CSCI 5901 - Process of Data Science - Assignment 1

Team Members

Lan Chen (B00809814, lan.chen@dal.ca (mailto:lan.chen@dal.ca)), Jamuna Loganath (B00811590, jamuna.loganath@dal.ca (mailto:jamuna.loganath@dal.ca))

Part 1 Introduction. (2 marks total)

a.Explain the dataset with your own words. Focus on the attributes description. (2 marks)

- The given dataset is Zomato restaurant's dataset. It has 51717 rows of data. It has the following attributes.
  - URL - denotes the uniform resource locator of the restaurant.
  - address - denotes the address of the restaurant.
  - name - denotes the name of the resturant.
  - online_order - denotes whether online ordering facility is available in the restaurant or not.
  - book_table - denotes whether the table booking facilty is available or not.
  - rate - denotes the rating of the restaurant out of 5.
  - votes - denotes the number of votes made for the restaurant.
  - phone - denotes the phone number of the restaurant.
  - location - denotes the location of the restaurant.
  - rest_type - denotes the restaurant type.
  - dish_liked - denotes the dish that is most liked by people in the restaurant.
  - cuisines - denotes the cuisines available in the restaurant.
  - approx_cost(for two people) - denotes the approximate cost to be spend when two people visit the restautant.
  - reviews_list - denotes the review given by the customers.
  - menu- denotes the menu items of the restaurants.
  - listed_in(type) - denotes the type of food delivery mode of the restaurant like delivery, dine out, cafe, etc.,
  - listed_in(city) - denotes the city of the restaurant.

Part 2. Data pre-processing and understanding. (28 marks total)

a. Load the data. (3 marks)

In [127]:

```
# importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
from decimal import Decimal
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.linear_model import SGDRegressor
from sklearn import metrics
from skrebate import ReliefF
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LinearRegression
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV
%matplotlib inline
```

In [96]:

```
# read data from csv file
data = pd.read_csv('zomato.csv')
```

- The data is loaded successfully.

In [97]:

```
# loading the first data three rows
data.head(3)
```

Out[97]:

	url	address	name	online_order	book_table	rat
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/

```
In [98]: # to get information about the dataframe including the data types of each column
         data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
url                            51717 non-null object
address                        51717 non-null object
name                           51717 non-null object
online_order                   51717 non-null object
book_table                     51717 non-null object
rate                           43942 non-null object
votes                          51717 non-null int64
phone                          50509 non-null object
location                       51696 non-null object
rest_type                      51490 non-null object
dish_liked                     23639 non-null object
cuisines                       51672 non-null object
approx_cost(for two people)    51371 non-null object
reviews_list                   51717 non-null object
menu_item                      51717 non-null object
listed_in(type)                51717 non-null object
listed_in(city)                51717 non-null object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

b. Explore the data. Plot the distribution of the attributes (frequency). What trends can you find in your data? Are there attributes that are useless at this point? (10 marks)

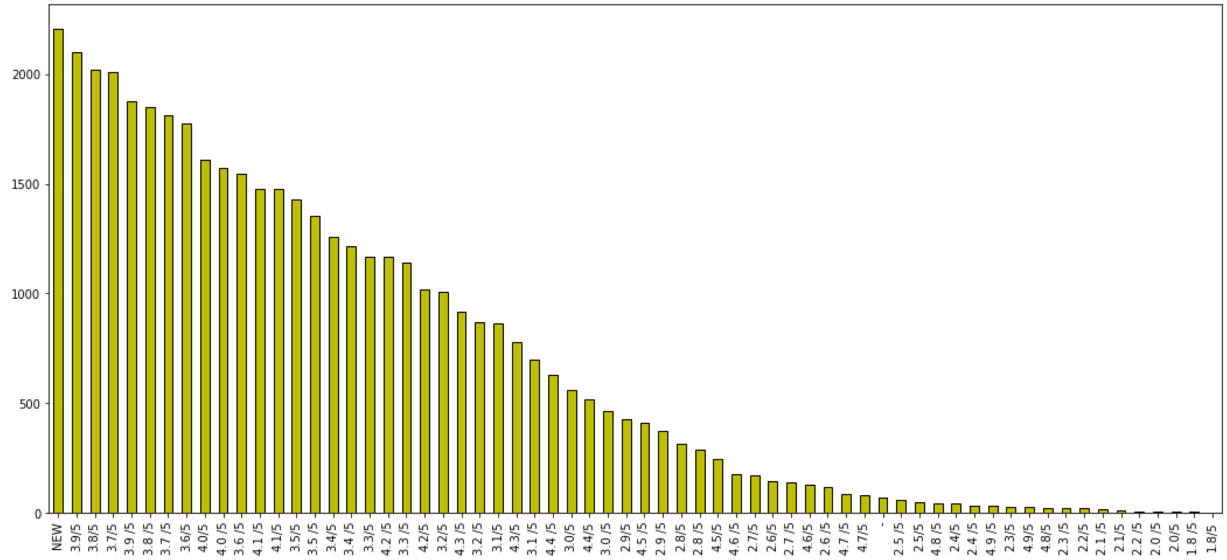
- The dataset has many attributes. Out of which only few attributes are chosen for plotting and analysis. Attributes such as listed_in(city),rating, listed_in(type), approx_cost(two people), votes, location are plotted below and the same is analyzed for understanding the trend.

In [99]:
```
# check unique values for every columns
for c in data.columns:
    print(c,data[c].nunique(),sep='---')
```

```
url---51717
address---11495
name---8792
online_order---2
book_table---2
rate---64
votes---2328
phone---14926
location---93
rest_type---93
dish_liked---5271
cuisines---2723
approx_cost(for two people)---70
reviews_list---22513
menu_item---9098
listed_in(type)---7
listed_in(city)---30
```

In [6]:
```
# plot histogram for column rating [1]
# refer to https://matplotlib.org/gallery/statistics/histogram_features.html
fig, ax = plt.subplots(figsize=(18,8))
data['rate'].value_counts().plot.bar(color='y',ec='black')
```

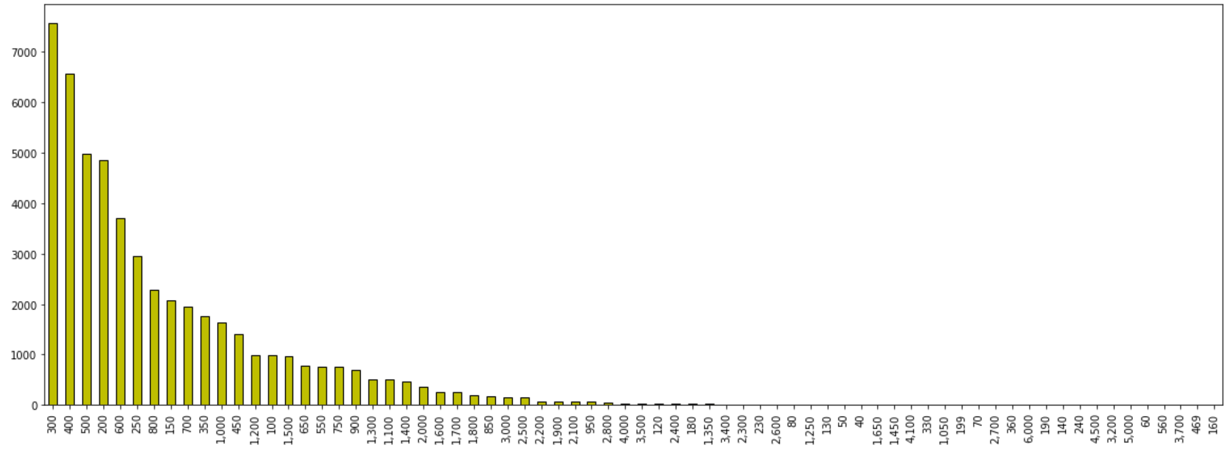
Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x243bb1038d0>
```



- The above chart shows the rating distribution of the restaurants. X-axis denotes the rating categories and y axis specifies the count of the specific rating. The plotting is done before cleaning the data. So, the corrupt data (rating) NEW is depicted as the majority one. Apart from that, It is clearly seen from the graph that 3.9 rating is the rating given by most of the people(roughly more than 2000 customers). And, 1.8 is the least rating given to the restaurants.

In [100]:
```
# Plotting for approximate cost for two people
fig, ax = plt.subplots(figsize=(20,7))
data['approx_cost(for two people)'].value_counts().plot.bar(color='y',ec='black')
```

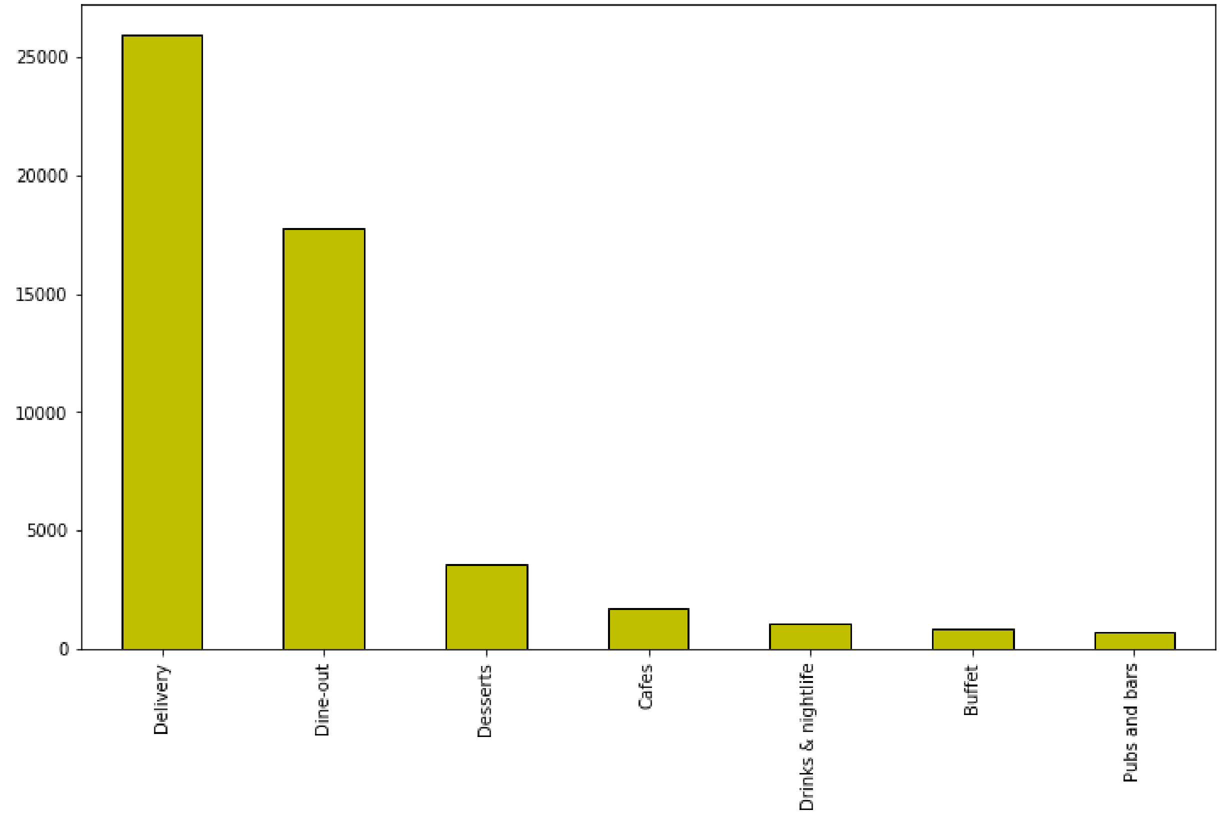
Out[100]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x243a4a1c6d8>
```



- The above graph represents the approximate cost need by two people to eat in the restaurant. x-axis denotes the approximate amount and y-axis denotes the number of restaurants. It is interpreted from the graph that 300 rupees is the approximate amount required by more than 7500 restaurants. 400 rupees is the approximate cost required in more than 6500 restaurants. The least number of restaurants require 1350 as the approximate cost.

```
In [101]: # plotting Listed_in (type)
          fig, ax = plt.subplots(figsize=(12,7))
          data['listed_in(type)'].value_counts().plot.bar(color='y',ec='black')
```

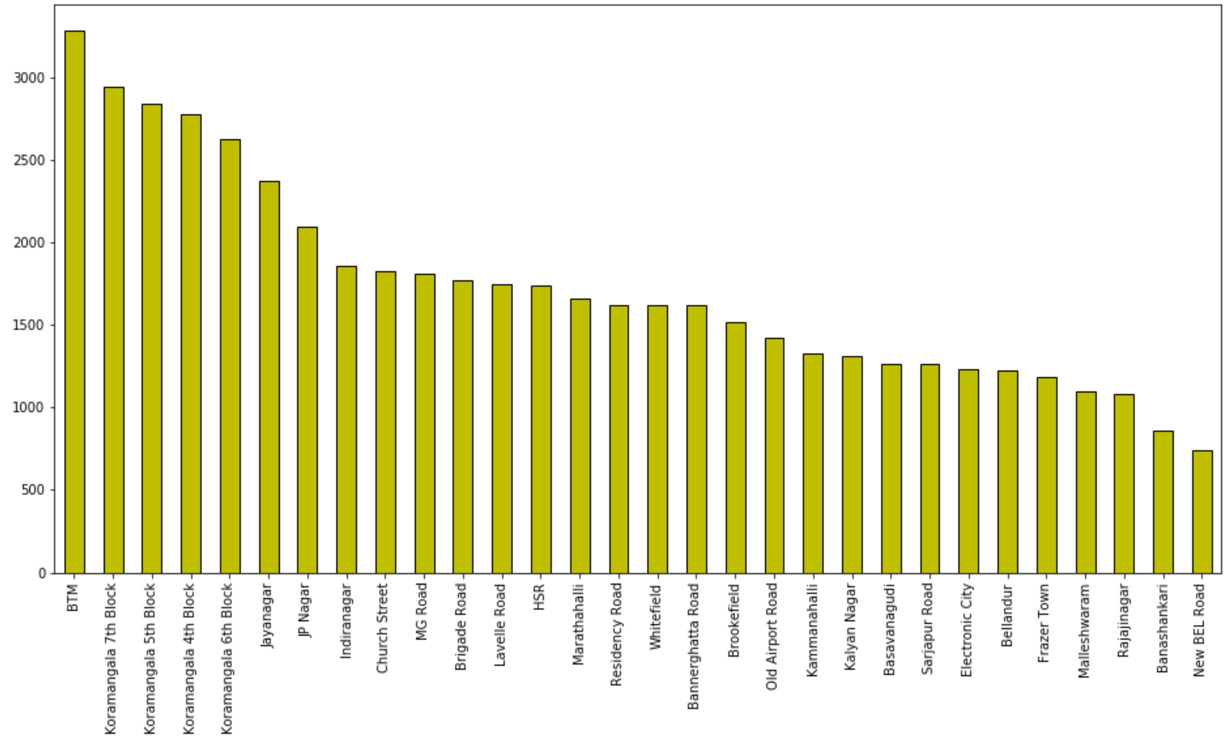
```
Out[101]: <matplotlib.axes._subplots.AxesSubplot at 0x24380eab400>
```



- The graph above is plotting to represent the listed_in(type) of the restaurants. The x-axis denotes the listed_in type and y-axis denotes the number of restaurants. It is evident from the graph that Delivery listed_in type is the type used in majority of the restaurants. More than 25,000 restaurants use delivery type. The next major one is Dine-out. Pubs and Bars are shown towards the right end of the graph with the least number of restaurants.

In [102]:
```
# plotting for listed_in(city)
fig, ax = plt.subplots(figsize=(16,8))
data['listed_in(city)'].value_counts().plot.bar(color='y',ec='black')
```

Out[102]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24380ff2be0>
```



- The above graph is plotted to show the frequency of distribution of the listed_in(city) of the restaurants. The x-axis denotes the listed_in(city) and y-axis denotes the number of restaurants. It is noticed from the graph that BTM is the city with highest number of restaurants (roughly more than 3000). The next higher city is Koramangala 7th block. Down the lane, New BEL Road is the city with least number of restaurants (roughly less than 1000).

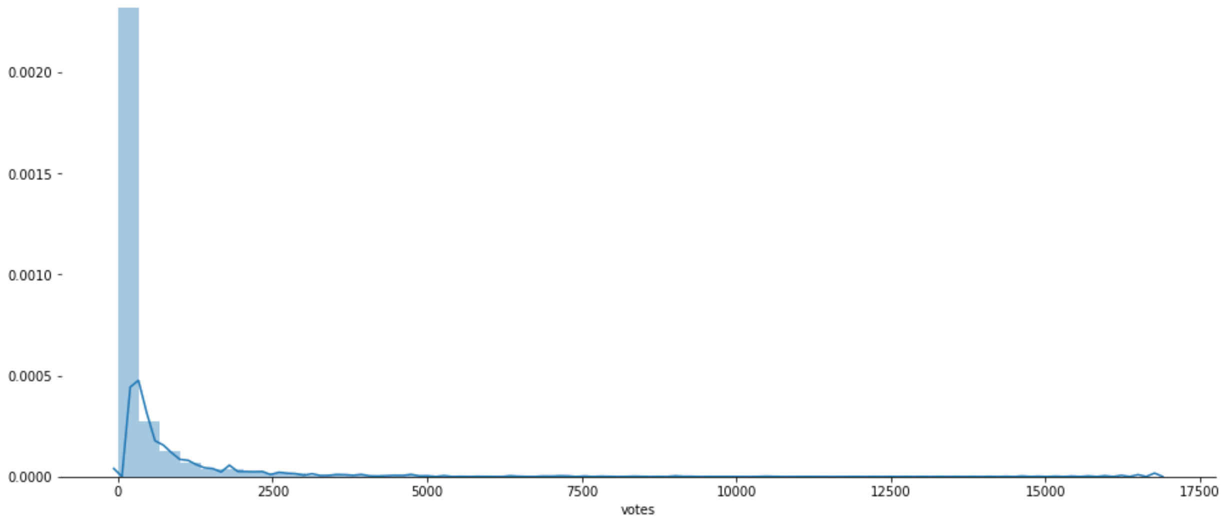
In [103]:

```
# plotting no of votes resturant using distplot and boxplot [2]
rest_rating = data.loc[data['votes'] > 0]
rest_rating = rest_rating['votes']

#f, axes = plt.subplots(1, 2, figsize=(15,5), sharex=True)
fig, ax = plt.subplots(figsize=(16,7),sharex=True)
sns.despine(left=True)
sns.distplot(rest_rating)
#sns.boxplot(rest_rating, ax=axes[1])
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\matplotlib\axes\_axes.py:6462: UserW
arning: The 'normed' kwarg is deprecated, and has been replaced by the 'densit
y' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24380e2e1d0>
```



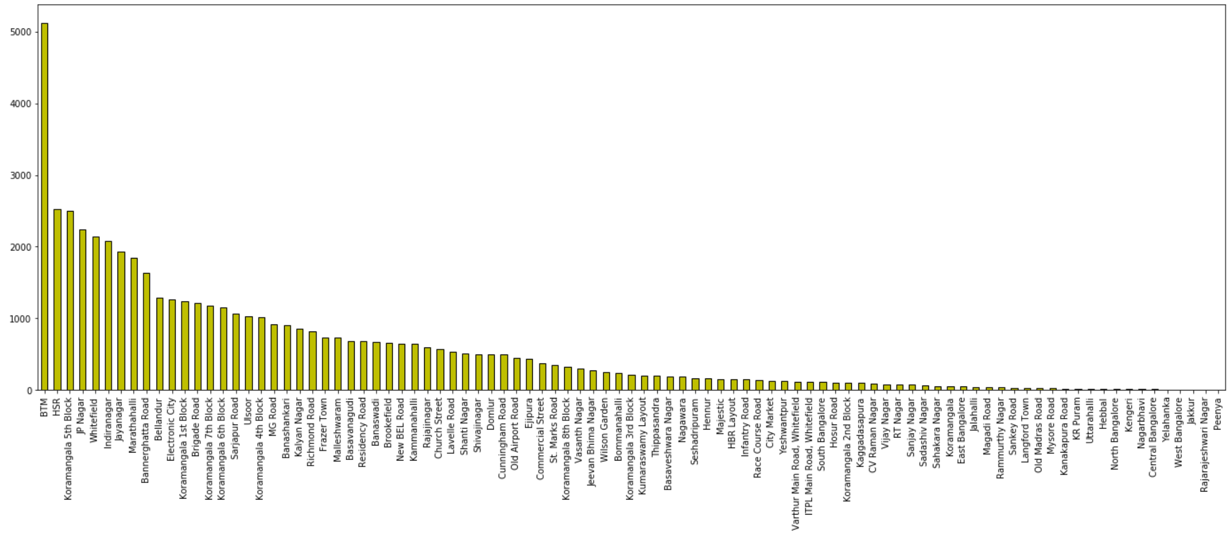
- The above distplot shows the frequency distribution of number of votes of the restaurants. The x-axis denotes the number of restaurants and the y-axis denotes the frequency distribution of the votes of the restaurants.The graph shows that the number of votes is highly distributed for less than 2500 votes.

In [104]:

```
# Plotting for location
fig, ax = plt.subplots(figsize=(24,8))
data['location'].value_counts().plot.bar(color='y',ec='black')
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243810b1d68>
```



- The above graph represents the various locations in which the restaurants are distributed. x-axis denotes the various locations and y-axis denotes the number of restaurants. It is interpreted from the graph that BTM is the location which has the most number of restaurants (roughly more than 5000 restaurants).

Are there attributes that are useless at this point?

- Yes, there are some attributes that are useless at this point. The URL, phone number, menu items, online_order, Book_table are some of the fields that are not useful.

c. Are there restaurant duplicates in the data? Detect and if there is, clean it.

In [128]:
```
#findout the same restaurant [3]
data_duplicate = data[data.duplicated(['name','address'])]
data_duplicate
```

Out[128]:

	url	address	
14	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	
44	https://www.zomato.com/bangalore/onesta-banash...	2469, 3rd Floor, 24th Cross, Opposite BDA Comp...	
104	https://www.zomato.com/bangalore/cafe-nova-ban...	101, Water Tank Road, 2nd Block, 3rd Stage, Ba...	
	https://www.zomato.com/bangalore/caf%C3%A9-	12,29 Near PES University Back	CafÃ□Â□Ã□Â□Ã□Â□Ã□Â□Ã□

- Yes, there are restaurant duplicates in the data. The duplicate restaurants are dropped using the following code. After dropping, the data is printed below.

In [129]:
```
#Finding the unique restaurants
data_duplicate['name'].unique()
```

Out[129]:
```
array(['San Churro Cafe', 'Onesta', 'CAFE NOVA', ...,
       'Chime - Sheraton Grand Bengaluru Whitefield Hotel &...',
       'Vinod Bar And Restaurant', 'The Nest - The Den Bengaluru'],
      dtype=object)
```

In [130]:
```
# remove duplicated restaurants [4]
data_single = data.drop_duplicates(['name','address'])
```

- The restaurant duplicates are successfully detected and removed from the dataset

In [131]:

```
# prints the data after removing the duplicates
data_single.head(10)
```

Out[131]:

	url	address	name	online_order	book_table
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No
5	https://www.zomato.com/bangalore/timepass-dinn...	37, 5-1, 4th Floor, Bosco Court, Gandhi Bazaar...	Timepass Dinner	Yes	No
6	https://www.zomato.com/bangalore/rosewood-inte...	19/1, New Timberyard Layout, Beside Satellite ...	Rosewood International Hotel - Bar & Restaurant	No	No
7	https://www.zomato.com/bangalore/onesta-banash...	2469, 3rd Floor, 24th Cross, Opposite BDA Comp...	Onesta	Yes	Yes
8	https://www.zomato.com/bangalore/penthouse-caf...	1, 30th Main Road, 3rd Stage, Banashankari, Ba...	Penthouse Cafe	Yes	No

	url	address	name	online_order	book_table
9	https://www.zomato.com/bangalore/smacznego-ban...	2470, 21 Main Road, 25th Cross, Banashankari, ...	Smacznego	Yes	No

d. What is the neighborhood with the highest average rating? What are the major characteristics of this neighborhood (e.g., type of restaurant, type of food they offer, etc).

- To findout the neighbourhood with highest average rating, the rating field should be cleaned. First finding out the unique data to understand the data clearly and to make sure the elements to be cleaned.

In [132]:

```
data_single['rate'].unique()
```

Out[132]:

```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', nan, '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5', '-',
       '3.6 /5', '4.8/5', '3.9 /5', '3.7 /5', '2.9 /5', '2.8 /5',
       '4.2 /5', '3.5 /5', '4.1 /5', '2.7 /5', '3.4 /5', '3.3 /5',
       '3.2 /5', '4.3 /5', '4.9/5', '2.1/5', '2.0/5', '4.4 /5', '4.5 /5',
       '1.8/5', '4.0 /5', '4.6 /5', '3.1 /5', '3.0 /5', '2.6 /5',
       '2.3 /5', '2.5 /5', '4.7 /5', '4.8 /5', '4.9 /5', '2.4 /5',
       '2.0 /5'], dtype=object)
```

In [133]:

```
#clean nan value for split. The nan values are replaced with '-'.
data_single['rate'].fillna('-', inplace=True)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\pandas\core\generic.py:5430: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  self._update_inplace(new_data)
```

In [134]:
```
data_single['rate'].unique()
```

Out[134]:
```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', '-', '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.6 /5', '4.8/5', '3.9 /5', '3.7 /5', '2.9 /5', '2.8 /5',
       '4.2 /5', '3.5 /5', '4.1 /5', '2.7 /5', '3.4 /5', '3.3 /5',
       '3.2 /5', '4.3 /5', '4.9/5', '2.1/5', '2.0/5', '4.4 /5', '4.5 /5',
       '1.8/5', '4.0 /5', '4.6 /5', '3.1 /5', '3.0 /5', '2.6 /5',
       '2.3 /5', '2.5 /5', '4.7 /5', '4.8 /5', '4.9 /5', '2.4 /5',
       '2.0 /5'], dtype=object)
```

In [135]:
```
# Removing '/5' from the data
data_single['rate']=data_single['rate'].apply(lambda x: x.split('/')[0])
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\ipykernel_launcher.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [136]:
```
data_single['rate'].unique()
```

Out[136]:
```
array(['4.1', '3.8', '3.7', '3.6', '4.6', '4.0', '4.2', '3.9', '3.1',
       '3.0', '3.2', '3.3', '2.8', '4.4', '4.3', 'NEW', '2.9', '3.5', '-',
       '2.6', '3.8 ', '3.4', '4.5', '2.5', '2.7', '4.7', '2.4', '2.2',
       '2.3', '3.6 ', '4.8', '3.9 ', '3.7 ', '2.9 ', '2.8 ', '4.2 ',
       '3.5 ', '4.1 ', '2.7 ', '3.4 ', '3.3 ', '3.2 ', '4.3 ', '4.9',
       '2.1', '2.0', '4.4 ', '4.5 ', '1.8', '4.0 ', '4.6 ', '3.1 ',
       '3.0 ', '2.6 ', '2.3 ', '2.5 ', '4.7 ', '4.8 ', '4.9 ', '2.4 ',
       '2.0 '], dtype=object)
```

In [137]:
```
# change value of rate is'NEW' to 0 for further proceed
# data_single[data_single['rate']=='NEW'] = 0
data_single['rate'].values[data_single['rate'].values =='NEW'] = 0
```

In [138]:
```
data_single['rate'].unique()
```

Out[138]:
```
array(['4.1', '3.8', '3.7', '3.6', '4.6', '4.0', '4.2', '3.9', '3.1',
       '3.0', '3.2', '3.3', '2.8', '4.4', '4.3', 0, '2.9', '3.5', '-',
       '2.6', '3.8 ', '3.4', '4.5', '2.5', '2.7', '4.7', '2.4', '2.2',
       '2.3', '3.6 ', '4.8', '3.9 ', '3.7 ', '2.9 ', '2.8 ', '4.2 ',
       '3.5 ', '4.1 ', '2.7 ', '3.4 ', '3.3 ', '3.2 ', '4.3 ', '4.9',
       '2.1', '2.0', '4.4 ', '4.5 ', '1.8', '4.0 ', '4.6 ', '3.1 ',
       '3.0 ', '2.6 ', '2.3 ', '2.5 ', '4.7 ', '4.8 ', '4.9 ', '2.4 ',
       '2.0 '], dtype=object)
```

In [139]:
```
# change value of rate is'-' to 0 for further proceed
# data_single[data_single['rate']=='-'] = 0
data_single['rate'].values[data_single['rate'].values =='-'] = 0
```

In [140]:
```
data_single['rate'].unique()
```

Out[140]:
```
array(['4.1', '3.8', '3.7', '3.6', '4.6', '4.0', '4.2', '3.9', '3.1',
       '3.0', '3.2', '3.3', '2.8', '4.4', '4.3', 0, '2.9', '3.5', '2.6',
       '3.8 ', '3.4', '4.5', '2.5', '2.7', '4.7', '2.4', '2.2', '2.3',
       '3.6 ', '4.8', '3.9 ', '3.7 ', '2.9 ', '2.8 ', '4.2 ', '3.5 ',
       '4.1 ', '2.7 ', '3.4 ', '3.3 ', '3.2 ', '4.3 ', '4.9', '2.1',
       '2.0', '4.4 ', '4.5 ', '1.8', '4.0 ', '4.6 ', '3.1 ', '3.0 ',
       '2.6 ', '2.3 ', '2.5 ', '4.7 ', '4.8 ', '4.9 ', '2.4 ', '2.0 '],
      dtype=object)
```

In [141]:
```
data_single.head(5)
```

Out[141]:

	url	address	name	online_order	book_table	rat
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.

In [142]:

```
# float values
data_single['rate'] = data_single['rate'].apply(lambda x : float(x))
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\ipykernel_launcher.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [143]:

```
data_single.head(5)
```

Out[143]:

	url	address	name	online_order	book_table	rat
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.
3	https://www.zomato.com/bangalore/addhuri-udupi...	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.
4	https://www.zomato.com/bangalore/grand-village...	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.

In [144]:

```
data_single['rate'].unique()
```

Out[144]:

```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 0. , 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4, 2.2,
       2.3, 4.8, 4.9, 2.1, 2. , 1.8])
```

In [145]:

```
np.mean(data_single['rate'])
```

Out[145]:

```
2.702312184974807
```

In [146]:

```
# change the 0 value to mean value of the rate
data_single['rate'].values[data_single['rate'].values ==0] = round(np.mean(data_s
```

In [147]:

```
data_single['rate'].unique()
```

Out[147]:

```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 2.7, 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 4.7, 2.4, 2.2, 2.3,
       4.8, 4.9, 2.1, 2. , 1.8])
```

In [148]:

```
data_single.head(3)
```

Out[148]:

	url	address	name	online_order	book_table	rate
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8

In [149]:

```
# Groupingup the neighbourhood [6]
neibor=data_single[['rate','location']].groupby('location').mean()
```

In [150]:

```
# displaying the neighbour in descending order based on the mean value calculated
neibor.sort_values(by=['rate'],ascending=False)
```

Out[150]:

	rate
location	
Lavelle Road	3.975000
Sankey Road	3.938462
St. Marks Road	3.894286
Rajarajeshwari Nagar	3.850000
Church Street	3.835088
Race Course Road	3.823333
Koramangala 5th Block	3.787313
Koramangala 3rd Block	3.713043
Kengeri	3.700000
Central Bangalore	3.700000
Residency Road	3.700000
Cunningham Road	3.693878
MG Road	3.662245
Sadashiv Nagar	3.653125
Langford Town	3.650000
Koramangala 7th Block	3.638621
Infantry Road	3.613333
Koramangala 6th Block	3.605298
Indiranagar	3.584586
Jayanagar	3.584469
Koramangala 4th Block	3.577444
Malleshwaram	3.571849
Sahakara Nagar	3.540000
Vasanth Nagar	3.520370
Basavanagudi	3.518009
Seshadripuram	3.512069
Kalyan Nagar	3.511157
Brigade Road	3.504580
Richmond Road	3.503191
Ulsoor	3.492437
...	...
Mysore Road	3.276923
Bannerghatta Road	3.272996

	rate
location	
Thippasandra	3.272222
Basaveshwara Nagar	3.271429
Bellandur	3.266942
Varthur Main Road, Whitefield	3.253333
Hennur	3.247692
Banaswadi	3.243062
HBR Layout	3.235556
East Bangalore	3.235000
Sanjay Nagar	3.231915
Jalahalli	3.231818
Shanti Nagar	3.228571
CV Raman Nagar	3.227273
KR Puram	3.223077
Rammurthy Nagar	3.215789
South Bangalore	3.200000
Peenya	3.200000
Magadi Road	3.200000
Electronic City	3.184521
Wilson Garden	3.182000
Hebbal	3.150000
North Bangalore	3.150000
Shivajinagar	3.116393
Ejipura	3.104615
City Market	3.097778
West Bangalore	3.050000
Bommanahalli	2.980000
Nagarbhavi	2.875000
Jakkur	2.700000

93 rows × 1 columns

- From the above output, it is noted that Lavelle Road neighbourhood has the highest average rating . The rating is 3.975000

In [151]:

```
highestneigbor = data_single[data_single['location'] == 'Lavelle Road']
```

In [152]:

```
#Displaying the Lavelle Road rows
highestneigbor
```

Out[152]:

	url	address
4981	https://www.zomato.com/bangalore/jw-kitchen-jw...	JW Marriott, 24/1, Vittal Mallya Road, Lavelle...
4987	https://www.zomato.com/bangalore/rasovara-lave...	Level 2, The Collection, UB City, Vithal Mally...
4994	https://www.zomato.com/bangalore/the-spice-baz...	40/2, Lavelle Road, Bangalore

What are the major characteristics of this neighborhood?

- The major characteristics of the neighborhood (Lavelle road) include cuisines, dish_liked, rest_type.

In [153]:

```
#characteristics of heighest neighbour - cuisines
highestneigborCuisines = []
highestneigbor['cuisines'].apply(lambda x : highestneigborCuisines.extend(x.split

cusisinesPd = pd.DataFrame(highestneigborCuisines, columns =['cuisines'])
cusisinesPd = cusisinesPd['cuisines'].apply(lambda x : x.strip())

cusisinesPd.value_counts()
```

Out[153]:

```
Italian          14
Continental      12
North Indian     12
Desserts         11
Salad             7
Cafe              7
Bakery            6
Asian             6
Chinese           6
Seafood           4
Thai              4
European          4
Japanese          4
Fast Food         3
Ice Cream         3
Pizza             3
Mediterranean     3
Spanish           2
BBQ               2
American          2
Mughlai           2
South Indian      2
Momos             2
French            2
Beverages         2
Steak             2
Healthy Food      2
Finger Food       2
Tex-Mex           1
Burger            1
Juices            1
North Eastern     1
Kebab             1
Modern Indian     1
Rajasthani        1
Mangalorean       1
Parsi             1
Mexican           1
Sandwich          1
Indonesian        1
Street Food       1
Name: cuisines, dtype: int64
```

- cusine is one of the important characteristics of the highest average rating neighbourhood. From the above results, Italian cuisine is the cuisine with the heighest value.

In [154]:

```
#characteristics of heighest neighbour - dish_liked
highestneigborDishLiked = []
highestneigbor['dish_liked'].apply(lambda x : highestneigborDishLiked.extend(str(

dishLikedPd = pd.DataFrame(highestneigborDishLiked, columns =['dish_liked'])

dishLikedPd = dishLikedPd['dish_liked'].apply(lambda x : x.strip())
dishLikedPd.value_counts()
```

Out[154]:

```
Cocktails                   16
Salads                      12
Pizza                       11
Pasta                       11
nan                          9
Brownie                      8
Tiramisu                     7
Coffee                       6
Sandwiches                   5
Sangria                      5
Mocktails                    5
Sushi                        4
Burgers                      4
Hot Chocolate                4
Martini                      4
Nachos                       4
Salad                        3
Sandwich                     3
Chaat                        3
Sea Food                     3
Momos                        3
Tempura Prawns               3
Beer                         3
Chocolate Cake               2
Pancakes                     2
Chicken Gyoza                2
Risotto                      2
Eclair                       2
Wine                         2
Dumplings                    2
                            ..
Virgin Sangria               1
Peri Peri Chicken Burger     1
Lunch Buffet                 1
Faluda                       1
Paratha                      1
Belgian Dark Chocolate       1
Aamras                       1
Arugula Salad                1
Jalapeno Poppers             1
Laksa                        1
Begun Pora                   1
Masala Dosa                  1
Rolls                        1
Penne Pasta                  1
Upma                         1
Tea                          1
```

```
Lasagne                      1
Stout                        1
Protein Smoothie             1
Chicken Soup                 1
Hot Chocolate Fudge          1
Orange Tart                  1
Saffron Pilaf                1
Peri Peri Chicken            1
Chocolate Volcano            1
Long Island Iced Tea         1
Craft Beer                   1
Appletini                    1
Pie                          1
Avocado Sandwich             1
Name: dish_liked, Length: 167, dtype: int64
```

- dish_liked is another important characteristic of the highest average rating neighbourhood. From the above results, Cocktails are liked very much by the people in that neighbourhood.

In [155]:

```
#characteristics of heighest neighbour - rest_type
highestneigborRestType = []
highestneigbor['rest_type'].apply(lambda x : highestneigborRestType.extend(str(x)

restTypesPd = pd.DataFrame(highestneigborRestType, columns =['rest_type'])

restTypesPd = restTypesPd['rest_type'].apply(lambda x : x.strip())
restTypesPd.value_counts()
```

Out[155]:

```
Casual Dining     21
Bar               10
Fine Dining        7
Bakery             6
Cafe               6
Quick Bites        6
Dessert Parlor     5
Microbrewery       4
Lounge             4
Pub                2
Beverage Shop      2
Confectionery      1
Kiosk              1
Delivery           1
Irani Cafee        1
Name: rest_type, dtype: int64
```

- rest_type is another major characteristic of the highest average rating neighbourhood. From the above results, Casual dining is at the top with 21 restaurants in the neighbourhood.

Part 3 Build the best model you can that forecasts the approximate cost of a meal for two people using the attributes location, rating, restaurant type, and cuisine. (70 marks total)

a. Explain what is the task you’re solving (e.g., supervised x unsupervised, classification x regression x clustering or similarity matching x etc). (5 marks)

- It's supervised task, because the task has a target variable to predict.
- It's regression problem, the task requires to predict some number of value.

b. What models will you choose? Why? (5 marks)

- We will choose model of regression, because it requires predict a numerical number. We compare the performance of three models namely, Linear regression, SGDRegressor and MPLRegressor and choose the best model for this task.

c. Which metrics will you use to evaluate your model? (5 marks)

- The Regression metrics of sklearn.metrics we used are as below:
  1. mean_squared_error
  2. r2_score

d. How do you make sure not to overfit? (5 marks)

- 1. split the dataset into training data and testing data, using training data to fit the model, and using testing data to evaluate the performance
- 2. compare the perfomance on training data with the performance on testing data, if the former is much better than the later, then there may be a overfitting in the model
- 3. use cross validation approach to evaluate the true performance of the model.

e. Build your model and verify how it performs (using the metrics you have chosen in Section 3(c)) in your training data. Justify which evaluation approach you are using?(Out of sample validation or Cross-validation). Use a plot to justify your findings. How good is your model? (10 marks). Question e and f are answered parallelly

f. Test your model in your testing set and evaluate its performance. Use a plot to justify your findings. How is it performing compared to your training data? (15 marks)

- Before building the model, there are some data preparationg task needed to be done, include:
- 1. convert the string type of target variable to int type
- 2. drop the missing value of the dataset for modeling
- 3. convert the string type of the target variable(approx_cost(for two people)) to int type
- 4. convert the categorical value type of the other variables to the binary value type

In [156]:

```
# change the nan value in the target variable
data_single['approx_cost(for two people)'].fillna('0', inplace=True)
data_single['approx_cost(for two people)'].unique()
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\pandas\core\generic.py:5430: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  self._update_inplace(new_data)
```

Out[156]:

```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
       '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
       '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
       '1,600', '230', '130', '50', '190', '1,700', '0', '1,400', '180',
       '1,350', '2,200', '2,000', '1,800', '1,900', '330', '2,500',
       '2,100', '3,000', '2,800', '3,400', '40', '1,250', '3,500',
       '4,000', '2,400', '2,600', '120', '1,450', '469', '70', '3,200',
       '60', '560', '240', '360', '6,000', '1,050', '2,300', '4,100',
       '5,000', '3,700', '1,650', '2,700', '4,500', '140'], dtype=object)
```

In [157]:

```
# convert the string type of target variable to int type
data_single.loc[:,'approx_cost(for two people)'] =data_single['approx_cost(for tw
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\pandas\core\indexing.py:543: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
  self.obj[item] = s
```

In [158]:

```
#using mean value to fill 0 value which is nan value before
data_single['approx_cost(for two people)'].values[data_single['approx_cost(for tw
data_single['approx_cost(for two people)'].unique()
```

Out[158]:

```
array([ 800,  300,  600,  700,  550,  500,  450,  650,  400,  900,  200,
        750,  150,  850,  100, 1200,  350,  250,  950, 1000, 1500, 1300,
        199,   80, 1100,  160, 1600,  230,  130,   50,  190, 1700,  485,
       1400,  180, 1350, 2200, 2000, 1800, 1900,  330, 2500, 2100, 3000,
       2800, 3400,   40, 1250, 3500, 4000, 2400, 2600,  120, 1450,  469,
         70, 3200,   60,  560,  240,  360, 6000, 1050, 2300, 4100, 5000,
       3700, 1650, 2700, 4500,  140], dtype=int64)
```

In [159]:

```
# get the sub data of the original dataset for modeling
d = data_single[['location','rate','listed_in(type)','cuisines','approx_cost(for
```

In [160]:

```
# check the row has missing value or nan value
d[d.isna().any(axis=1)]
```

Out[160]:

	location	rate	listed_in(type)	cuisines	approx_cost(for two people)
438	Banashankari	2.7	Delivery	NaN	150
440	Kumaraswamy Layout	3.3	Delivery	NaN	100
1662	NaN	2.7	Delivery	NaN	485
4037	Marathahalli	2.7	Delivery	NaN	200
6887	Whitefield	3.6	Delivery	NaN	400
6897	Whitefield	2.7	Delivery	NaN	400
7277	Whitefield	2.7	Delivery	NaN	400
7555	Marathahalli	2.7	Delivery	NaN	500
13591	Electronic City	2.7	Delivery	NaN	500
13693	NaN	2.7	Delivery	NaN	485
16351	NaN	2.7	Delivery	NaN	485
22974	Kumaraswamy Layout	2.7	Delivery	NaN	500
24725	Kalyan Nagar	3.3	Dine-out	NaN	600
26519	NaN	2.7	Buffet	NaN	485
27672	NaN	2.7	Delivery	NaN	485
40354	NaN	2.7	Dine-out	NaN	485
40556	NaN	2.7	Cafes	NaN	485
46586	NaN	2.7	Delivery	NaN	485
46609	NaN	2.7	Delivery	NaN	485

In [161]:

```
#drop all the nan value in the sub-data
d.dropna(inplace=True)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\ipykernel_launcher.py:2: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stab
le/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-doc
s/stable/indexing.html#indexing-view-versus-copy)
```

In [162]:
```
d.head(5)
```

Out[162]:

	location	rate	listed_in(type)	cuisines	approx_cost(for two people)
0	Banashankari	4.1	Buffet	North Indian, Mughlai, Chinese	800
1	Banashankari	4.1	Buffet	Chinese, North Indian, Thai	800
2	Banashankari	3.8	Buffet	Cafe, Mexican, Italian	800
3	Banashankari	3.7	Buffet	South Indian, North Indian	300
4	Basavanagudi	3.8	Buffet	North Indian, Rajasthani	600

In [163]:
```
# check the values of each features of dataset
d['location'].unique()
```

Out[163]:
```
array(['Banashankari', 'Basavanagudi', 'Mysore Road', 'Jayanagar',
       'Kumaraswamy Layout', 'Rajarajeshwari Nagar', 'Vijay Nagar',
       'Uttarahalli', 'JP Nagar', 'South Bangalore', 'City Market',
       'Nagarbhavi', 'Bannerghatta Road', 'BTM', 'Kanakapura Road',
       'Bommanahalli', 'CV Raman Nagar', 'Electronic City', 'HSR',
       'Marathahalli', 'Sarjapur Road', 'Wilson Garden', 'Shanti Nagar',
       'Koramangala 5th Block', 'Koramangala 8th Block', 'Richmond Road',
       'Koramangala 7th Block', 'Jalahalli', 'Koramangala 4th Block',
       'Bellandur', 'Whitefield', 'East Bangalore', 'Old Airport Road',
       'Indiranagar', 'Koramangala 1st Block', 'Frazer Town', 'RT Nagar',
       'MG Road', 'Brigade Road', 'Lavelle Road', 'Church Street',
       'Ulsoor', 'Residency Road', 'Shivajinagar', 'Infantry Road',
       'St. Marks Road', 'Cunningham Road', 'Race Course Road',
       'Commercial Street', 'Vasanth Nagar', 'HBR Layout', 'Domlur',
       'Ejipura', 'Jeevan Bhima Nagar', 'Old Madras Road', 'Malleshwaram',
       'Seshadripuram', 'Kammanahalli', 'Koramangala 6th Block',
       'Majestic', 'Langford Town', 'Central Bangalore', 'Sanjay Nagar',
       'Brookefield', 'ITPL Main Road, Whitefield',
       'Varthur Main Road, Whitefield', 'KR Puram',
       'Koramangala 2nd Block', 'Koramangala 3rd Block', 'Koramangala',
       'Hosur Road', 'Rajajinagar', 'Banaswadi', 'North Bangalore',
       'Nagawara', 'Hennur', 'Kalyan Nagar', 'New BEL Road', 'Jakkur',
       'Rammurthy Nagar', 'Thippasandra', 'Kaggadasapura', 'Hebbal',
       'Kengeri', 'Sankey Road', 'Sadashiv Nagar', 'Basaveshwara Nagar',
       'Yeshwantpur', 'West Bangalore', 'Magadi Road', 'Yelahanka',
       'Sahakara Nagar', 'Peenya'], dtype=object)
```

In [164]:
```
d['rate'].unique()
```

Out[164]:
```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 2.7, 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 4.7, 2.4, 2.2, 2.3,
       4.8, 4.9, 2.1, 2. , 1.8])
```

In [165]:
```
d['listed_in(type)'].unique()
```

Out[165]:
```
array(['Buffet', 'Cafes', 'Delivery', 'Desserts', 'Dine-out',
       'Drinks & nightlife', 'Pubs and bars'], dtype=object)
```

In [167]:
```
d['approx_cost(for two people)'].unique()
```

Out[167]:
```
array([ 800,  300,  600,  700,  550,  500,  450,  650,  400,  900,  200,
        750,  150,  850,  100, 1200,  350,  250,  950, 1000, 1500, 1300,
        199,   80, 1100,  160, 1600,  230,  130,   50,  190, 1700,  485,
       1400,  180, 1350, 2200, 2000, 1800, 1900,  330, 2500, 2100, 3000,
       2800, 3400,   40, 1250, 3500, 4000, 2400, 2600,  120, 1450,  469,
         70, 3200,   60,  560,  240,  360, 6000, 1050, 2300, 4100, 5000,
       3700, 1650, 2700, 4500,  140], dtype=int64)
```

In [168]:
```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12480 entries, 0 to 51714
Data columns (total 5 columns):
location                       12480 non-null object
rate                           12480 non-null float64
listed_in(type)                12480 non-null object
cuisines                       12480 non-null object
approx_cost(for two people)    12480 non-null int64
dtypes: float64(1), int64(1), object(3)
memory usage: 585.0+ KB
```

In [169]:
```
# convert categorial value of cuisines to binary value
# refer to https://datascience.stackexchange.com/questions/14847/multiple-categor
cuisines = d['cuisines']
```

In [53]:
```
cleaned_cuisine = cuisines.str.split(',', expand=True).stack()
cuisine_binary =pd.get_dummies(cleaned_cuisine, prefix='cuisine_').groupby(level=
cuisine_binary.head(5)
```

Out[53]:

	cuisine__ Afghan	cuisine__ Afghani	cuisine__ African	cuisine__ American	cuisine__ Andhra	cuisine__ Arabian	cuisine__ Asian	cuisine__ Assamese	cuisine__ Awadhi
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

5 rows × 190 columns

In [170]:
```
# convert the categorial value of location, rate, list_in(type) to binary value
convert_other = d[['location','rate','listed_in(type)']]
data_dummies = pd.get_dummies(convert_other, prefix_sep='_', drop_first=True)
```

In [171]:

```
data_dummies.head(5)
```

Out[171]:

	rate	location_Banashankari	location_Banaswadi	location_Bannerghatta Road	location_Basavanagudi
0	4.1	1	0	0	0
1	4.1	1	0	0	0
2	3.8	1	0	0	0
3	3.7	1	0	0	0
4	3.8	0	0	0	1

5 rows × 99 columns

In [172]:

```
# the features used to predict
X= data_dummies.join(cuisine_binary)
```

In [173]:

```
#target variable
y = d['approx_cost(for two people)']
y.unique()
```

Out[173]:

```
array([ 800,  300,  600,  700,  550,  500,  450,  650,  400,  900,  200,
        750,  150,  850,  100, 1200,  350,  250,  950, 1000, 1500, 1300,
        199,   80, 1100,  160, 1600,  230,  130,   50,  190, 1700,  485,
       1400,  180, 1350, 2200, 2000, 1800, 1900,  330, 2500, 2100, 3000,
       2800, 3400,   40, 1250, 3500, 4000, 2400, 2600,  120, 1450,  469,
         70, 3200,   60,  560,  240,  360, 6000, 1050, 2300, 4100, 5000,
       3700, 1650, 2700, 4500,  140], dtype=int64)
```

* Build the model and verify how it performs in your training data.

*** We used cross validation for evaluation, so we evaluate the performances of training data and testing data at the same time Note: Question e and f are anwsered parallelly as the training and testing data are handled at the same time.**

*** In this assignment, we evaluate 3 regression models. They are LinearRegression, SGDRegressor, MLPRegressor**

Model 1. LinearRegression

In [174]:

```
#Build the Model LinearRegression

lm = LinearRegression()
# using r2 score and squred error to evaluate performance using cross validation
# refer to https://scikit-learn.org/stable/modules/generated/sklearn.model_select
train_r2 = [];
train_error = [];
test_r2 = [];
test_error = [];


# using cross validation

kf = KFold(n_splits=5)
kf.get_n_splits(X)

for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    lm.fit(X_train,y_train)
    train_prediction = lm.predict(X_train)
    test_prediction = lm.predict(X_test)

    #using metrics to evaluate the performance on training data for every fold
    train_r2.append(metrics.r2_score(y_train, train_prediction))
    train_error.append(metrics.mean_squared_error(y_train, train_prediction))
    #using metrics to evaluate the performance on testing data for every fold
    test_r2.append(metrics.r2_score(y_test, test_prediction))
    test_error.append(metrics.mean_squared_error(y_test, test_prediction))

# caculate the mean metrics for the whole data set
mean_train_r2 = np.mean(train_r2)
mean_train_error = np.mean(train_error)
mean_test_r2 = np.mean(test_r2)
mean_test_error = np.mean(test_error)
```

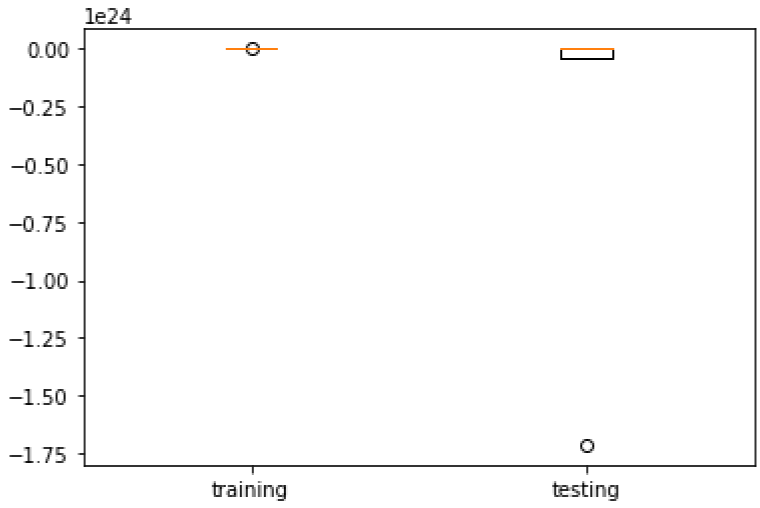
In [175]:

```

print("traing data r2 score: ", train_r2)
print("testing data r2 score: ",test_r2)
print("training data squared error:", train_error)
print("testing data squared error:",test_error)


print("average traing data r2 score: ", mean_train_r2)
print("average testing data r2 score: ",mean_test_r2)
print("average training data squared error:", mean_train_error)
print("average data squared error:",mean_test_error)
```

```
traing data r2 score:  [0.5765477091303313, 0.5624232275220173, 0.5832379185538
111, 0.5815195570651586, 0.5771011144539118]
testing data r2 score:  [-1.916792670777695e+24, -8.762256801732944e+20, -4.553
1541073471373e+20, -1.778161310118866e+20, -3.0621376195073954e+23]
training data squared error: [72044.05492413502, 53944.70564616796, 69076.08741
554235, 65936.00433687064, 60734.06396572407]
testing data squared error: [1.4885737902482625e+29, 2.246106155529634e+26, 4.3
95782156792545e+25, 2.3231812314113385e+25, 5.716971250811889e+28]
average traing data r2 score:  0.576165905345046
average testing data r2 score:  -4.4490315799007095e+23
average training data squared error: 64346.983257688014
average data squared error: 4.126377835647603e+28
```

```
In [80]: plt.boxplot([train_r2,test_r2],labels=['training','testing'])
```

```
Out[80]: {'whiskers': [<matplotlib.lines.Line2D at 0x18f44f8d7f0>,
           <matplotlib.lines.Line2D at 0x18f44f8db70>,
           <matplotlib.lines.Line2D at 0x18f44f96f60>,
           <matplotlib.lines.Line2D at 0x18f44fa02e8>],
          'caps': [<matplotlib.lines.Line2D at 0x18f44f8deb8>,
           <matplotlib.lines.Line2D at 0x18f44f96240>,
           <matplotlib.lines.Line2D at 0x18f44fa0630>,
           <matplotlib.lines.Line2D at 0x18f44fa0978>],
          'boxes': [<matplotlib.lines.Line2D at 0x18f44f8d6a0>,
           <matplotlib.lines.Line2D at 0x18f44f96be0>],
          'medians': [<matplotlib.lines.Line2D at 0x18f44f96588>,
           <matplotlib.lines.Line2D at 0x18f44fa0cc0>],
          'fliers': [<matplotlib.lines.Line2D at 0x18f44f968d0>,
           <matplotlib.lines.Line2D at 0x18f44fa8048>],
          'means': []}
```



*** Based on metrics of LinearRegression performance, we can say that the it's performance on training data is much better than the testing data. We assumed that there is an overfitting for this model.**

Model 2. MLPRegressor

In [176]:

```
#Build the Model MLPRegressor

mlpr = MLPRegressor()
# using r2 score and squred error to evaluate performance using cross validation
# refer to https://scikit-learn.org/stable/modules/generated/sklearn.model_select
train_r2 = [];
train_error = [];
test_r2 = [];
test_error = [];


# using cross validation

kf = KFold(n_splits=5)
kf.get_n_splits(X)

for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    mlpr.fit(X_train,y_train)
    train_prediction = mlpr.predict(X_train)
    test_prediction = mlpr.predict(X_test)

    #using metrics to evaluate the performance on training data for every fold
    train_r2.append(metrics.r2_score(y_train, train_prediction))
    train_error.append(metrics.mean_squared_error(y_train, train_prediction))
    #using metrics to evaluate the performance on testing data for every fold
    test_r2.append(metrics.r2_score(y_test, test_prediction))
    test_error.append(metrics.mean_squared_error(y_test, test_prediction))

# caculate the mean metrics for the whole data set
mean_train_r2 = np.mean(train_r2)
mean_train_error = np.mean(train_error)
mean_test_r2 = np.mean(test_r2)
mean_test_error = np.mean(test_error)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
```

```
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [178]:

```
print("traing data r2 score: ", train_r2)
print("testing data r2 score: ",test_r2)
print("training data squared error:", train_error)
print("testing data squared error:",test_error)


print("average traing data r2 score: ", mean_train_r2)
print("average testing data r2 score: ",mean_test_r2)
print("average training data squared error:", mean_train_error)
print("average data squared error:",mean_test_error)
```

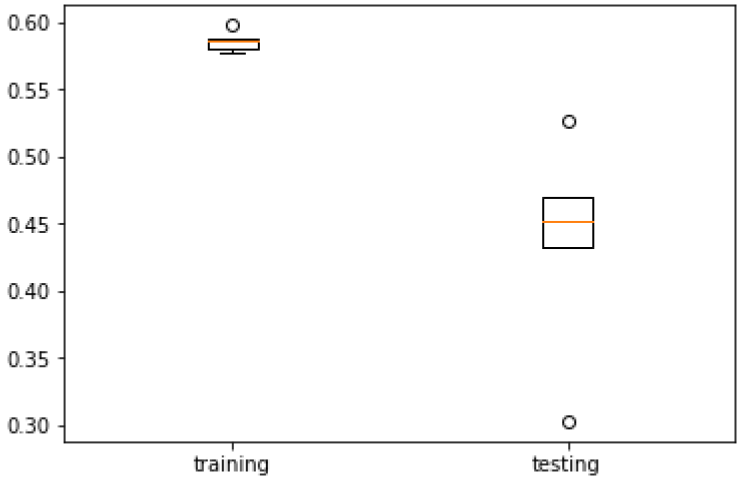
```
traing data r2 score:  [0.5808304971449777, 0.5765575392978717, 0.5977939637080
398, 0.5866449901770276, 0.5871721814314987]
testing data r2 score:  [0.30291503904159967, 0.527252823616845, 0.432274483704
52093, 0.4695337953871568, 0.4517185378792179]
training data squared error: [71315.40279115934, 52202.21990145675, 66663.50073
297031, 65128.43833086457, 59287.72100544402]
testing data squared error: [54135.34902749289, 121183.43103953011, 54810.30590
3341025, 69305.81176418594, 102363.43841404052]
average traing data r2 score:  0.5857998343518831
average testing data r2 score:  0.4367389359258681
average training data squared error: 62919.45655237899
average data squared error: 80359.6672297181
```

In [179]:

```
plt.boxplot([train_r2,test_r2],labels=['training','testing'])
```

Out[179]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x243a4b53828>,
  <matplotlib.lines.Line2D at 0x243a4b53c50>,
  <matplotlib.lines.Line2D at 0x243a4b645c0>,
  <matplotlib.lines.Line2D at 0x243a4b649e8>],
 'caps': [<matplotlib.lines.Line2D at 0x243a4b5b0b8>,
  <matplotlib.lines.Line2D at 0x243a4b5b4e0>,
  <matplotlib.lines.Line2D at 0x243a4b64e10>,
  <matplotlib.lines.Line2D at 0x243a4b6b278>],
 'boxes': [<matplotlib.lines.Line2D at 0x243a4b536d8>,
  <matplotlib.lines.Line2D at 0x243a4b64160>],
 'medians': [<matplotlib.lines.Line2D at 0x243a4b5b908>,
  <matplotlib.lines.Line2D at 0x243a4b6b6a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x243a4b5bd30>,
  <matplotlib.lines.Line2D at 0x243a4b6bac8>],
 'means': []}
```



*** On comparing the results of MLPRegressor algorithm's both train data and test data with Linear regression, MLP performs better. In MLP, training data performs better than the testing data.**

Model 3. SGDRegressor

In [180]:

```
#Build the Model SGDRegressor

sgd = SGDRegressor()
# using r2 score and squred error to evaluate performance using cross validation
# refer to https://scikit-learn.org/stable/modules/generated/sklearn.model_select
train_r2 = [];
train_error = [];
test_r2 = [];
test_error = [];


# using cross validation

kf = KFold(n_splits=5)
kf.get_n_splits(X)

for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    sgd.fit(X_train,y_train)
    train_prediction = sgd.predict(X_train)
    test_prediction = sgd.predict(X_test)

    #using metrics to evaluate the performance on training data for every fold
    train_r2.append(metrics.r2_score(y_train, train_prediction))
    train_error.append(metrics.mean_squared_error(y_train, train_prediction))
    #using metrics to evaluate the performance on testing data for every fold
    test_r2.append(metrics.r2_score(y_test, test_prediction))
    test_error.append(metrics.mean_squared_error(y_test, test_prediction))

# caculate the mean metrics for the whole data set
mean_train_r2 = np.mean(train_r2)
mean_train_error = np.mean(train_error)
mean_test_r2 = np.mean(test_r2)
mean_test_error = np.mean(test_error)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:128: FutureWarning: max_iter and tol parameters have been added in <cla
ss 'sklearn.linear_model.stochastic_gradient.SGDRegressor'> in 0.19. If both ar
e left unset, they default to max_iter=5 and tol=None. If tol is not None, max_
iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and d
efault tol will be 1e-3.
  "and default tol will be 1e-3." % type(self), FutureWarning)
```

In [181]:

```
sgd
```

Out[181]:

```
SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', max_iter=None, n_iter=None, penalty='l2',
       power_t=0.25, random_state=None, shuffle=True, tol=None, verbose=0,
       warm_start=False)
```

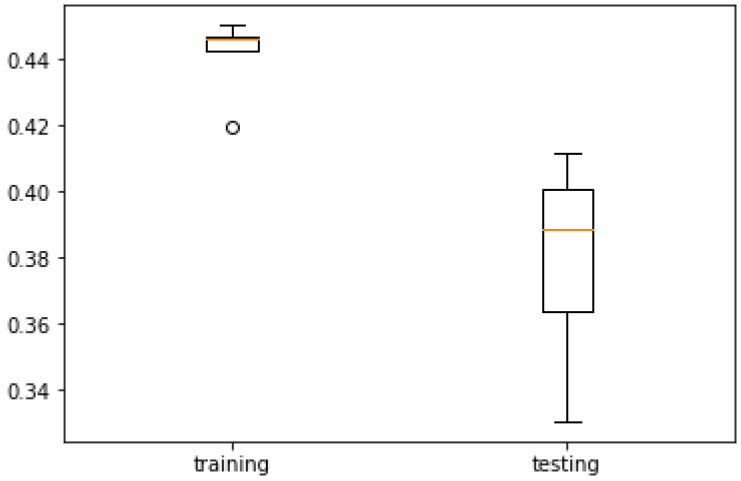
In [182]:

```
print("traing data r2 score: ", train_r2)
print("testing data r2 score: ",test_r2)
print("training data squared error:", train_error)
print("testing data squared error:",test_error)


print("average traing data r2 score: ", mean_train_r2)
print("average testing data r2 score: ",mean_test_r2)
print("average training data squared error:", mean_train_error)
print("average data squared error:",mean_test_error)
```

```
traing data r2 score:  [0.44648334134762224, 0.4194121794533131, 0.450236456746
8673, 0.4459088894397901, 0.4423914859160224]
testing data r2 score:  [0.33062283303440476, 0.36355337498662477, 0.3888327213
903956, 0.4117512167232288, 0.4006985142063699]
training data squared error: [94172.55595778373, 71575.18646106176, 91120.36882
015377, 87302.89428270368, 80080.20905157454]
testing data squared error: [51983.57243985842, 163145.94680971684, 59004.33314
550957, 76855.14946997672, 111888.81071263552]
average traing data r2 score:  0.440886470580723
average testing data r2 score:  0.3790917320682048
average training data squared error: 84850.2429146555
average data squared error: 92575.56251553942
```

```
In [184]: plt.boxplot([train_r2,test_r2],labels=['training','testing'])
```

```
Out[184]: {'whiskers': [<matplotlib.lines.Line2D at 0x243805c0320>,
            <matplotlib.lines.Line2D at 0x243805c07b8>,
            <matplotlib.lines.Line2D at 0x243805d2128>,
            <matplotlib.lines.Line2D at 0x243805d2550>],
           'caps': [<matplotlib.lines.Line2D at 0x243805c0be0>,
            <matplotlib.lines.Line2D at 0x243805c9048>,
            <matplotlib.lines.Line2D at 0x243805d2978>,
            <matplotlib.lines.Line2D at 0x243805d2da0>],
           'boxes': [<matplotlib.lines.Line2D at 0x243805c01d0>,
            <matplotlib.lines.Line2D at 0x243805c9c88>],
           'medians': [<matplotlib.lines.Line2D at 0x243805c9470>,
            <matplotlib.lines.Line2D at 0x243805db208>],
           'fliers': [<matplotlib.lines.Line2D at 0x243805c9898>,
            <matplotlib.lines.Line2D at 0x243805db630>],
           'means': []}
```



* Comparing the results of SGDRegressor with Linear regression and MLPRegressor, MLPRegressor performs best on unseen data.

* g. Can you tune your model to perform better? Explain the technique you’re using and justify why it is improving your results. (25 marks)

*** Yes. Comparing these 3 models, SGDRegressor and MLPRegressor have better performance. So we are going to tune these two models to select the best model for this task. The GridSearchCV is the technique used to tune the model and to choose the best features of the model.**

Tune model SGDRegressor

In [78]:

```
parameters = {'loss':['squared_loss','huber','epsilon_insensitive','squared_epsil
tune_sgd= GridSearchCV(sgd, parameters, cv=5)
tune_sgd.fit(X, y)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
```

```
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
```

```
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\linear_model\stochastic_grad
ient.py:1022: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  ConvergenceWarning)
```

Out[78]:

```
GridSearchCV(cv=5, error_score='raise',
       estimator=SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.
01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='invscaling',
       loss='squared_loss', max_iter=None, n_iter=None, penalty='l2',
       power_t=0.25, random_state=None, shuffle=True, tol=None, verbose=0,
       warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'loss': ['squared_loss', 'huber', 'epsilon_insensitive', 'sq
uared_epsilon_insensitive'], 'max_iter': [1000, 2000, 4000], 'tol': [0.001, 0.0
005, 0.005], 'learning_rate': ['constant', 'optimal', 'invscaling']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [79]:

```
tune_sgd.best_score_
```

Out[79]:

```
0.462304537094318
```

In [80]:

```
tune_sgd.best_estimator_
```

Out[80]:

```
SGDRegressor(alpha=0.0001, average=False, epsilon=0.1, eta0=0.01,
       fit_intercept=True, l1_ratio=0.15, learning_rate='constant',
       loss='squared_loss', max_iter=4000, n_iter=None, penalty='l2',
       power_t=0.25, random_state=None, shuffle=True, tol=0.0005,
       verbose=0, warm_start=False)
```

In [81]:

```
tune_sgd.best_params_
```

Out[81]:

```
{'learning_rate': 'constant',
 'loss': 'squared_loss',
 'max_iter': 4000,
 'tol': 0.0005}
```

In [82]:

```
# using the best estimator to make a better performance
sgd_better = SGDRegressor(learning_rate='constant', loss='squared_loss', max_iter
```

In [83]:

```
train_r2 = [];
train_error = [];
test_r2 = [];
test_error = [];


# validate new model performance


for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    sgd_better.fit(X_train,y_train)
    train_prediction = sgd_better.predict(X_train)
    test_prediction = sgd_better.predict(X_test)

    #using metrics to evaluate the performance on training data for every fold
    train_r2.append(metrics.r2_score(y_train, train_prediction))
    train_error.append(metrics.mean_squared_error(y_train, train_prediction))
    #using metrics to evaluate the performance on testing data for every fold
    test_r2.append(metrics.r2_score(y_test, test_prediction))
    test_error.append(metrics.mean_squared_error(y_test, test_prediction))

# caculate the mean metrics for the whole data set
sgd_mean_train_r2 = np.mean(train_r2)
sgd_mean_train_error = np.mean(train_error)
sgd_mean_test_r2 = np.mean(test_r2)
sgd_mean_test_error = np.mean(test_error)
```

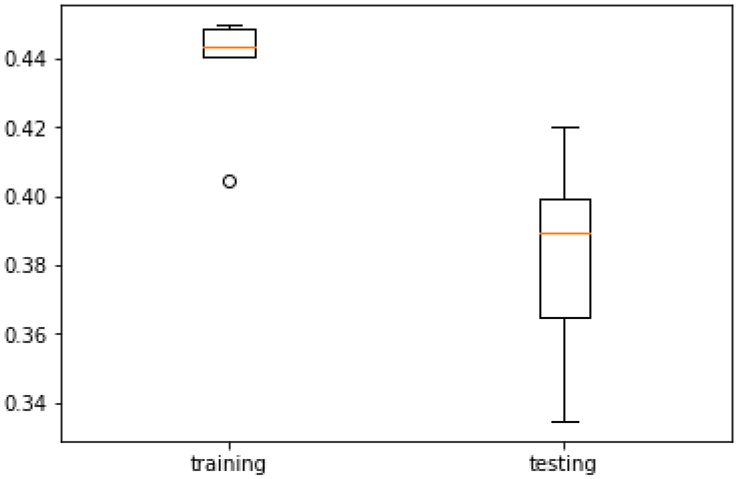
In [85]:

```
print("traing data r2 score: ", train_r2)
print("testing data r2 score: ",test_r2)
print("training data squared error:", train_error)
print("testing data squared error:",test_error)


print("average traing data r2 score: ", sgd_mean_train_r2)
print("average testing data r2 score: ",sgd_mean_test_r2)
print("average training data squared error:", sgd_mean_train_error)
print("average data squared error:",sgd_mean_test_error)
```

```
traing data r2 score:  [0.5583835845695345, 0.5132464288518392, 0.5382126488170
857, 0.5251444372441685, 0.5304126750198285]
testing data r2 score:  [0.43102754115149455, 0.4410113928468544, 0.41257602776
536584, 0.4274234469956165, 0.4134255332425608]
training data squared error: [75134.40823127922, 60007.24848604666, 76538.78521
532766, 74818.49862726017, 67439.16242770717]
testing data squared error: [44186.18155876814, 143290.4535677729, 56712.067167
99879, 74807.56070421205, 109512.69275258957]
average traing data r2 score:  0.5330799549004913
average testing data r2 score:  0.42509278840037845
average training data squared error: 70787.62059752418
average data squared error: 85701.7911502683
```

```
In [90]: plt.boxplot([train_r2,test_r2],labels=['training','testing'])
```

```
Out[90]: {'whiskers': [<matplotlib.lines.Line2D at 0x2438015aeb8>,
            <matplotlib.lines.Line2D at 0x243801d23c8>,
            <matplotlib.lines.Line2D at 0x243801ea358>,
            <matplotlib.lines.Line2D at 0x243801ea780>],
           'caps': [<matplotlib.lines.Line2D at 0x243801d2dd8>,
            <matplotlib.lines.Line2D at 0x243801e2240>,
            <matplotlib.lines.Line2D at 0x243801eaba8>,
            <matplotlib.lines.Line2D at 0x243801eafd0>],
           'boxes': [<matplotlib.lines.Line2D at 0x243801d2940>,
            <matplotlib.lines.Line2D at 0x243801e2ef0>],
           'medians': [<matplotlib.lines.Line2D at 0x243801e2668>,
            <matplotlib.lines.Line2D at 0x243801f1438>],
           'fliers': [<matplotlib.lines.Line2D at 0x243801e2a90>,
            <matplotlib.lines.Line2D at 0x243801f1860>],
           'means': []}
```



Tune model MLPRegressor

In [60]:
```
parameters = {'hidden_layer_sizes':[50,100,200],'learning_rate':['constant','invs
tune_mlp= GridSearchCV(mlpr, parameters, cv=5)
tune_mlp.fit(X, y)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_
perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iteratio
ns (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_
perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iteratio
ns (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_
perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iteratio
ns (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_
perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iteratio
ns (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_
perceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iteratio
ns (200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [61]:
```
tune_mlp.best_score_
```

Out[61]:
```
0.4411326408538944
```

In [62]:
```
tune_mlp.best_estimator_
```

Out[62]:
```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=200, learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

In [63]:
```
tune_mlp.best_params_
```

Out[63]:
```
{'hidden_layer_sizes': 200, 'learning_rate': 'constant'}
```

In [69]:
```
# using the best estimator to make a better performance
mlp_better = MLPRegressor( hidden_layer_sizes=200, learning_rate='constant')
```

In [88]:

```
MLPRegressor()
```

Out[88]:

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

In [70]:

```
train_r2 = [];
train_error = [];
test_r2 = [];
test_error = [];


# validate new model performance


for train_index, test_index in kf.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    mlp_better.fit(X_train,y_train)
    train_prediction = mlp_better.predict(X_train)
    test_prediction = mlp_better.predict(X_test)

    #using metrics to evaluate the performance on training data for every fold
    train_r2.append(metrics.r2_score(y_train, train_prediction))
    train_error.append(metrics.mean_squared_error(y_train, train_prediction))
    #using metrics to evaluate the performance on testing data for every fold
    test_r2.append(metrics.r2_score(y_test, test_prediction))
    test_error.append(metrics.mean_squared_error(y_test, test_prediction))

# caculate the mean metrics for the whole data set
mlp_mean_train_r2 = np.mean(train_r2)
mlp_mean_train_error = np.mean(train_error)
mlp_mean_test_r2 = np.mean(test_r2)
mlp_mean_test_error = np.mean(test_error)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [71]:

```
print("traing data r2 score: ", train_r2)
print("testing data r2 score: ",test_r2)
print("training data squared error:", train_error)
print("testing data squared error:",test_error)


print("average traing data r2 score: ", mlp_mean_train_r2)
print("average testing data r2 score: ",mlp_mean_test_r2)
print("average training data squared error:", mlp_mean_train_error)
print("average data squared error:",mlp_mean_test_error)
```

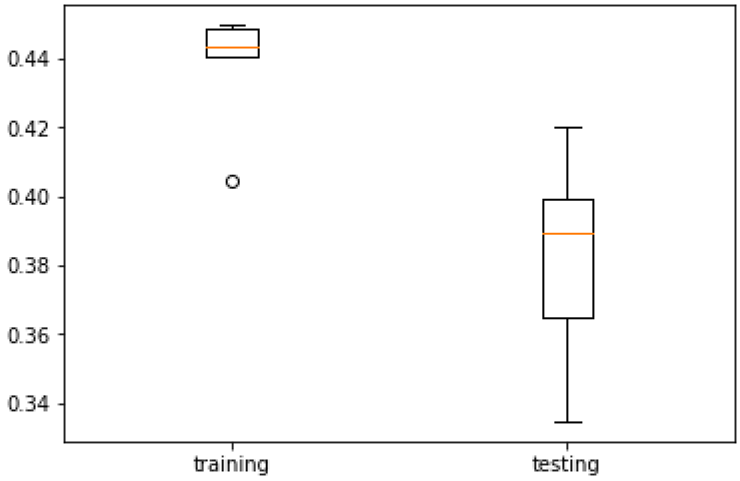
```
traing data r2 score:  [0.6307893869217713, 0.6010322046444477, 0.6348218654886
584, 0.6460861830063669, 0.6264423207457472]
testing data r2 score:  [0.2452819263597309, 0.5253176704813975, 0.460638949070
88554, 0.5073872236230199, 0.4574861111966565]
training data squared error: [62815.647147764066, 49184.969670296574, 60526.323
92615381, 55762.85192329541, 53647.9919009923]
testing data squared error: [58611.11431481773, 121679.48581945412, 52071.89626
894646, 64360.232669550576, 101286.63995035649]
average traing data r2 score:  0.5856508845472577
average testing data r2 score:  0.4314950087494066
average training data squared error: 62918.67183272929
average data squared error: 81105.04175506887
```

In [92]:

```
plt.boxplot([train_r2,test_r2],labels=['training','testing'])
```

Out[92]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x243804d3550>,
  <matplotlib.lines.Line2D at 0x243804d39e8>,
  <matplotlib.lines.Line2D at 0x243804e32e8>,
  <matplotlib.lines.Line2D at 0x243804e3710>],
 'caps': [<matplotlib.lines.Line2D at 0x243804d3e10>,
  <matplotlib.lines.Line2D at 0x243804db278>,
  <matplotlib.lines.Line2D at 0x243804e3b38>,
  <matplotlib.lines.Line2D at 0x243804e3f60>],
 'boxes': [<matplotlib.lines.Line2D at 0x243804d3400>,
  <matplotlib.lines.Line2D at 0x243804dbe48>],
 'medians': [<matplotlib.lines.Line2D at 0x243804db6a0>,
  <matplotlib.lines.Line2D at 0x243804ec3c8>],
 'fliers': [<matplotlib.lines.Line2D at 0x243804dba58>,
  <matplotlib.lines.Line2D at 0x243804ec7f0>],
 'means': []}
```



Results before Tuning - r2 score

Model	Average training r2 score	Average testing r2 score
MLPRegressor	0.5857998343518831	0.4367389359258681
SGDRegressor	0.440886470580723	0.3790917320682048

Results after Tuning - r2 score

Model	Average training r2 score	Average testing r2 score
MLPRegressor	0.5856508845472577	0.4314950087494066
SGDRegressor	0.5330799549004913	0.42509278840037845

Results before Tuning - Average squared error

Model	Average training sq error	Average testing sq error
MLPRegressor	62919.4565523789	80359.6672297181

Model	Average training sq error	Average testing sq error
SGDRegressor	84850.2429146555	92575.5625155394

Results after Tuning - Average squared error

Model	Average training sq error	Average testing sq error
MLPRegressor	62918.6718327292	81105.0417550688
SGDRegressor	70787.6205975241	85701.7911502683

- Based on the results obtained from tuning both the models, it is noticed that the tuning is well performed in SGDRegressor for both training and testing data than the MLPRegressor. Even though after tuning, MLPRegressor appears to be the best model among this three.
- Based on the Average squarred error, For MLPRegressor model, the error doesn't reduce after tuning. But for SGDregressor, the error reduce after tuning. It means that tuning help to improve the SGD model.But even though after tuning, MLPRegressor performs better than SGDRegressor based on the metric error.

h. (Bonus) Use relief feature selection to improve your model.(10 marks)

- After tuning the model, MLP has the best performance on unseen data. We applied feature selection on it in order to improve the model.

In [185]:

```
# feature importance ranking
# using reliefF refer to https://epistasislab.github.io/scikit-rebate/using/
X_train, X_test, y_train, y_test = train_test_split(X, y)
fs = ReliefF()
fs.fit(X_train.values, y_train.values)
```

Out[185]:

```
ReliefF(discrete_threshold=10, n_features_to_select=10, n_jobs=1,
    n_neighbors=100, verbose=False)
```

In [186]:

```
#getting the feature importances and rank it
features = [[]];
for feature_name, feature_score in zip(X.columns,fs.feature_importances_):
    features.append([feature_name,feature_score])

feature_df = pd.DataFrame(data=features[1:])
feature_df.set_index(0,inplace=True)
feature_df.columns = ['importance']
feature_df.sort_values(by=['importance'],ascending=False,inplace=True)
feature_df.head(20)
```

Out[186]:

	importance
0	
listed_in(type)_Delivery	0.196904
listed_in(type)_Dine-out	0.151011
cuisine__North Indian	0.140995
cuisine__ Chinese	0.125823
cuisine__ North Indian	0.086091
cuisine__ Continental	0.066313
cuisine__South Indian	0.050641
cuisine__Continental	0.045113
cuisine__ South Indian	0.044796
cuisine__Chinese	0.042309
cuisine__Cafe	0.042114
cuisine__ Fast Food	0.040931
cuisine__Finger Food	0.032437
location_Electronic City	0.027672
cuisine__ Biryani	0.026243
listed_in(type)_Cafes	0.026170
location_Whitefield	0.023662
cuisine__ Italian	0.023599
listed_in(type)_Desserts	0.020578
cuisine__ Thai	0.018627

In [190]:

```
#select 50, 100, 150, 200, 250 features respectively to compare the performance o
nums = [50, 100, 150,200,250]
feature_selected = [];
for n in nums:
    feature_selected.append(feature_df[:n].index)
```

In [191]:

```
test_sel_r2 = [];
train_sel_r2 = [];
test_sel_error = [];
train_sel_error = [];

def numberFeatureSelecte(n):
    feature_selected = feature_df[:n].index
    data_selected_X = X[feature_selected]
    selected_test_r2 = [];
    selected_test_squred = [];
    selected_train_r2 = [];
    selected_train_squred = [];

    for train_index, test_index in kf.split(data_selected_X):
        X_train_s, X_test_s = data_selected_X.iloc[train_index], data_selected_X.

        y_train_s, y_test_s = y.iloc[train_index], y.iloc[test_index]

        mlp_better.fit(X_train_s,y_train_s)
        train_prediction_selected = mlp_better.predict(X_train_s)
        test_predictions_selected = mlp_better.predict(X_test_s)
        selected_test_r2.append(metrics.r2_score(y_test_s, test_predictions_selec
        selected_test_squred.append(metrics.mean_squared_error(y_test_s, test_pre
        selected_train_r2.append(metrics.r2_score(y_train_s, train_prediction_sel
        selected_train_squred.append(metrics.mean_squared_error(y_train_s, train_
    test_sel_r2.append(np.mean(selected_test_r2))
    train_sel_r2.append(np.mean(selected_train_r2))
    test_sel_error.append(np.mean(selected_test_squred))
    train_sel_error.append(np.mean(selected_train_squred))
```

In [192]:

```
# select the top features and evaluate the performance
for i in np.arange(50,251,50):
    numberFeatureSelecte(i)
```

```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

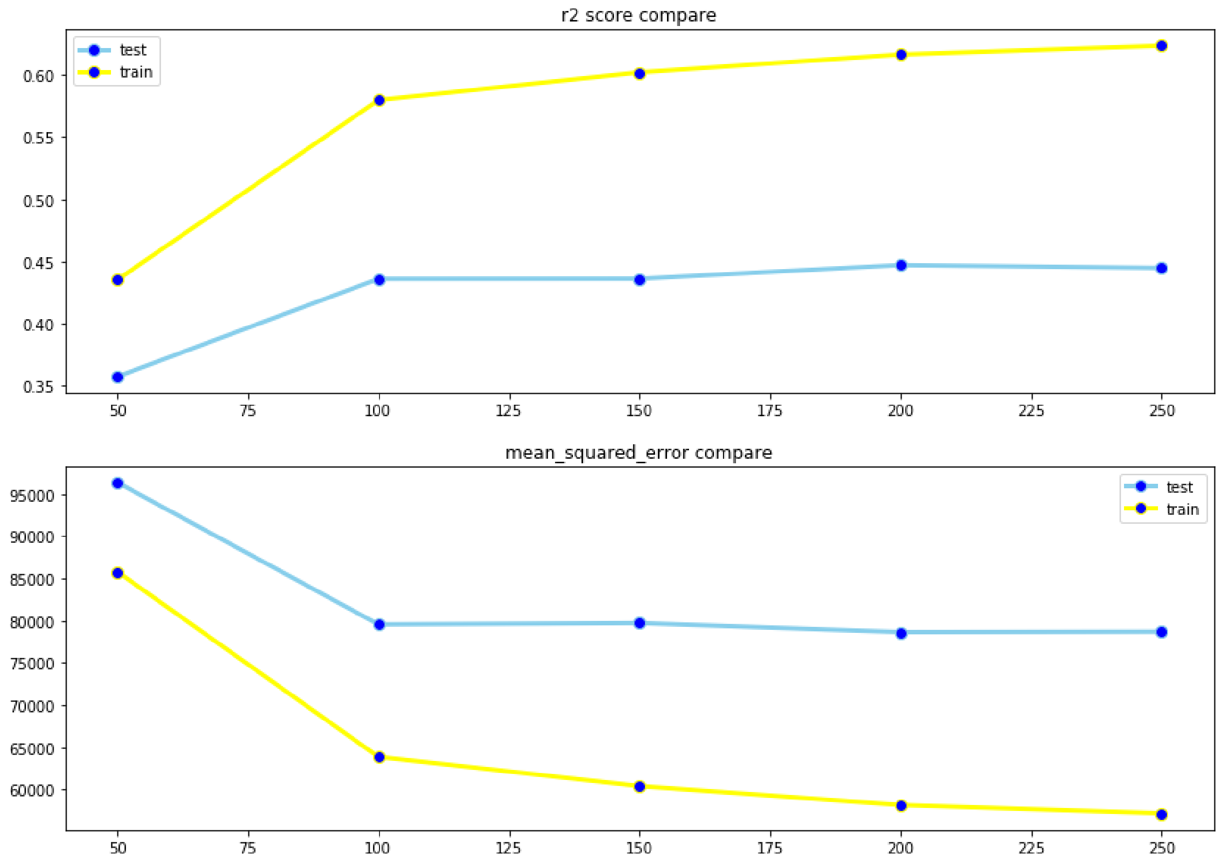
```
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
C:\Users\jamun\Anaconda1\lib\site-packages\sklearn\neural_network\multilayer_pe
rceptron.py:564: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(200) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

In [194]:

```
fig, axes = plt.subplots(2,1,figsize=(14,10))
axes[0].plot(np.arange(50,251,50),test_sel_r2, marker='o', markerfacecolor='blue'
axes[0].plot(np.arange(50,251,50),train_sel_r2, marker='o', markerfacecolor='blue
axes[0].legend()
axes[0].set_title("r2 score compare")
axes[1].plot(np.arange(50,251,50),test_sel_error, marker='o', markerfacecolor='bl
axes[1].plot(np.arange(50,251,50),train_sel_error, marker='o', markerfacecolor='b
axes[1].set_title("mean_squared_error compare")
axes[1].legend()
```

Out[194]:

```
<matplotlib.legend.Legend at 0x24380742e10>
```



From the result, we can see that if the model is fitted on over about 150 most important features selection, it doesn't change much on the performance of data for both r2 score and mean_squared_error

So for this model, we can select about 150 most important features to build the model which almost have the same performance as the whole features.But it can help the model to reduce computing complexity.

References

- [1] Matplotlib.org. (2019). Demo of the histogram (hist) function with a few features — Matplotlib 3.1.0 documentation. [online] Available at: https://matplotlib.org/gallery/statistics/histogram_features.html (https://matplotlib.org/gallery/statistics/histogram_features.html) [Accessed 1 Jul. 2019].
- [2] [Seaborn.pydata.org. (2019). seaborn.distplot — seaborn 0.9.0 documentation. [online] Available at: https://seaborn.pydata.org/generated/seaborn.distplot.html (https://seaborn.pydata.org/generated/seaborn.distplot.html) [Accessed 1 Jul. 2019].
- [3] GeeksforGeeks. (2019). Python | Pandas Dataframe.duplicated() - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/python-pandas-dataframe-duplicated/ (https://www.geeksforgeeks.org/python-pandas-dataframe-duplicated/) [Accessed 1 Jul. 2019].
- [4] Pandas.pydata.org. (2019). pandas.DataFrame.drop — pandas 0.24.2 documentation. [online] Available at: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html) [Accessed 1 Jul. 2019].
- [5] Pandas.pydata.org. (2019). pandas.DataFrame.fillna — pandas 0.22.0 documentation. [online] Available at: https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.DataFrame.fillna.html (https://pandas.pydata.org/pandas-docs/version/0.22/generated/pandas.DataFrame.fillna.html) [Accessed 1 Jul. 2019].
- [6] GeeksforGeeks. (2019). Python | Pandas dataframe.groupby() - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/python-pandas-dataframe-groupby/ (https://www.geeksforgeeks.org/python-pandas-dataframe-groupby/) [Accessed 1 Jul. 2019].
- [7] matplotlib, s. and ElNesr, M. (2019). split title of a figure in matplotlib. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/8598163/split-title-of-a-figure-in-matplotlib (https://stackoverflow.com/questions/8598163/split-title-of-a-figure-in-matplotlib) [Accessed 2 Jul. 2019].
- [8] DataScience Made Simple. (2019). Box plot in Python with matplotlib - DataScience Made Simple. [online] Available at: http://www.datasciencemadesimple.com/box-plot-in-python/ (http://www.datasciencemadesimple.com/box-plot-in-python/) [Accessed 2 Jul. 2019].

In [ ]: