Restaurant Recommender

In this project we are going to assist a restaurant consolidator to build a restaurant recommender which will help a customer find the best restaurants in an area and find out which cuisines are served best so on and so forth. Factors such as delivery options, ratings and price options will be explored and the insights will be presented. Also we will find out what makes a restaurant more appealing to the customer and which factors make the ratings go up or down.

Step 1: Setting up the work environment

We are going to download the necessary packages for our work. We are going to view the dataset and check the datatypes.

In [1]:
```
#installing packages
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sb
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
import os
from collections import Counter
from numpy import mean
from numpy import std
from pandas import read_csv

from sklearn.dummy import DummyClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc
from sklearn.metrics import make_scorer
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.naive_bayes import GaussianNB
from pandas import Series;  from numpy.random import randn
from statsmodels.stats.weightstats import ttest_ind
import scipy.stats as stats
from scipy.stats import ttest_ind

%matplotlib inline
from matplotlib import pyplot as plt
from scipy.stats import shapiro
from scipy import stats
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from datetime import datetime
from datetime import date
import statsmodels.api as sm
from statsmodels.formula.api import ols
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression as lgr

from sklearn import preprocessing as preproc
from sklearn import metrics
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
```

In [2]:
```
#downloading the file
zom= pd.read_excel("C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for ML\\Regression\\data.xlsx")
zom.head()
```

Out[2]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	cuisine count	Average Cost for two	Currency
0	53	Amber	1	New Delhi	N-19, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.220891	28.630197	North Indian, Chinese, Mughlai	3.0	1800	Indian Rupees(INR)
1	55	Berco's	1	New Delhi	G-2/43, Middle Circle, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.217298	28.632452	Chinese, Thai	2.0	1100	Indian Rupees(INR)
2	60	Colonel's Kababz	1	New Delhi	29, Defence Colony Market, Defence Colony, New...	Defence Colony	Defence Colony, New Delhi	77.230591	28.574036	North Indian, Mughlai	2.0	900	Indian Rupees(INR)
3	64	Diva - The Italian Restaurant	1	New Delhi	M-8A, M Block Market, Greater Kailash (GK) 2, ...	Greater Kailash (GK) 2	Greater Kailash (GK) 2, New Delhi	77.243186	28.534202	Italian	1.0	2500	Indian Rupees(INR)
4	65	Drums of Heaven	1	New Delhi	S-14, Green Park Extension, Green Park, New Delhi	Green Park	Green Park, New Delhi	77.205934	28.558018	Chinese, Seafood, Thai	3.0	1800	Indian Rupees(INR)

In [3]:
```
#getting the secondary data
cn= pd.read_excel("C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Datasets for DV\\Country-Code.xlsx")
cn.head()
```

Out[3]:

	Country Code	Country
0	1	India
1	14	Australia
2	30	Brazil
3	37	Canada
4	94	Indonesia

In [4]:
```
#merging secondary into primary
rr=pd.merge(zom, cn, on='Country Code', how='left')
rr.head()
```

Out[4]:

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Average Cost for two	Currency
0	53	Amber	1	New Delhi	N-19, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.220891	28.630197	North Indian, Chinese, Mughlai	...	1800	Indian Rupees(INR)
1	55	Berco's	1	New Delhi	G-2/43, Middle Circle, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.217298	28.632452	Chinese, Thai	...	1100	Indian Rupees(INR)
2	60	Colonel's Kababz	1	New Delhi	29, Defence Colony Market, Defence Colony, New...	Defence Colony	Defence Colony, New Delhi	77.230591	28.574036	North Indian, Mughlai	...	900	Indian Rupees(INR)
3	64	Diva - The Italian Restaurant	1	New Delhi	M-8A, M Block Market, Greater Kailash (GK) 2, ...	Greater Kailash (GK) 2	Greater Kailash (GK) 2, New Delhi	77.243186	28.534202	Italian	...	2500	Indian Rupees(INR)
4	65	Drums of Heaven	1	New Delhi	S-14, Green Park Extension, Green Park, New Delhi	Green Park	Green Park, New Delhi	77.205934	28.558018	Chinese, Seafood, Thai	...	1800	Indian Rupees(INR)

5 rows × 21 columns

In [5]:
```
#examining the dataset
rr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9551 entries, 0 to 9550
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Restaurant ID         9551 non-null   int64
 1   Restaurant Name       9550 non-null   object
 2   Country Code          9551 non-null   int64
 3   City                  9551 non-null   object
 4   Address               9551 non-null   object
 5   Locality              9551 non-null   object
 6   Locality Verbose      9551 non-null   object
 7   Longitude             9551 non-null   float64
 8   Latitude              9551 non-null   float64
 9   Cuisines              9542 non-null   object
 10  cuisine count         9550 non-null   float64
 11  Average Cost for two  9551 non-null   int64
 12  Currency              9551 non-null   object
 13  Has Table booking     9551 non-null   object
 14  Has Online delivery   9551 non-null   object
 15  Price range           9551 non-null   int64
 16  Aggregate rating      9551 non-null   float64
 17  Rating color          9551 non-null   object
 18  Rating text           9551 non-null   object
 19  Votes                 9551 non-null   int64
 20  Country               9551 non-null   object
dtypes: float64(4), int64(5), object(12)
memory usage: 1.6+ MB
```

In [6]:
```
#examining the dataset
rr.shape
```

Out[6]:
```
(9551, 21)
```

Step 2: Data cleaning

In this step, we are going to clean our dataset. We are going to look for null values and replace them with mean and mode. We are going to modify some variables if it is necessary and change datatypes for better analysis. We will also remove outliers from the dataset. Outliers hamper the machine learning algorithms and hence they have to be removed.

In [7]:
```
# column name cleaning

rr.columns = rr.columns.str.replace(' ', '')
rr.columns
```

Out[7]:
```
Index(['RestaurantID', 'RestaurantName', 'CountryCode', 'City', 'Address',
       'Locality', 'LocalityVerbose', 'Longitude', 'Latitude', 'Cuisines',
       'cuisinecount', 'AverageCostfortwo', 'Currency', 'HasTablebooking',
       'HasOnlinedelivery', 'Pricerange', 'Aggregaterating', 'Ratingcolor',
       'Ratingtext', 'Votes', 'Country'],
      dtype='object')
```

In [8]:
```
#checking null values
rr.isna().sum(axis=0)
```

Out[8]:
```
RestaurantID         0
RestaurantName       1
CountryCode          0
City                 0
Address              0
Locality             0
LocalityVerbose      0
Longitude            0
Latitude             0
Cuisines             9
cuisinecount         1
AverageCostfortwo    0
Currency             0
HasTablebooking      0
HasOnlinedelivery    0
Pricerange           0
Aggregaterating      0
Ratingcolor          0
Ratingtext           0
Votes                0
Country              0
dtype: int64
```

In [9]:
```
#drop null
rr=rr.dropna()
rr
```

Out[9]:

	RestaurantID	RestaurantName	CountryCode	City	Address	Locality	LocalityVerbose	Longitude	Latitude	Cuisines	...
0	53	Amber	1	New Delhi	N-19, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.220891	28.630197	North Indian, Chinese, Mughlai	...
1	55	Berco's	1	New Delhi	G-2/43, Middle Circle, Connaught Place, New Delhi	Connaught Place	Connaught Place, New Delhi	77.217298	28.632452	Chinese, Thai	...
2	60	Colonel's Kababz	1	New Delhi	29, Defence Colony Market, Defence Colony, New...	Defence Colony	Defence Colony, New Delhi	77.230591	28.574036	North Indian, Mughlai	...
3	64	Diva - The Italian Restaurant	1	New Delhi	M-8A, M Block Market, Greater Kailash (GK) 2, ...	Greater Kailash (GK) 2	Greater Kailash (GK) 2, New Delhi	77.243186	28.534202	Italian	...
4	65	Drums of Heaven	1	New Delhi	S-14, Green Park Extension, Green Park, New Delhi	Green Park	Green Park, New Delhi	77.205934	28.558018	Chinese, Seafood, Thai	...
...	...	...	...	...	...	...	...	...	...	...	...
9546	18499493	Zombiez	1	New Delhi	E 379, Khosla Complex, Samrat Apartment, Vasun...	Vasundhara Enclave	Vasundhara Enclave, New Delhi	0.000000	0.000000	Chinese, North Indian	...
9547	18500618	Veg. Darbar	1	New Delhi	Near Metro Pillar 682, Uttam Nagar West, Uttam...	Uttam Nagar	Uttam Nagar, New Delhi	77.057052	28.621884	North Indian, Chinese	...
9548	18500628	Grill & Cafe	1	Gurgaon	48/5, Mahrauli Road, Near Government College, ...	Sector 14	Sector 14, Gurgaon	77.040480	28.466274	Chinese, Mughlai, North Indian	...
9549	18500639	Chandni Chowk 2 China	1	Noida	H-38, Sector 51, Noida	Sector 51	Sector 51, Noida	0.000000	0.000000	North Indian, Chinese	...
9550	18500652	Mahek By Greenz	1	Gurgaon	A 201, Belvedere Towers, DLF Phase 2, Gurgaon	DLF Phase 2	DLF Phase 2, Gurgaon	0.000000	0.000000	North Indian	...

9540 rows × 21 columns

In [10]:
```
#dropping useless columns
rr=rr.drop(['Address','Locality','LocalityVerbose','Longitude','Latitude','Ratingcolor','RestaurantID'], axis=1
rr.head()
```

Out[10]:

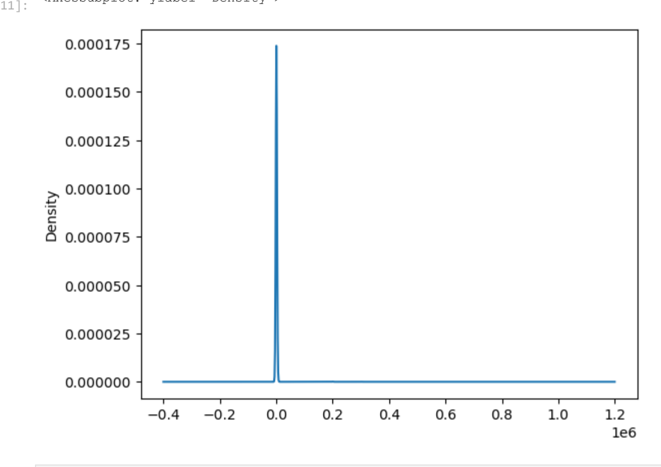
	RestaurantName	CountryCode	City	Cuisines	cuisinecount	AverageCostfortwo	Currency	HasTablebooking	HasOnlinedelivery
0	Amber	1	New Delhi	North Indian, Chinese, Mughlai	3.0	1800	Indian Rupees(INR)	Yes	Yes
1	Berco's	1	New Delhi	Chinese, Thai	2.0	1100	Indian Rupees(INR)	Yes	Yes
2	Colonel's Kababz	1	New Delhi	North Indian, Mughlai	2.0	900	Indian Rupees(INR)	Yes	No
3	Diva - The Italian Restaurant	1	New Delhi	Italian	1.0	2500	Indian Rupees(INR)	Yes	Yes
4	Drums of Heaven	1	New Delhi	Chinese, Seafood, Thai	3.0	1800	Indian Rupees(INR)	Yes	Yes

Univariate analysis

We are using boxplot,histogram and plot to find the outliers,extreme values and distribution of the numerical variables.

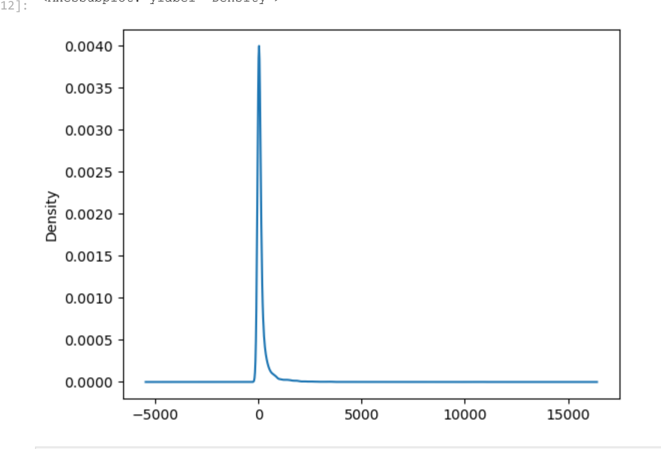
In [11]:
```
#checking distribution for Average Cost for two
rr['AverageCostfortwo'].plot.density()
```

Out[11]:
```
<AxesSubplot: ylabel='Density'>
```



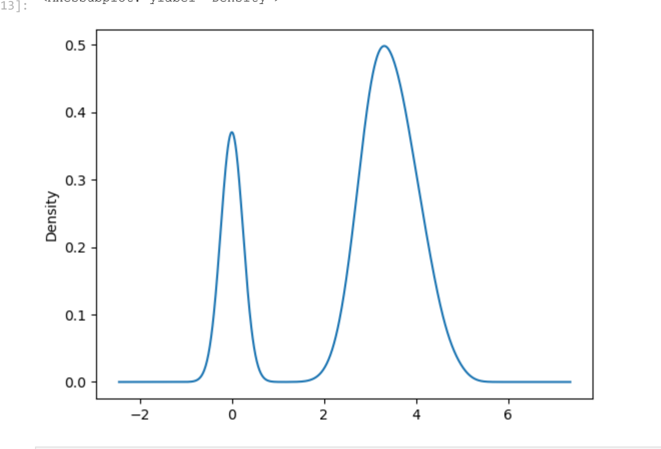
In [12]:
```
#checking distribution for Votes
rr['Votes'].plot.density()
```

Out[12]:
```
<AxesSubplot: ylabel='Density'>
```



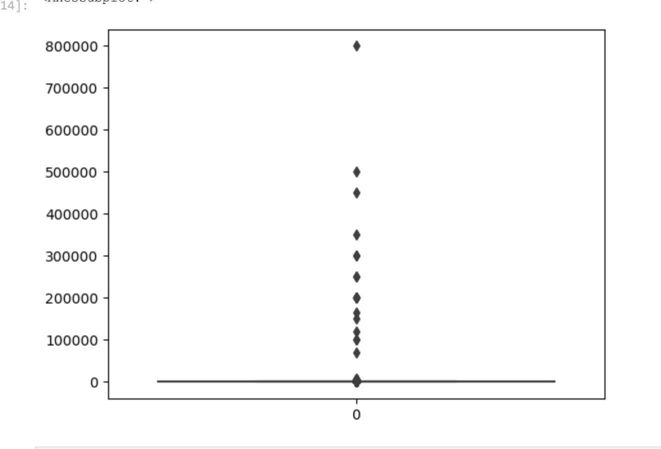
In [13]:
```
#checking distribution for Aggregate rating
rr['Aggregaterating'].plot.density()
```

Out[13]:
```
<AxesSubplot: ylabel='Density'>
```



In [14]:
```
#checking outliers using boxplot Average Cost for two
sb.boxplot(rr['AverageCostfortwo'])
```

Out[14]:
```
<AxesSubplot: >
```



In [15]:
```
#outlier treatment for variable Average Cost for two
rr['AverageCostfortwo'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[15]:
```
0.10     100.0
0.25     250.0
0.50     400.0
0.70     600.0
0.90    1200.0
0.95    1700.0
0.99    3300.0
Name: AverageCostfortwo, dtype: float64
```

In [16]:
```
#seeing limits
AverageCosttwo_HE=rr[rr['AverageCostfortwo'] > 600].copy()
```

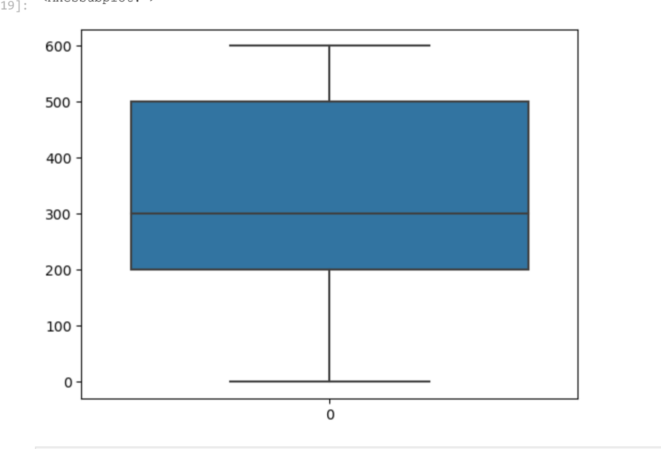
In [17]:
```
#putting them at one place
print(len(AverageCosttwo_HE))
```

```
2636
```

In [18]:
```
#removing outliers
for x in rr['AverageCostfortwo']:
    if x > 600.0:
        rr['AverageCostfortwo'].replace(x,np.nan,inplace=True)
```

In [19]:
```
#checking outliers gone or not
sb.boxplot(rr['AverageCostfortwo'])
```

Out[19]:
```
<AxesSubplot: >
```

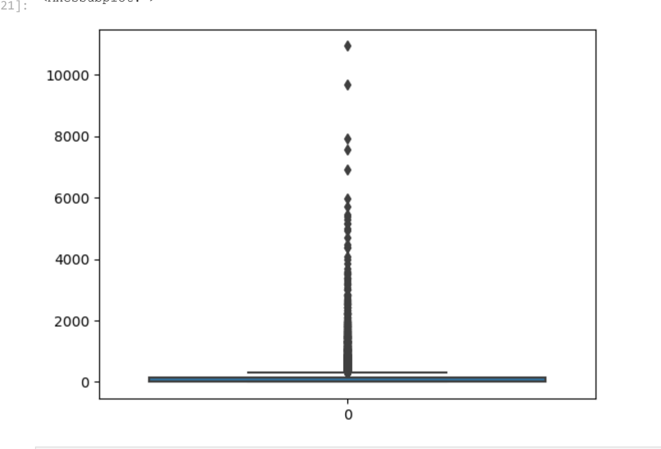


In [20]:
```
#replacing with median values

medacft=rr['AverageCostfortwo'].median()
print(medacft)
rr['AverageCostfortwo'].replace(np.nan, medacft, inplace=True)
```

```
300.0
```

In [21]:
```
#checking outliers using boxplot Votes
sb.boxplot(rr['Votes'])
```

Out[21]:
```
<AxesSubplot: >
```



In [22]:
```
#outlier treatment for variable votes
rr['Votes'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

Out[22]:
```
0.10       0.00
0.25       5.00
0.50      31.00
0.70      97.00
0.90     376.10
0.95     697.00
0.99    1883.49
Name: Votes, dtype: float64
```

In [23]:
```
#seeing limits
vote_HE=rr[rr['Votes'] > 31].copy()
```

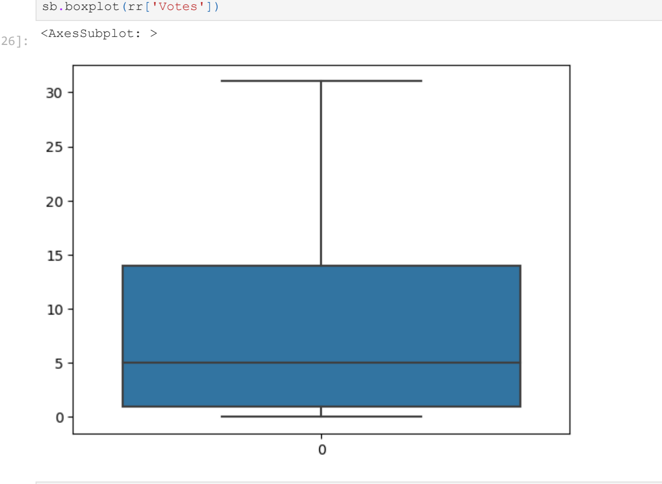
In [24]:
```
#putting them at one place
print(len(vote_HE))
```

```
4712
```

In [25]:
```
#removing outliers
for x in rr['Votes']:
    if x > 31:
        rr['Votes'].replace(x,np.nan,inplace=True)
```

In [26]:
```
#checking if outliers gone
sb.boxplot(rr['Votes'])
```

Out[26]:
```
<AxesSubplot: >
```

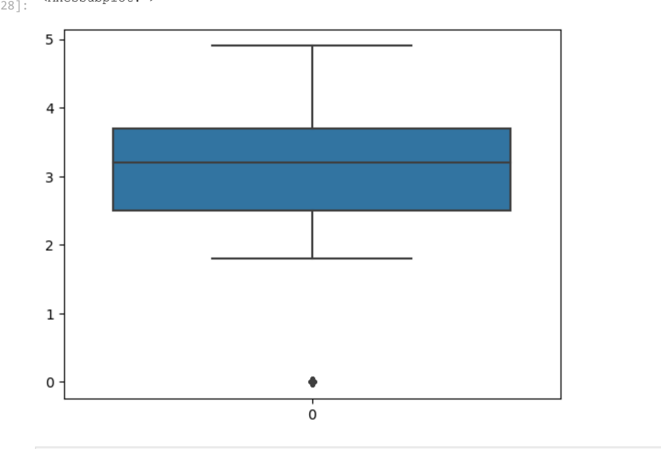


In [27]:
```
#replacing with median values

medvote=rr['Votes'].median()
print(medvote)
rr['Votes'].replace(np.nan, medvote, inplace=True)
```

```
5.0
```

In [28]:
```
#checking outliers using boxplot Aggregate rating
sb.boxplot(rr['Aggregaterating'])
```

Out[28]:
```
<AxesSubplot: >
```



In [29]:
```
#outlier treatment for variable Aggregate rating
rr['Aggregaterating'].quantile([0.1, 0.25, 0.5, 0.7, 0.9, 0.95, 0.99])
```

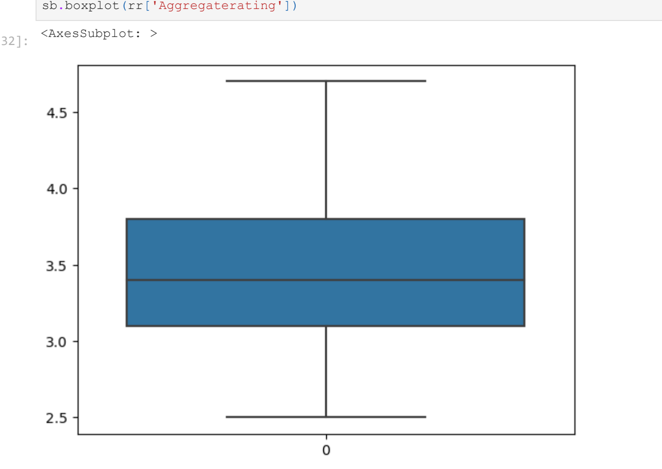
Out[29]:
```
0.10    0.0
0.25    2.5
0.50    3.2
0.70    3.6
0.90    4.1
0.95    4.3
0.99    4.7
Name: Aggregaterating, dtype: float64
```

In [30]:
```
#seeing limits
ar_LE=rr[rr['Aggregaterating'] <2.5].copy()
ar_HE=rr[rr['Aggregaterating'] >4.7].copy()
```

In [31]:
```
#removing outliers
for x in rr['Aggregaterating']:
    if x < 2.5:
        rr['Aggregaterating'].replace(x,np.nan,inplace=True)

for x in rr['Aggregaterating']:
    if x > 4.7:
        rr['Aggregaterating'].replace(x,np.nan,inplace=True)
```

In [32]:
```
#checking if outliers gone
sb.boxplot(rr['Aggregaterating'])
```

Out[32]:
```
<AxesSubplot: >
```



In [33]:
```
#replacing with median values

medar=rr['Aggregaterating'].median()
print(medar)
rr['Aggregaterating'].replace(np.nan, medar, inplace=True)
```

```
3.4
```

Step 3:Exploratory data analysis

In this step, we are going to explore the dataset. Perform hypothesis tests,bivariate analysis and check for correlation between variables.

In [34]:
```
#summary stats
rr.describe().round(2)
```

Out[34]:

	CountryCode	cuisinecount	AverageCostfortwo	Pricerange	Aggregaterating	Votes
count	9540.00	9540.00	9540.00	9540.00	9540.00	9540.00
mean	18.17	1.12	318.56	1.80	3.44	6.73
std	56.45	1.34	144.81	0.91	0.44	6.65
min	1.00	0.00	0.00	1.00	2.50	0.00
25%	1.00	0.00	250.00	1.00	3.20	5.00
50%	1.00	1.00	300.00	2.00	3.40	5.00
75%	1.00	2.00	400.00	2.00	3.70	5.00
max	216.00	8.00	600.00	4.00	4.70	31.00

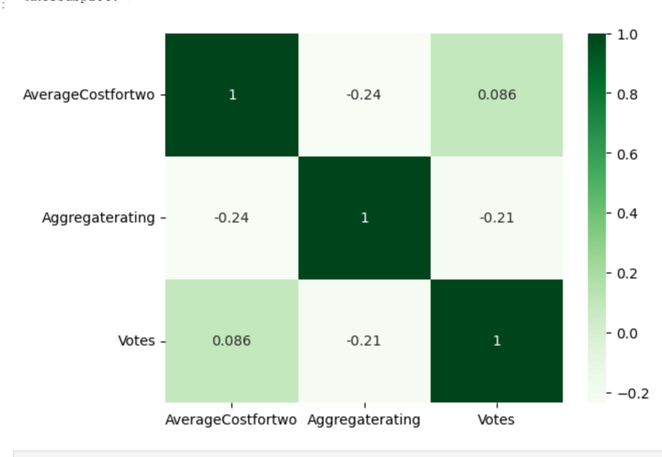
In [35]:
```
#seeing correlation of numerical variables
num_vars=['AverageCostfortwo','Aggregaterating','Votes']
cordata =rr[num_vars].corr()
cordata.round(2)
```

Out[35]:

	AverageCostfortwo	Aggregaterating	Votes
AverageCostfortwo	1.00	-0.24	0.09
Aggregaterating	-0.24	1.00	-0.21
Votes	0.09	-0.21	1.00

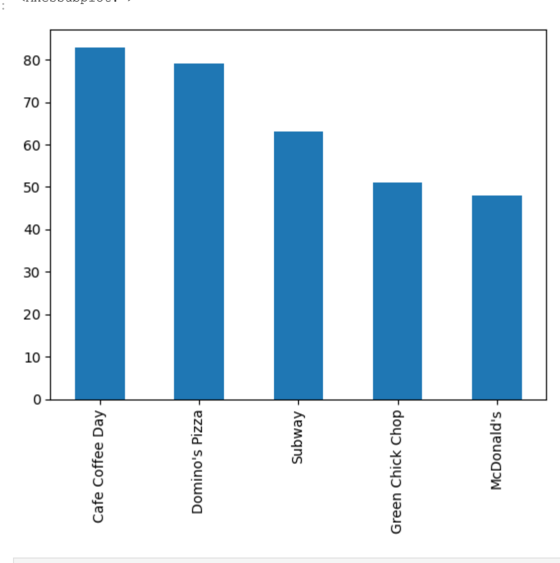
In [36]:
```
#heatmap of correlation
sb.heatmap(cordata, annot=True, cmap="Greens")
```

Out[36]:
```
<AxesSubplot: >
```



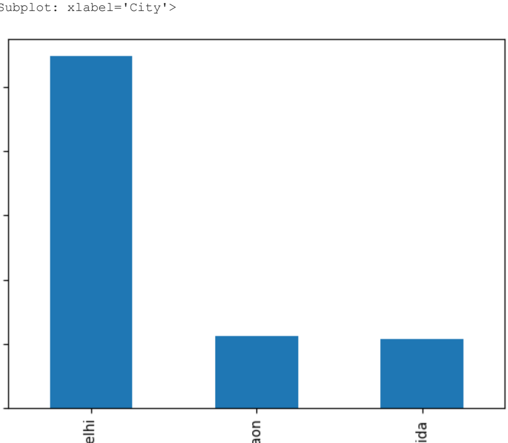
In [37]:
```
#seeing the top franchise
francnt=rr['RestaurantName'].value_counts().head(5)
francnt.plot(kind = "bar")
```

Out[37]:
```
<AxesSubplot: >
```

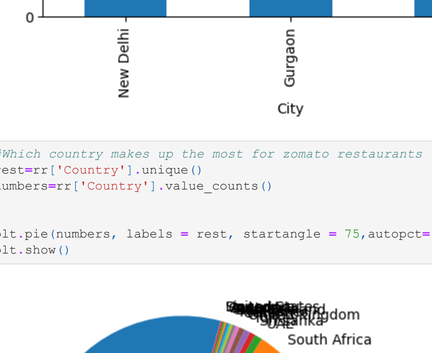


In [38]:
```
#top 3 cities with most restaurants

n_by_city= rr.groupby("City")["RestaurantName"].count().nlargest(3)
n_by_city.plot(kind = "bar")
```

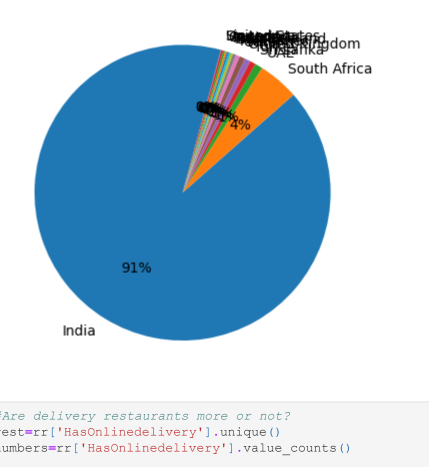
Out[38]: <AxesSubplot: xlabel='City'>



In [120…
```python
#Which country makes up the most for zomato restaurants
rest=rr['Country'].unique()
numbers=rr['Country'].value_counts()


plt.pie(numbers, labels = rest, startangle = 75,autopct='%.0f%%')
plt.show()
```



In [39]:
```python
#Are delivery restaurants more or not?
rest=rr['HasOnlinedelivery'].unique()
numbers=rr['HasOnlinedelivery'].value_counts()


plt.pie(numbers, labels = rest, startangle = 75,autopct='%.2f%%')
plt.show()
```



In [40]:
```python
#Are Table booking restaurants more or not?
bookyes= len(rr[rr['HasTablebooking']=='Yes'])
bookno=len(rr[rr['HasTablebooking']=='No'])

ratio=bookyes/bookno

print("The ratio to restaurants that allow bookings vs dont allow",ratio)
```

```
The ratio to restaurants that allow bookings vs dont allow 0.1381531853972799
```

In [41]:
```python
#difference of votes btw delivery non delivery restaurants
bookyes= rr[rr['HasOnlinedelivery']=='Yes']
bookno=rr[rr['HasOnlinedelivery']=='No']
bookyesvote= bookyes["Votes"].sum()
booknovote= bookno["Votes"].sum()
diff=bookyesvote-booknovote
print("The difference of votes between delivery and non delivery restaurants is",diff)
```

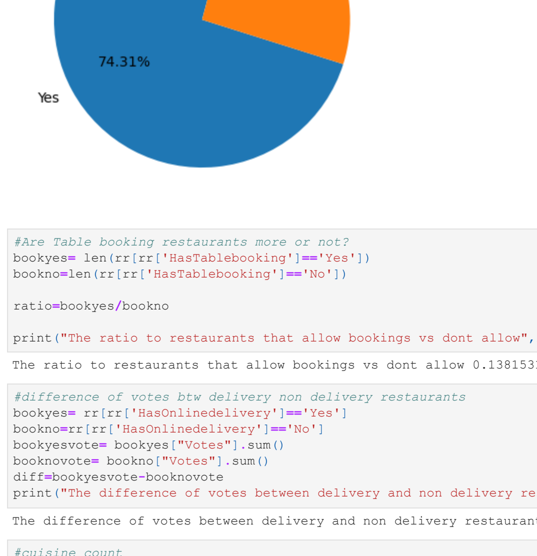
```
The difference of votes between delivery and non delivery restaurants is -27374.0
```

In [42]:
```python
#cuisine count
print("The most number of cuisines offered by a restaurant is",rr['cuisinecount'].max(),
      "and the least is",rr['cuisinecount'].min())
```

```
The most number of cuisines offered by a restaurant is 8.0 and the least is 0.0
```

In [43]:
```python
#Average cost of meals across top restaurants

n_by_city= rr.groupby("RestaurantName")["AverageCostfortwo"].mean().nlargest(7)
n_by_city.plot(kind = "bar")
```

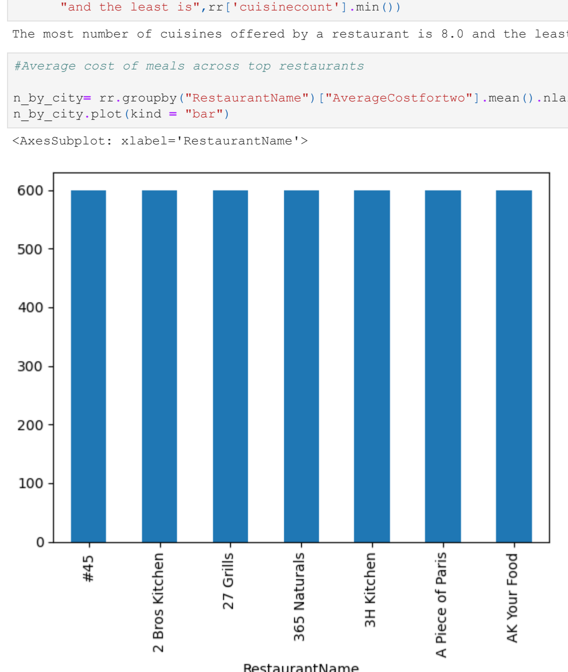
Out[43]: <AxesSubplot: xlabel='RestaurantName'>



In [54]:
```python
#box plot to see distribution of ratings across price range
plt.figure(figsize=[12,6])
rr.boxplot(by="Pricerange",column="Aggregaterating", grid = False)
```

Out[54]: <AxesSubplot: title={'center': 'Aggregaterating'}, xlabel='Pricerange'>

<Figure size 1200x600 with 0 Axes>



In [44]:
```python
#checking if Rating differs according to the Price range of the restaurant
mod = ols("Aggregaterating ~ Pricerange", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

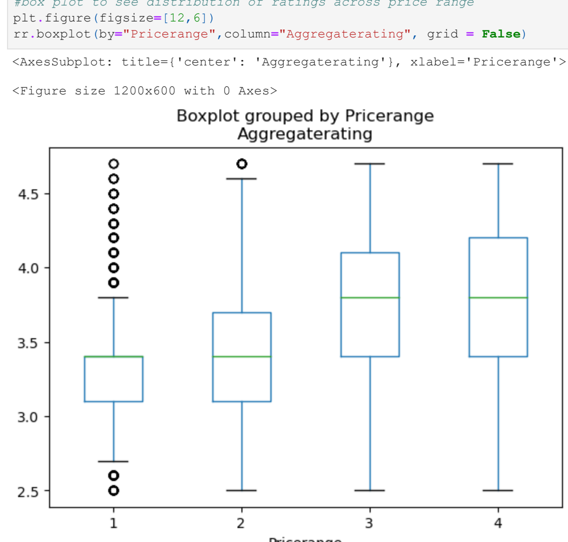
Out[44]:

	df	sum_sq	mean_sq	F	PR(>F)
Pricerange	1.0	264.426240	264.426240	1605.975663	0.0
Residual	9538.0	1570.445642	0.164651	NaN	NaN

In [55]:
```python
#box plot to see distribution of ratings across delivering/non delivering restaurants
plt.figure(figsize=[12,6])
rr.boxplot(by="HasOnlinedelivery",column="Aggregaterating", grid = False)
```

Out[55]: <AxesSubplot: title={'center': 'Aggregaterating'}, xlabel='HasOnlinedelivery'>

<Figure size 1200x600 with 0 Axes>



In [56]:
```python
#checking if Rating differs across delivering/non delivering restaurants
mod = ols("Aggregaterating ~ HasOnlinedelivery", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

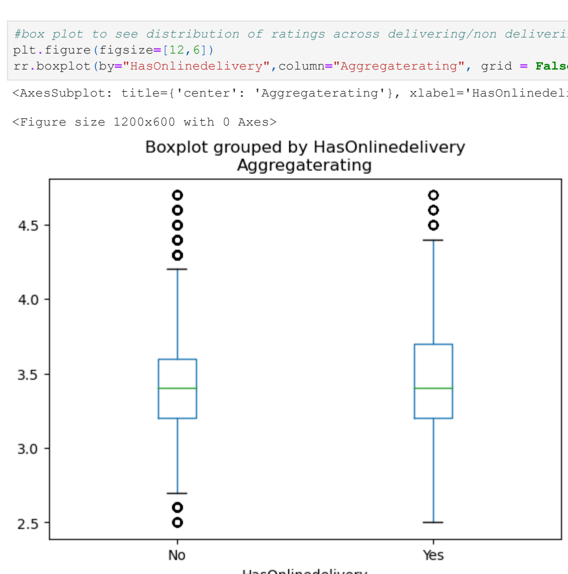
Out[56]:

	df	sum_sq	mean_sq	F	PR(>F)
HasOnlinedelivery	1.0	0.152289	0.152289	0.791691	0.373612
Residual	9538.0	1834.719594	0.192359	NaN	NaN

In [57]:
```python
#box plot to see distribution of ratings across online/non online booking restaurants
plt.figure(figsize=[12,6])
rr.boxplot(by="HasTablebooking",column="Aggregaterating", grid = False)
```

Out[57]: <AxesSubplot: title={'center': 'Aggregaterating'}, xlabel='HasTablebooking'>

<Figure size 1200x600 with 0 Axes>



In [58]:
```python
#checking if Rating differs across online/non online booking restaurants
mod = ols("Aggregaterating ~ HasTablebooking", data = rr).fit()
anov_table = sm.stats.anova_lm(mod)
anov_table
```

Out[58]:

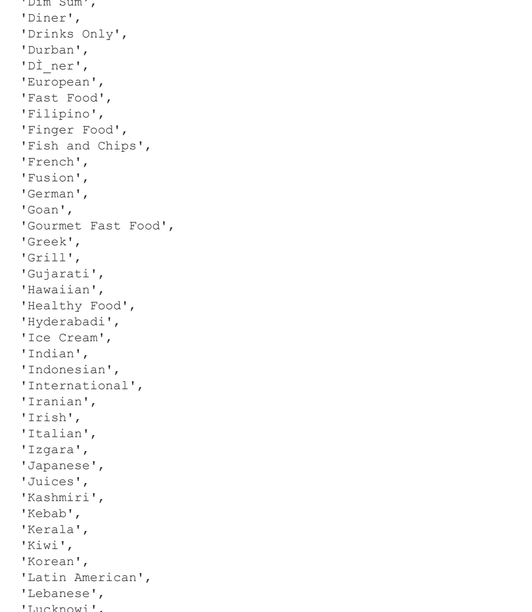
	df	sum_sq	mean_sq	F	PR(>F)
HasTablebooking	1.0	28.970611	28.970611	153.010408	7.049106e-35
Residual	9538.0	1805.901271	0.189338	NaN	NaN

In [90]:
```python
#number of distinct cuisines
list_all = rr['Cuisines'].str.split(r'(?:,|;)\s*').dropna().to_numpy()

list_unique = np.unique(sum(list_all, []))
list_unique=list(list_unique)
list_unique
```

Out[90]:
```
['Afghani',
 'African',
 'American',
 'Andhra',
 'Arabian',
 'Argentine',
 'Armenian',
 'Asian',
 'Asian Fusion',
 'Assamese',
 'Australian',
 'Awadhi',
 'BBQ',
 'Bakery',
 'Bar Food',
 'Belgian',
 'Bengali',
 'Beverages',
 'Bihari',
 'Biryani',
 'Brazilian',
 'Breakfast',
 'British',
 'Bubble Tea',
 'Burger',
 'Burmese',
 'BÌ_rek',
 'Cafe',
 'Cajun',
 'Canadian',
 'Cantonese',
 'Caribbean',
 'Charcoal Grill',
 'Chettinad',
 'Chinese',
 'Coffee and Tea',
 'Contemporary',
 'Continental',
 'Cuban',
 'Cuisine Varies',
 'Curry',
 'Deli',
 'Desserts',
 'Dim Sum',
 'Diner',
 'Drinks Only',
 'Durban',
 'DÌ_ner',
 'European',
 'Fast Food',
 'Filipino',
 'Finger Food',
 'Fish and Chips',
 'French',
 'Fusion',
 'German',
 'Goan',
 'Gourmet Fast Food',
 'Greek',
 'Grill',
 'Gujarati',
 'Hawaiian',
 'Healthy Food',
 'Hyderabadi',
 'Ice Cream',
 'Indian',
 'Indonesian',
 'International',
 'Iranian',
 'Irish',
 'Italian',
 'Izgara',
 'Japanese',
 'Juices',
 'Kashmiri',
 'Kebab',
 'Kerala',
 'Kiwi',
 'Korean',
 'Latin American',
 'Lebanese',
 'Lucknowi',
 'Maharashtrian',
 'Malay',
 'Malaysian',
 'Malwani',
 'Mangalorean',
 'Mediterranean',
 'Mexican',
 'Middle Eastern',
 'Mineira',
 'Mithai',
 'Modern Australian',
 'Modern Indian',
 'Moroccan',
 'Mughlai',
 'Naga',
 'Nepalese',
 'New American',
 'North Eastern',
 'North Indian',
 'Oriya',
 'Pakistani',
 'Parsi',
 'Patisserie',
 'Peranakan',
 'Persian',
 'Peruvian',
 'Pizza',
 'Portuguese',
 'Pub Food',
 'Rajasthani',
 'Ramen',
 'Raw Meats',
 'Restaurant Cafe',
 'Salad',
 'Sandwich',
 'Scottish',
 'Seafood',
 'Singaporean',
 'Soul Food',
 'South African',
 'South American',
 'South Indian',
 'Southern',
 'Southwestern',
 'Spanish',
 'Sri Lankan',
 'Steak',
 'Street Food',
 'Sunda',
 'Sushi',
 'Taiwanese',
 'Tapas',
 'Tea',
 'Teriyaki',
 'Tex-Mex',
 'Thai',
 'Tibetan',
 'Turkish',
 'Turkish Pizza',
 'Vegetarian',
 'Vietnamese',
 'Western',
 'World Cuisine']
```

In [94]:
```python
#most popular cuisine type
francnt=rr['Cuisines'].value_counts().head(5)
francnt.plot(kind = "bar")
```

Out[94]: <AxesSubplot: >



Insights

1.The most popular dishes are chinese and North indian.

2.The most number of restaurants are in India (around 91%).

3.The average cost of meals is mostly same everywhere.

4.New delhi has the most restaurants.

5.CCD, MCD and Subway are some top franchises.

6.The ratings go down when the votes are more and the cost is more.

7.Non delivery restaurants have higher number of votes than delivery ones.

8.Online booking restaurants are 13% to 14%.

9.Around 3/4th of restaurants have home delivery.

10.There are around 143 different cuisines.

11.The ratings doesnt depend on home delivery options but on table booking options and price level.

12.Restaurants can offer upto at most 8 types of cuisines.

Step 4: Model Building

After the data has been cleaned and formatted. Now its time to analyse and get insights. We will uuse Linear Regression to get the factors affecting ratings.

In [95]:
```python
rr.head()
```

Out[95]:

	RestaurantName	CountryCode	City	Cuisines	cuisinecount	AverageCostfortwo	Currency	HasTablebooking	HasOnlinedelivery	Priceran
0	Amber	1	New Delhi	North Indian, Chinese, Mughlai	3.0	300.0	Indian Rupees(INR)	Yes	Yes	
1	Berco's	1	New Delhi	Chinese, Thai	2.0	300.0	Indian Rupees(INR)	Yes	Yes	
2	Colonel's Kababz	1	New Delhi	North Indian, Mughlai	2.0	300.0	Indian Rupees(INR)	Yes	No	
3	Diva - The Italian Restaurant	1	New Delhi	Italian	1.0	300.0	Indian Rupees(INR)	Yes	Yes	
4	Drums of Heaven	1	New Delhi	Chinese, Seafood, Thai	3.0	300.0	Indian Rupees(INR)	Yes	Yes	

In [98]:
```python
#dropping not needed variables
rr1=rr.drop(['RestaurantName','City','Ratingtext'],axis=1)
rr1.head(0)
```

Out[98]:

CountryCode	Cuisines	cuisinecount	AverageCostfortwo	Currency	HasTablebooking	HasOnlinedelivery	Pricerange	Aggregaterating	Votes

In [96]:
```python
#creating target variable
target=rr['Aggregaterating']
```

In [97]:
```python
#removing target variable from main dataframe
rr1= rr.drop(['Aggregaterating'], axis=1)
```

In [101…
```python
#defining category variables
category_variables=['CountryCode', 'cuisinecount','Cuisines',
        'HasTablebooking', 'HasOnlinedelivery', 'Pricerange', 'Country']
category_variables
```

Out[101]:
```
['CountryCode',
 'cuisinecount',
 'Cuisines',
 'HasTablebooking',
 'HasOnlinedelivery',
 'Pricerange',
 'Country']
```

In [105…
```python
#initalizing encoder
label_encoder=preproc.LabelEncoder()
```

In [106…
```python
#encoding categorical variables
cusenc=label_encoder.fit_transform(rr1['Cuisines'])
odenc=label_encoder.fit_transform(rr1['HasOnlinedelivery'])
tbenc=label_encoder.fit_transform(rr1['HasTablebooking'])
cntenc=label_encoder.fit_transform(rr1['Country'])
```

In [107…
```python
# Create linear regression object
regressor = LinearRegression()
```

In [110…
```python
#setting values of S and y
X=pd.DataFrame([cusenc,
                odenc,
                tbenc,
                cntenc,
                rr1['CountryCode'],
                rr1['cuisinecount'],
                rr1['Pricerange']]).T
y= target.values
```

In [111…
```python
# Train, test, split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = .30, random_state= 1000)
```

In [112…
```python
# Fit model to training data
regressor.fit(X_train,y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=-2, normalize=False)
```

Out[112]:
```
LinearRegression
LinearRegression(n_jobs=-2, normalize=False)
```

In [113…
```python
# Predict
# Predicting test set results
y_pred = regressor.predict(X_test)
y_pred
```

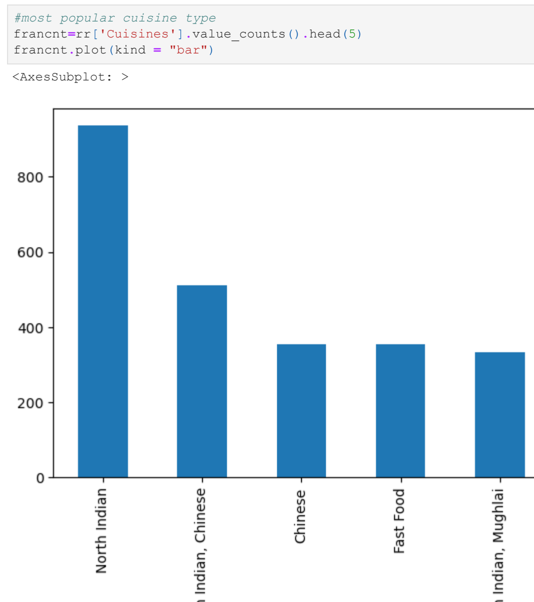
Out[113]:
```
array([3.79473488, 3.44930975, 3.57030208, ..., 3.28853989, 3.76064075,
       3.2468455 ])
```

In [114…
```python
# Calculated R Squared
print('R^2 =',metrics.explained_variance_score(y_test,y_pred))
```

```
R^2 = 0.2577181190893376
```

In [115…
```python
# Actual v predictions scatter
plt.scatter(y_test, y_pred)
```

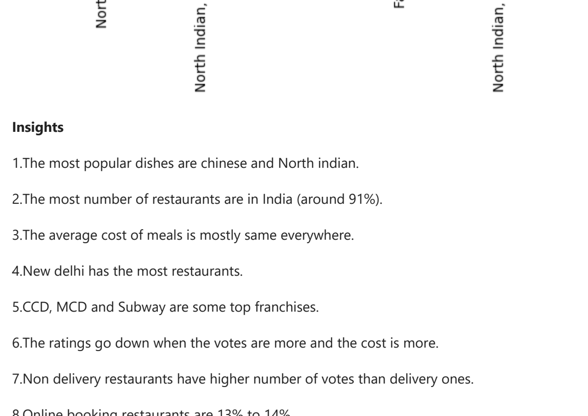
Out[115]: <matplotlib.collections.PathCollection at 0x1860c433e20>



In [116…
```python
# Histogram of the distribution of residuals
sb.distplot((y_test - y_pred))
```

```
C:\Users\sujoydutta\AppData\Local\Temp\ipykernel_5948\225282534.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sb.distplot((y_test - y_pred))
```

Out[116]: <AxesSubplot: ylabel='Density'>



In [117…
```python
#coefficients
cdf = pd.DataFrame(data = regressor.coef_, index = X.columns, columns = ['Coefficients'])
cdf
```

Out[117]:

	Coefficients
0	-0.000079
1	0.016249
2	-0.024005
3	-0.015809
4	0.003059
5	-0.022538
6	0.155563

Conclusion

The model is not very good as it can explain only 25% of the change in the dependent variable. The fit is not achieved. The variables are not enough and maybe another methodology needs to be used to develop this model.