey, United States  \
0                                                      0
1                                                      0
2                                                      0
3                                                      0
4                                                      0
...                                                  ...
28017                                                  0
```

```
28018                                                          0
28019                                                          0
28020                                                          0
28021                                                          0

       neighbourhood_group_cleansed__Bronx  \
0                                        0
1                                        0
2                                        0
3                                        0
4                                        0
...                                    ...
28017                                    0
28018                                    0
28019                                    0
28020                                    0
28021                                    0

       neighbourhood_group_cleansed__Brooklyn  \
0                                           0
1                                           1
2                                           1
3                                           0
4                                           0
...                                       ...
28017                                       0
28018                                       1
28019                                       1
28020                                       1
28021                                       0

       neighbourhood_group_cleansed__Manhattan  \
0                                            1
1                                            0
2                                            0
3                                            1
4                                            1
...                                        ...
28017                                        0
28018                                        0
28019                                        0
28020                                        0
28021                                        0

       neighbourhood_group_cleansed__Queens  \
0                                         0
1                                         0
```

```
2                                     0
3                                     0
4                                     0
...                                 ...
28017                                 1
28018                                 0
28019                                 0
28020                                 0
28021                                 1

       neighbourhood_group_cleansed__Staten Island  \
0                                                0
1                                                0
2                                                0
3                                                0
4                                                0
...                                            ...
28017                                            0
28018                                            0
28019                                            0
28020                                            0
28021                                            0

       room_type__Entire home/apt  room_type__Hotel room  \
0                               1                      0
1                               1                      0
2                               1                      0
3                               0                      0
4                               0                      0
...                           ...                    ...
28017                           0                      0
28018                           1                      0
28019                           0                      0
28020                           1                      0
28021                           0                      0

       room_type__Private room  room_type__Shared room
0                            0                       0
1                            0                       0
2                            0                       0
3                            1                       0
4                            1                       0
...                        ...                     ...
28017                        1                       0
28018                        0                       0
28019                        1                       0
28020                        0                       0
```

```
28021                        1                      0

[28022 rows x 99 columns]
```

[31]:
```
# Splitting labeled examples into training and test sets:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,␣
↪random_state = 1234)
X_train.head()
```

[31]:
```
       host_response_rate  host_acceptance_rate  host_is_superhost  \
16860            1.000000              0.950000               True
17993            0.906901              0.791953               True
5214             1.000000              0.980000               True
2220             0.906901              0.791953               True
16547            1.000000              0.730000               True

       host_listings_count  host_total_listings_count  host_has_profile_pic  \
16860                  0.0                        0.0                  True
17993                  0.0                        0.0                  True
5214                   3.0                        3.0                  True
2220                   1.0                        1.0                  True
16547                  2.0                        2.0                  True

       host_identity_verified  accommodates  bathrooms  bedrooms  ...  \
16860                    True             2        1.0       1.0  ...
17993                    True             1        1.0       1.0  ...
5214                     True            14        3.0       3.0  ...
2220                     True             2        1.0       1.0  ...
16547                    True             2        1.0       1.0  ...

       host_location_Princeton, New Jersey, United States  \
16860                                                  0
17993                                                  0
5214                                                   0
2220                                                   0
16547                                                  0

       neighbourhood_group_cleansed__Bronx  \
16860                                    0
17993                                    0
5214                                     0
2220                                     0
16547                                    1

       neighbourhood_group_cleansed__Brooklyn  \
16860                                       1
17993                                       0
```

```
5214                                         1
2220                                         1
16547                                        0

       neighbourhood_group_cleansed__Manhattan  \
16860                                        0
17993                                        1
5214                                         0
2220                                         0
16547                                        0

       neighbourhood_group_cleansed__Queens  \
16860                                     0
17993                                     0
5214                                      0
2220                                      0
16547                                     0

       neighbourhood_group_cleansed__Staten Island  \
16860                                            0
17993                                            0
5214                                             0
2220                                             0
16547                                            0

       room_type__Entire home/apt  room_type__Hotel room  \
16860                           0                      0
17993                           0                      0
5214                            1                      0
2220                            1                      0
16547                           0                      0

       room_type__Private room  room_type__Shared room
16860                        1                       0
17993                        1                       0
5214                         0                       0
2220                         0                       0
16547                        1                       0

[5 rows x 99 columns]
```

[32]:
```python
# Creating and fitting the LinearRegression model:

model_LR = LinearRegression()
model_LR.fit(X_train, y_train)
```

[32]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[33]:
```
# Making predictions on the test data using predict()
y_LR_pred = model_LR.predict(X_test)
```

[34]:
```
# Computing the RMSE and R2 values for the LinearRegression model:

rmse_LR = mean_squared_error(y_test, y_LR_pred, squared=False)
r2_LR = r2_score(y_test, y_LR_pred)

print('[LR] Root Mean Squared Error: {0}'.format(rmse_LR))
print('[LR] R2: {0}'.format(r2_LR))
```

```
[LR] Root Mean Squared Error: 0.3101082890860899
[LR] R2: 0.44205162858670577
```

[35]:
```
# Creating a dictionary called param_grid that contains possible hyperparameter␣
 ↪values for max_depth and
# min_samples_leaf:

param_grid = {'max_depth': [4, 8], 'min_samples_leaf': [25, 50]}
```

[36]:
```
# Creating a DecisionTreeRegressor model:

regressor_DT = DecisionTreeRegressor()
```

[37]:
```
# Running a Grid Search with 3-fold cross-validation and fitting the grid␣
 ↪search:

grid_DT = GridSearchCV(estimator = regressor_DT, param_grid = param_grid, cv =␣
 ↪3, scoring='neg_root_mean_squared_error')
grid_search_DT = grid_DT.fit(X_train, y_train)
print('Done')
```

```
Done
```

[38]:
```
# Printing the RMSE score of the best DT model using the best_score_ attribute␣
 ↪of the fitted grid:

rmse_DT = -1 * grid_search_DT.best_score_
print("[DT] RMSE for the best model is : {:.2f}".format(rmse_DT) )
```

```
[DT] RMSE for the best model is : 0.32
```

[39]:
```
# Printing the best model hyperparameters identified by the grid search:

best_params_DT = grid_search_DT.best_params_
best_params_DT
```

[39]:
```
{'max_depth': 8, 'min_samples_leaf': 50}
```

[40]:
```
# Initializing a DecisionTreeRegressor model object, but now I'm supplying the␣
↪best values of hyperparameters
# max_depth and min_samples_leaf as arguments:

model_DT = DecisionTreeRegressor(max_depth=best_params_DT['max_depth'],␣
↪min_samples_leaf= best_params_DT['min_samples_leaf'])
model_DT.fit(X_train, y_train)
```

[40]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=8,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=50, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

[41]:
```
# Using the fitted model to make predictions on the test data:

y_DT_pred = model_DT.predict(X_test)
```

[42]:
```
# Computing the RMSE and R2 scores for the DecisionTreeRegressor model:

rmse_DT = mean_squared_error(y_test, y_DT_pred, squared=False)
r2_DT = r2_score(y_test, y_DT_pred)

print('[DT] Root Mean Squared Error: {0}'.format(rmse_DT))
print('[DT] R2: {0}'.format(r2_DT))
```

```
[DT] Root Mean Squared Error: 0.31570007897144337
[DT] R2: 0.4217486631801226
```

[43]:
```
# Creating a gradient boosted decision tree model using max_depth = 3 and␣
↪n_estimators = 300:

model_GBDT = GradientBoostingRegressor(max_depth=3, n_estimators=300)
model_GBDT.fit(X_train, y_train)
print('Done')
```

```
Done
```

[44]:
```
# Using the fitted model to make predictions on the test data:

y_GBDT_pred = model_GBDT.predict(X_test)
```

[45]:
```
# Computing the RMSE and R2 scores for the GBDT model:

rmse_DBDT = mean_squared_error(y_test, y_GBDT_pred, squared=False)
r2_GBDT = r2_score(y_test, y_GBDT_pred)
```

```
print('[GBDT] Root Mean Squared Error: {0}'.format(rmse_DBDT))
print('[GBDT] R2: {0}'.format(r2_GBDT))
```

```
[GBDT] Root Mean Squared Error: 0.3012938464075124
[GBDT] R2: 0.47331883448493695
```

[46]:
```
# Creating a RandomForestRegressor model using max_depth = 32 and n_estimators␣
↪= 300:

model_RF = RandomForestRegressor(max_depth=32, n_estimators=300)
model_RF.fit(X_train, y_train)
print('Done')
```

```
Done
```

[47]:
```
# Using the fitted model to make predictions on the test data:

y_RF_pred = model_RF.predict(X_test)
```

[48]:
```
# Computing the RMSE and R2 scores for the RandomForestRegressor model:

rmse_RF = mean_squared_error(y_test, y_RF_pred, squared=False)
r2_RF = r2_score(y_test, y_RF_pred)

print('[RF] Root Mean Squared Error: {0}'.format(rmse_RF))
print('[RF] R2: {0}'.format(r2_RF))
```

```
[RF] Root Mean Squared Error: 0.304534929154202
[RF] R2: 0.4619266428758483
```

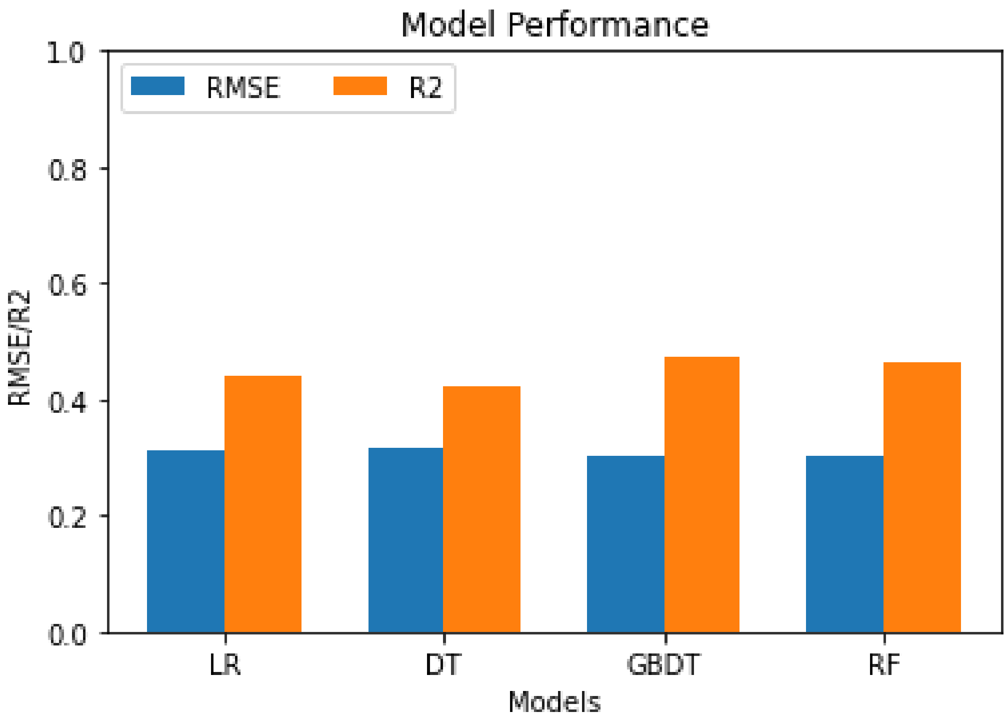
[49]:
```
# Plotting the RMSE and R2 score for each regressor:

RMSE_Results = [rmse_LR, rmse_DT, rmse_DBDT, rmse_RF]
R2_Results = [r2_LR, r2_DT, r2_GBDT, r2_RF]
labels = ['LR', 'DT', 'GBDT', 'RF']

rg= np.arange(4)
width = 0.35
plt.bar(rg, RMSE_Results, width, label="RMSE")
plt.bar(rg+width, R2_Results, width, label='R2')
plt.xticks(rg + width/2, labels)
plt.xlabel("Models")
plt.ylabel("RMSE/R2")
plt.ylim([0,1])

plt.title('Model Performance')
```

```
plt.legend(loc='upper left', ncol=2)
plt.show()
```



Analysis:

```
From the results above, we can see that the RMSE values of all the regressor models range
around the ~0.30-0.31 value, which indicates that my machine learning models generalizes well,
as its predictions are closer to the actual values in the dataset.

As far as the R2 score goes, the scores aquired are of the range ~0.42-0.47. These results
indicate that the model generalizes decently, but does not respond well to the variablity in
the data set.

Even though all four regressor models behaved similarily, the best performing regressor model
for the data set is the GBDT regressor, with the lowest RMSE and highest R2 value.
```

[ ]: