Xtern FoodieX Report

Introduction

For this project, I was given a dataset of Restaurants information. The dataset contains a total of 10 columns (variables). The total number of observations in the dataset are 2019. The following report discusses **four** conclusions made from the given data. Each conclusion tries to identify some features and insights into the delivery business.

The original dataset looks like the following image:

	Restaurant	Latitude	Longitude	Cuisines	Average_Cost	Minimum_Order	Rating	Votes	Reviews	Cook_Time
0	ID_6321	39.262605	-85.837372	Fast Food, Rolls, Burger, Salad, Wraps	$20.00	$50.00	3.5	12	4	30 minutes
1	ID_2882	39.775933	-85.740581	Ice Cream, Desserts	$10.00	$50.00	3.5	11	4	30 minutes
2	ID_1595	39.253436	-85.123779	Italian, Street Food, Fast Food	$15.00	$50.00	3.6	99	30	65 minutes
3	ID_5929	39.029841	-85.332050	Mughlai, North Indian, Chinese	$25.00	$99.00	3.7	176	95	30 minutes
4	ID_6123	39.882284	-85.517407	Cafe, Beverages	$20.00	$99.00	3.2	521	235	65 minutes

Data Cleaning

The first step in analyzing and visualizing the data is to clean the data. Looking at the “Rating”, “Votes” and “Reviews” columns, it was found that some of the rows did not have any values, e.g. they had ‘-’. So, all the instances where the entry was ‘-’, was replaced by “0”. I saw that there were 2 string Ratings: "NEW" and "Opening Soon". For our scoring method, I did not consider the rows that have the Rating as "NEW" or "Opening Soon". So, I will remove those from the dataset for all other parts too for consistency.

```
# Replace new, opening soon, and - ratings / votes with 0
df['Rating'] = df['Rating'].str.replace("NEW","0").str.replace("Opening Soon", "0").str.replace("-","0")
df['Votes'] = df['Votes'].str.replace("NEW","0").str.replace("-","0")
df['Reviews'] = df['Reviews'].str.replace("-","0")
```

The “$” sign with the “Avergae_Cost” and “Minimum_Order” was stripped from each entry. Additionally, the “minutes” term with the “Cook_Time” entries were also removed. Moreover, the datatypes of all the columns was also corrected as per the requirement of this project. Finally, the “Cuisines” column was also cleaned to convert the values to a list and remove all the leading or trailing spaces with each word in the row, if it may exist.

```
df['Average_Cost'] = df['Average_Cost'].str.replace("$","").str.replace(",",".").astype("float32")
```

```
df['Minimum_Order'] = df['Minimum_Order'].str.replace("$","").astype("float32")
```

```
df['Rating'] = df['Rating'].astype("float32")
```

```
df['Votes'] = df['Votes'].astype("int32")
```

```
df['Reviews'] = df['Reviews'].astype("int32")
```

```
df["Cook_Time"] = df['Cook_Time'].str.replace("minutes","").str.strip()
```

```
li = df['Cuisines'].str.split(",").tolist()
strip_li = []

for x in li:
  strip_li.append(list(map(str.strip,x)))

df['Cuisines'] = strip_li
```

The cleaned dataset now looks as following:

	Restaurant	Latitude	Longitude	Cuisines	Average_Cost	Minimum_Order	Rating	Votes	Reviews	Cook_Time
0	ID_6321	39.262605	-85.837372	[Fast Food, Rolls, Burger, Salad, Wraps]	20.0	50.0	3.5	12	4	30
1	ID_2882	39.775933	-85.740581	[Ice Cream, Desserts]	10.0	50.0	3.5	11	4	30
2	ID_1595	39.253436	-85.123779	[Italian, Street Food, Fast Food]	15.0	50.0	3.6	99	30	65
3	ID_5929	39.029841	-85.332050	[Mughlai, North Indian, Chinese]	25.0	99.0	3.7	176	95	30
4	ID_6123	39.882284	-85.517407	[Cafe, Beverages]	20.0	99.0	3.2	521	235	65

Conclusion 1

Find the Trending Restaurants based on their Rating and Votes. For simplicity I will display the Top 10 Trending Restaurants from the list.
Since the dataset has been cleaned, I will only need to Sort the Data with respect to Rating and Votes.

```
# sort the dataset based on the Rating and Votes
# the data is sorted with both Rating and Votes in descending order
con1_df = df
con1_df = con1_df.sort_values(["Rating", "Votes"], ascending = (False, False))
```

Now that the data is sorted, I will extract the 3 relevant columns: “Restaurant”, “Rating”, “Votes”, that will display the Top Trending Restaurant. The scoring algorithm is simple, it compares the Rating of the Restaurant and the number of Votes that the restaurant received.

```
# extract the 3 relevant columns that shows the Top 10 Restaurents
con1_df = con1_df[['Restaurant' ,'Rating', 'Votes']]
con1_df = con1_df[:10]
```

Top 10 Most Trending Restaurants based on Rating and Votes

	Restaurant	Rating	Votes
1325	ID_4728	4.8	650
169	ID_7412	4.8	326
1180	ID_1064	4.7	9054
1428	ID_2051	4.7	3975
1501	ID_7924	4.7	1112
35	ID_1160	4.7	914
325	ID_383	4.7	707
144	ID_6537	4.7	706
225	ID_6278	4.7	441
1803	ID_2201	4.7	129

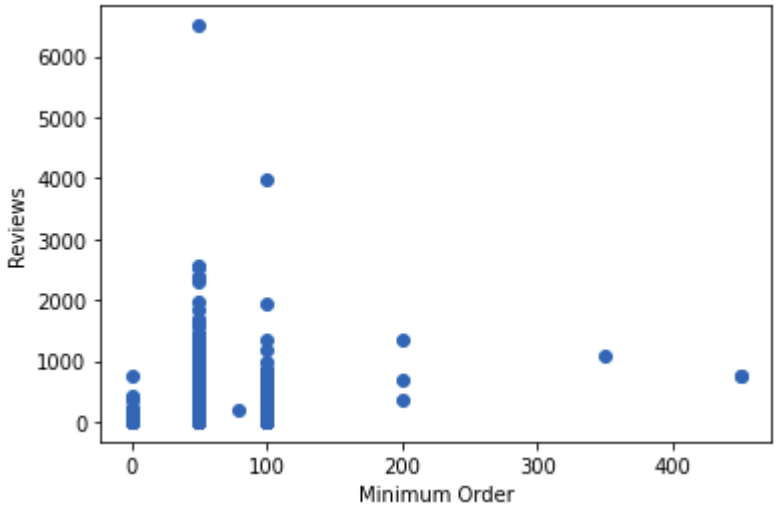
Conclusion 2

Finding out the relationship between Reviews and Minimum Order. I will be seeing how the minimum order price can relate to the number of reviews for a restaurant and in turn lead to deciding the number of orders that a particular restaurant receives.
For this purpose, I will plot a scatter plot and visualize the results.

The following command plots a scatter plot.

```
plt.scatter(con2_df['Minimum_Order'], con2_df['Reviews'])
plt.xlabel('Minimum Order')
_ =plt.ylabel('Reviews')
```

The scatter plot obtained looks like this:



From the plot I can deduce that as the minimum order price increases, the number of reviews per restaurant also decreases. **It can be safely said that since the number of reviews decreases, people are less prone to order from restaurant from pricey restaurants**.

Conclusion 3

Figuring the optimal number of pickup zones for the FoodieX Delivery Service. The optimal number of zones will be represented by clusters. Each cluster will have the optimal number of restaurants so that all the restaurants are spread evenly. At the end the model can predict that to which cluster a new restaurant will belong to.

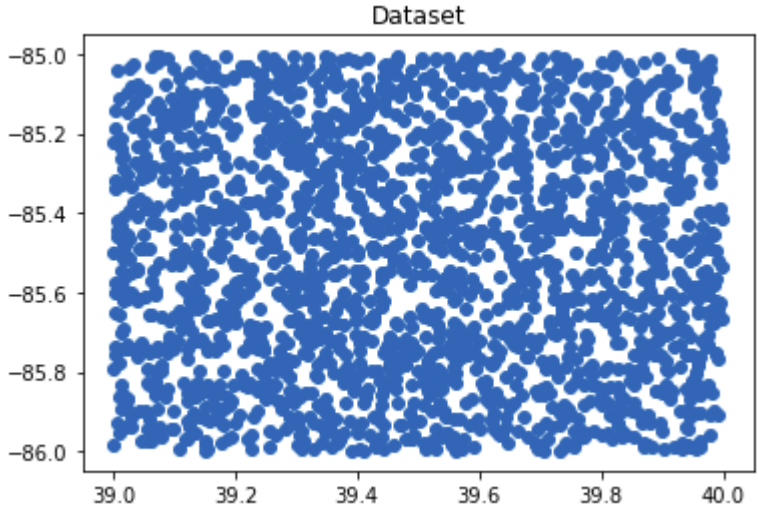
I will use the latitude and longitude columns in this part. I will plot them and then run K-Means Clustering Algorithm from the sklearn library. The K-Means algorithm will be used to define and make clusters to identify the most optimal zones for pickups.

Firstly, I will prepare the data for this part.

```
x1 = np.array(con2_df['Latitude']) # create a new variable to store all the latitudes
x2 = np.array(con2_df['Longitude']) # create a new variable to store all the longitudes
X = np.array(list(zip(x1,x2))) # join the 2 arrays to make a lists of lists and keep them in together

# plot a scatter plot to show all the points
plt.plot()
plt.title('Dataset')
plt.scatter(x1, x2)
plt.show()
```

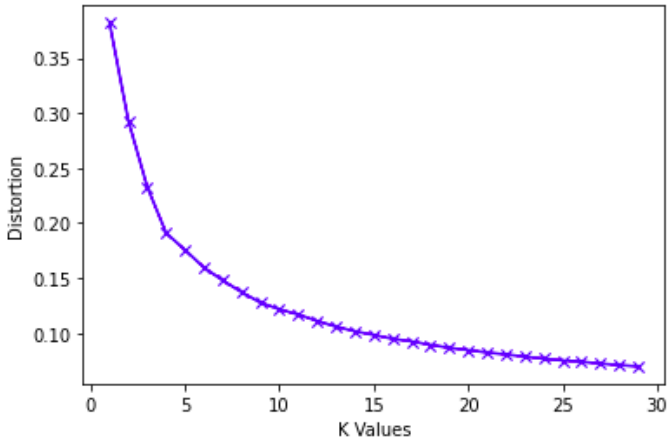
The scatter plot of the latitude and longitude is displayed below.



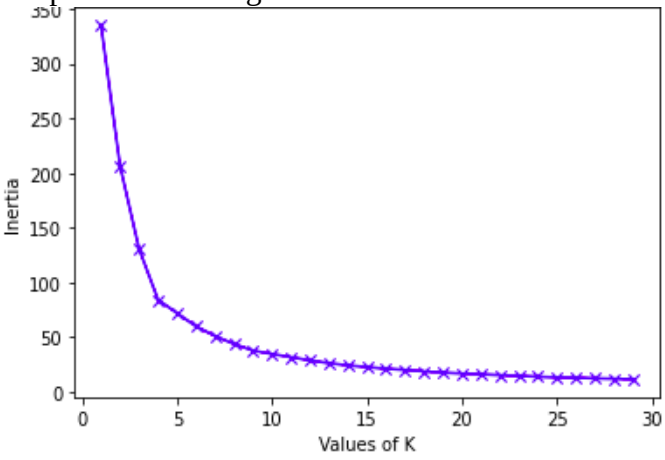
I used the Elbow Method to find the optimal value of k. k is the number of clusters that our dataset will be divided into. I used elbow method because our data does not have easily identifiable clusters, as seen in the scatter plot above.

In the Elbow Method I computed the distortion and inertia of our K-Means model. The distortion computes the sum of squared distances from each point to its assigned center. Inertia is the sum of squared distances of samples to their closest cluster center. I provided a range for the value of k and compute the distortion and inertia for each k.

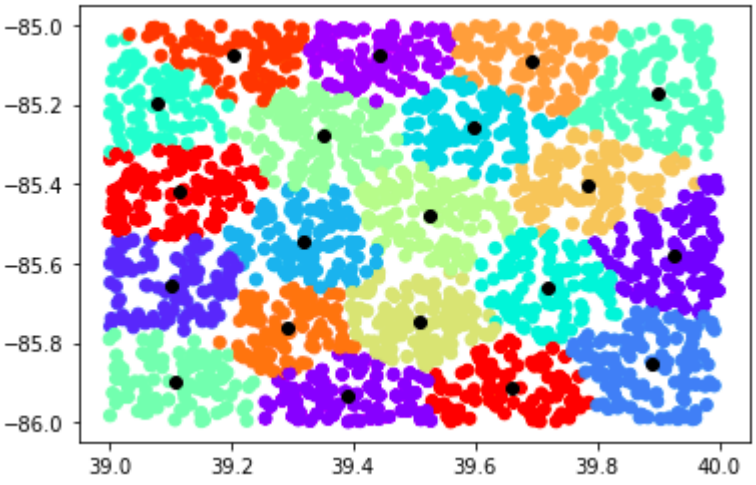
The plot of distortion against the value of k is shown below.



The plot of inertia against the value of k is shown below.



From the above graphs, it can be seen that the optimal value of k is 20. This means that we will divide our data in 20 clusters. Each cluster will represent a zone for pick up. The plot below shows our data divided into 20 clusters. The black dots represent the centers of each cluster.



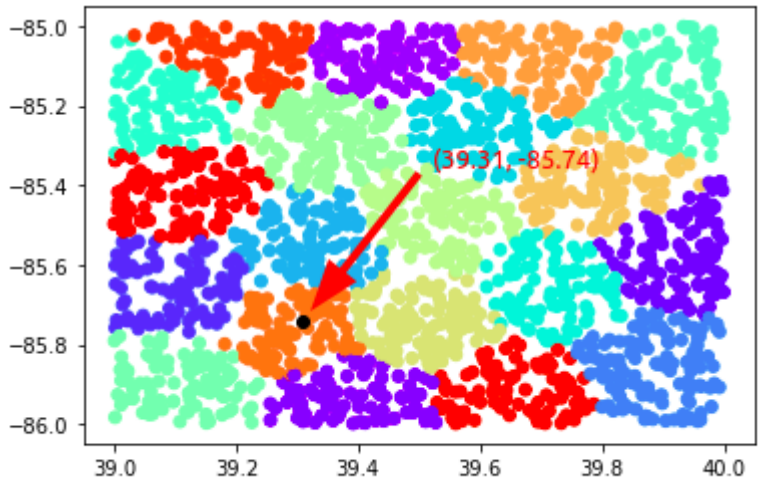
Visualizing the above plot, I can see that there are 20 optimal pickup zones for the FoodieX riders. The 20 clusters are the optimal number of zones that can there given the total number of restaurants. Each zone covers the optimal number of restaurants. Increasing the number of clusters will lead to overfitting and decreasing the number of clusters will lead to underfitting. Hence, the optimal number of pickup zones for the Delivery Service are 20.

I also predicted a location based on our model above. The model will predict the number of the cluster that the new location will belong to.

```
# predict a location
predicted = kmeans.predict([[39.31, -85.74]])
for label in predicted:
  print("The predicted cluster is: ", label)
```

```
The predicted cluster is:  16
```

I can also test the above prediction from the cluster plot was plotted above.



Conclusion 4

In this conclusion, I will be estimating the cook time based in the cuisine of a restaurant. To perform this, I will create a one-hot encoded vector for each cuisine in our dataset. This means that each dish in our dataset, will be a one-hot encoded vector. For example, suppose there are 89 unique dishes, in all the cuisines, in our dataset and restaurant A only serves 5 of those 89 dishes. Then our one-hot vector, for that restaurant, will have "1" in place of the dish present in the restaurant's cuisine and "0" for all other dishes.

Our model will predict cook time based on the cuisines of the restaurant. So, firstly I created a one-hot vector for each cuisine category. I created a new dataset with all the one-hot encoded vectors of all the restaurants, the restaurant name and the Cook Time of each restaurant.

The new dataset looks as following (the remaining columns have not been displayed due to space complexity):

	Restaurant	Cook_Time	Afghan	American	Andhra	Arabian	Asian	Assamese	Awadhi	BBQ	Bakery	Bangladeshi	Bar Food	Belgian	Bengali	Beverages
0	ID_6321	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	ID_2882	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	ID_1595	65	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	ID_5929	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	ID_6123	65	0	0	0	0	0	0	0	0	0	0	0	0	0	1

5 rows × 91 columns

In the next step, I will split our dataset into test and train sets. I assigned 80% of the data to train set and 20% to the train set.

```
X_train, X_test, y_train, y_test = train_test_split(X, Y,train_size=0.80)
```

I looked at the following link to train my model using different Classification Algorithms.

https://scikit-learn.org/stable/modules/multiclass.html

The RandomForestClassifier() gave the best accuracy amongst all. The model was trained, and then fitted on the test data and finally the accuracy was printed and analyzed.

Note: The small size of the dataset it probably what caused a lower accuracy score

```
# using the RandomForestClassifier because it gives the best accuracy as compared to other classifiers.
clf = RandomForestClassifier(criterion="entropy")
clf.fit(X_train, y_train)
print("The accuracy of predicting Cook Time using Cuisines is : ",clf.score(X_test,y_test)*100)
```

```
The accuracy of predicting Cook Time using Cuisines is :  64.10891089108911
```

The final accuracy shows that our model, given the small dataset, predicted the Cook Time of the restaurant, based on the cuisines, correctly 64% of the times. This accuracy can be considered usable since our dataset was small.