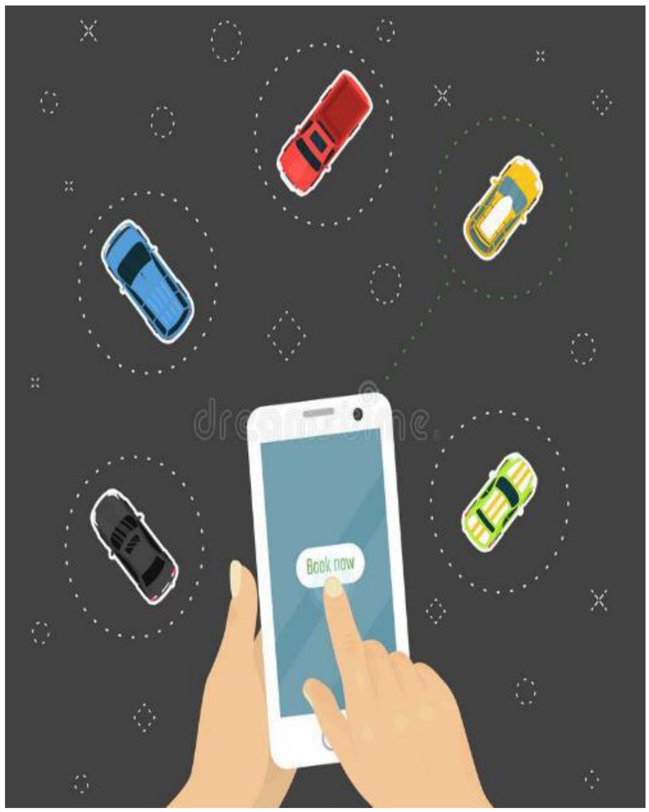
Market Segment Analysis of ‘Online Vehicle Booking Market’



Team members:

Parth Gupta
Kushagra Sharma
Arya BS
Raunak Shukla
Kovuri Mohan Kumar

Problem Statement

In this project we have to analyse the Online Vehicle market in India using Segmentation analysis and come up with a feasible strategy to enter the market, targeting the segments where there can be possible profit by offering Vehicle booking service.

In this report we analyze the current market scenarios, strategy implemented by market giants in India using segments such as distance travelled by the customers, time, destination, price etc.

1.Data Sources

The Dataset that we came across and found to be beneficial is:

https://www.kaggle.com/datasets/ravi72munde/uber-lyft-cab-prices

2.Data Pre-processing

Data preprocessing is the process of preparing the raw data and making it suitable for machine learning models. Data preprocessing includes data cleaning for making the data ready to be given to machine learning model. Before using the data for segmentation analysis we have to do data preprocessing. The following are various steps that we Data Preprocessing

- Import the .csv file and creation of Data Frame.
- Null Values removal from dataset.
- Data info and Summary
- Categorization of data based on data types, continuous or categorical.
- Exploratory Data Analysis and Data Visualization.

2.1 Importing the Dataset

First of all, let us have a look at the dataset we are going to use

```
df = pd.read_csv('/content/drive/MyDrive/data/Uber_segmentation/archive/rideshare_kaggle.csv')
df.head()
```

	id	timestamp	hour	day	month	datetime	timezone	source	destination	cab_type	...	precipIntensityMax	uvIndexTime	temperatureMin	tempera
0	424553bb-7174-41ea-aeb4-fe06d4f4b9d7	1.544953e+09	9	16	12	2018-12-16 09:30:07	America/New_York	Haymarket Square	North Station	Lyft	...	0.1276	1544979600	39.89	
1	4bd23055-6827-41c6-b23b-3c491f24e74d	1.543284e+09	2	27	11	2018-11-27 02:00:23	America/New_York	Haymarket Square	North Station	Lyft	...	0.1300	1543251600	40.49	
2	981a3613-77af-4620-a42a-0c0866077d1e	1.543367e+09	1	28	11	2018-11-28 01:00:22	America/New_York	Haymarket Square	North Station	Lyft	...	0.1064	1543338000	35.36	
3	c2d88af2-d278-4bfd-a8d0-29ca77cc5512	1.543554e+09	4	30	11	2018-11-30 04:53:02	America/New_York	Haymarket Square	North Station	Lyft	...	0.0000	1543507200	34.67	
4	e0126e1f-8ca9-4f2e-82b3-50505a09db9a	1.543463e+09	3	29	11	2018-11-29 03:49:20	America/New_York	Haymarket Square	North Station	Lyft	...	0.0001	1543420800	33.10	

5 rows × 57 columns

2.2 Handling of Missing Data

Data cleansing is an important step before you even begin the algorithmic trading process, which begins with historical data analysis for making the prediction model as accurate as possible. Missing Data can occur when no information is provided for one or more items or for a whole unit. In order to check missing values, we use a function *isnull()* function .

```
] df.isnull().sum()
```

After execution of this code we can find that there are not much missing values only 7 percent of the price values are missing. And to remove the rows where the price is not present we use *dropna()* function.

```
# remove the rows where the price is not present
df.dropna(axis = 0 , inplace = True)
df.isnull().sum()
```

2.3 Handling of Categorical Data

Categorical data is simply information aggregated into groups rather than being in numeric formats. They are present in almost all real-life datasets, yet the current algorithms still struggle to deal with them.the steps we use are,

```
[ ]  categorical = df.select_dtypes('object').columns.tolist()
     categorical

     ['destination', 'source', 'product_id', 'name']
```

```
[ ]  for cat in categorical:
         print('category : ' ,cat)
         print(df[cat].value_counts())
         print('\n')
```


```
[ ]  categorical = df.select_dtypes('object').columns.tolist()
     categorical

     ['destination', 'source', 'product_id', 'name']

[ ]  for cat in categorical:
         print('category : ' ,cat)
         print(df[cat].value_counts())
         print('\n')
```


```
category :  destination
Financial District         54192
Back Bay                   53190
Theatre District           53189
Haymarket Square           53171
Boston University          53171
Fenway                     53166
Northeastern University    53165
North End                  53164
South Station              53159
West End                   52992
Beacon Hill                52840
North Station              52577
Name: destination, dtype: int64
```

```
[ ]  category :  source
     Financial District         54197
     Back Bay                   53201
     Theatre District           53201
     Boston University          53172
     North End                  53171
     Fenway                     53166
     Northeastern University    53164
     South Station              53160
     Haymarket Square           53147
     West End                   52980
     Beacon Hill                52841
     North Station              52576
     Name: source, dtype: int64


     category :  product_id
     6f72dfc5-27f1-42e8-84db-ccc7a75f6969    55096
     9a0e7b09-b92b-4c41-9779-2ad22b4d779d    55096
     6d318bcc-22a3-4af6-bddd-b409bfce1546    55096
     6c84fd89-3f11-4782-9b50-97c468b19529    55095
     55c66225-fbe7-4fd5-9072-eab1ece5e23e    55094
     997acbb5-e102-41e1-b155-9df7de0a73f2    55091
     lyft_premier                            51235
     lyft                                    51235
     lyft_luxsuv                             51235
     lyft_plus                               51235
     lyft_lux                                51235
     lyft_line                               51233
     Name: product_id, dtype: int64
```

```
category :  name
UberXL          55096
WAV             55096
Black SUV       55096
Black           55095
UberX           55094
UberPool        55091
Lux             51235
Lyft            51235
Lux Black XL    51235
Lyft XL         51235
Lux Black       51235
Shared          51233
Name: name, dtype: int64
```

```
[ ] def one_hot_encode(df , column , prefix):
        dummy = pd.get_dummies(df[column] ,prefix = prefix)
        df = pd.concat([df , dummy] ,axis =1)
        df =df.drop(column , axis =1)

        return df
```

```
[ ] categorical
```

```
['destination', 'source', 'product_id', 'name']
```

```
[ ] df = one_hot_encode(df ,column =  'destination' , prefix = 'desti')
    df = one_hot_encode(df ,column =  'source' , prefix = 'src')
    df = one_hot_encode(df ,column =  'product_id' , prefix = 'pid')
    df = one_hot_encode(df ,column =  'name' , prefix = 'nm')

    df
```

day	month	price	distance	surge_multiplier	latitude	longitude	temperature	...	ls_ Rain in the morning and afternoon.	ls_ Rain throughout the day.	until morning, starting again in the evening.	ic_ clear-day	ic_ clear-night	ic_ cloudy	ic_ fog	ic_ partly-cloudy-day	p c
-0.180090	0.839574	-1.238169	-1.540640	-0.157905	-2.577771	1.632431	0.410021	...	-0.027065	3.431860	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
0.921885	-1.191081	-0.594693	-1.540640	-0.157905	-2.577771	1.632431	0.594394	...	-0.027065	-0.291387	3.670980	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	-0.
1.022065	-1.191081	-1.023677	-1.540640	-0.157905	-2.577771	1.632431	-0.186218	...	-0.027065	-0.291387	-0.272407	-0.200599	3.237150	-0.679275	-0.114754	-0.447233	-0.
1.222424	-1.191081	1.013998	-1.540640	-0.157905	-2.577771	1.632431	-0.773535	...	-0.027065	-0.291387	-0.272407	-0.200599	3.237150	-0.679275	-0.114754	-0.447233	-0.
1.122244	-1.191081	-0.809185	-1.540640	-0.157905	-2.577771	1.632431	-0.318550	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
-1.682784	0.839574	-0.755562	-1.047427	-0.157905	0.287088	0.090822	-0.376538	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
-1.682784	0.839574	-0.380201	-1.047427	-0.157905	0.287088	0.090822	-0.376538	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
-1.682784	0.839574	-0.755562	-1.047427	-0.157905	0.287088	0.090822	-0.376538	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
-1.682784	0.839574	1.121244	-1.047427	-0.157905	0.287088	0.090822	-0.376538	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.
-1.682784	0.839574	-0.701939	-1.047427	-0.157905	0.287088	0.090822	-0.376538	...	-0.027065	-0.291387	-0.272407	-0.200599	-0.308914	-0.679275	-0.114754	-0.447233	1.

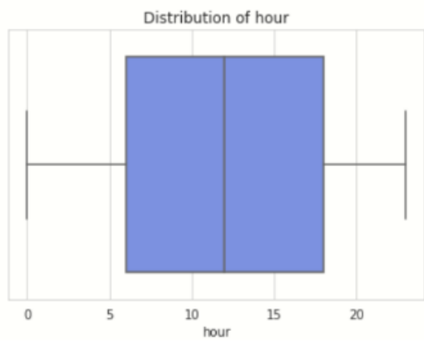
Observations

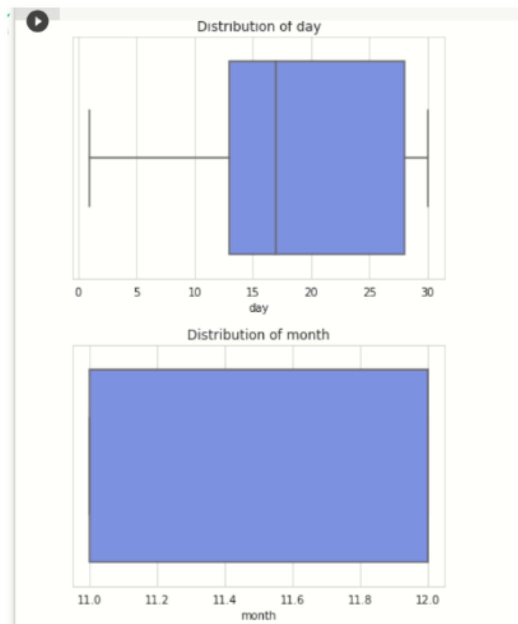
- Data is almost equally distributed for source and destination
- Weather was cloudy when most of the customers booked cab and least of them booked on foggy day.
- Most of the cabs were booked in the midnight mostly after 10 P.M.
- Month end found to be the busiest days
- People mostly booking the cabs which are budget friendly
- In summer People likely to book a cab

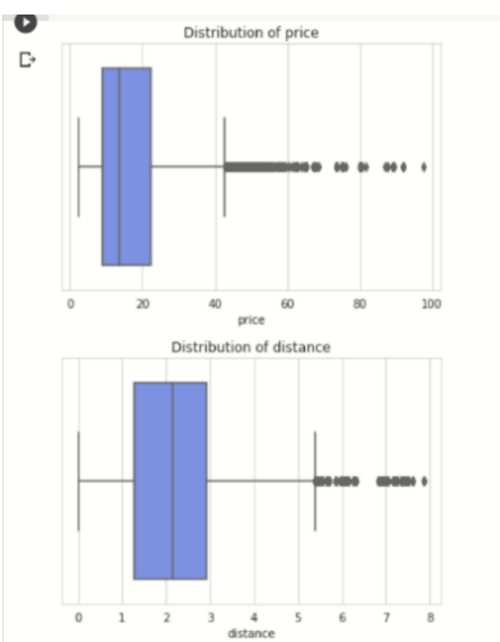
2.4 Data Visualization

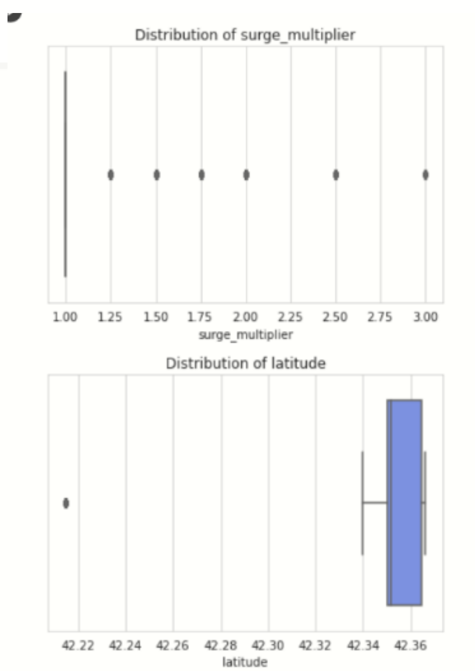
Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data.

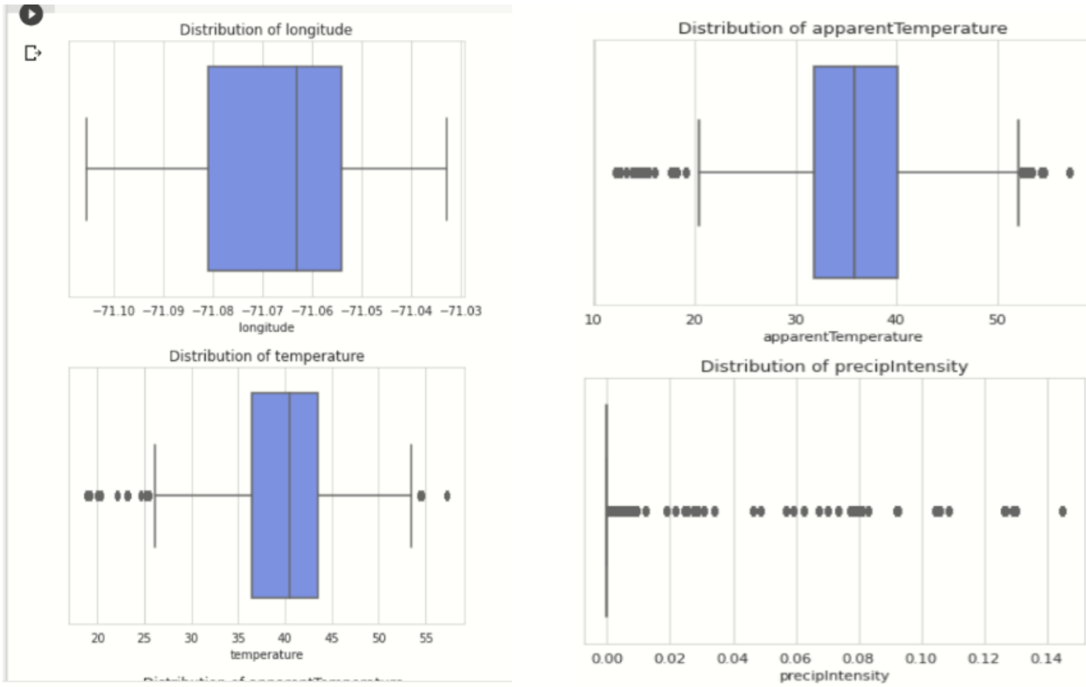
```
[ ]  sns.set_style('whitegrid')
     sns.set_palette('coolwarm')

     for i in df_1.columns:
       if df_1[i].dtype != 'O':
         sns.boxplot(x = df_1[i])
         plt.title('Distribution of '+i)
         plt.show()
```











Observations

- hour, day, month, latitude, longitude, precipintensity, humidity, windspeed, temperatureLow, pressure, ozone doesn't have outliers
- Others have outliers needs to check the data for treatment of the outliers

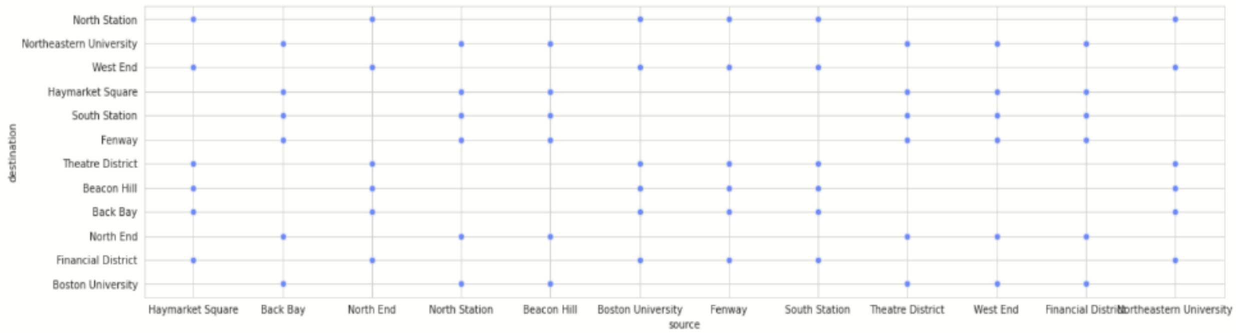
Demographic Analysis

Demographic Analysis is performed by using countplot and scatterplot

```
[ ] l = ['hour','day','month','source','destination']

    for i in l:
      plt.figure(figsize=(18,3))
      sns.countplot(x=df_1[i])
      plt.title('Distribution of '+i)
      plt.show()
```

```
[ ] plt.figure(figsize=(20,5))
    sns.scatterplot(x=df_1['source'],y = df_1['destination'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4f91260ed0>
```



Observations

- 10 A.M. to 6 P.M. and after 10 P.M. are the busiest hour when the customers have booked the cab.
- Month ends seems to be the busiest days for the cab drivers where as from 4 to 13 seems cabs are not much used
- Data is of only last 2 months of the year 2018
- Almost equally distributed among all the sources and destinations

Geomatric Analysis

```
[ ] from folium import plugins
    from folium.plugins import HeatMap
    # extracting longitude and latitude values to separate lists
    longs = df_1.longitude.to_list()
    lats = df_1.latitude.to_list()
    # calculating mean longitude and latitude values
    import statistics
    meanLong = statistics.mean(longs)
    meanLat = statistics.mean(lats)
    # create base map object using Map()
    mapObj = folium.Map(location=[meanLat, meanLong], tiles="openstreetmap", zoom_start = 10)
```

```
[ ] # create heatmap layer
    df_1.dropna(inplace = True)
    heatmap = HeatMap( list(zip(lats, longs, df_1["price"])),
                       min_opacity=0.2,
                       max_val=df_1["price"].max(),
                       radius=50, blur=50,
                       max_zoom=1)
    # add heatmap layer to base map
    heatmap.add_to(mapObj)
    mapObj
```

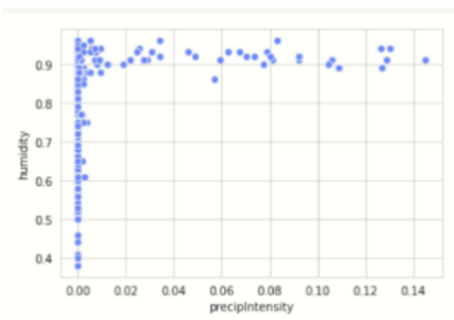
Psychographic Analysis

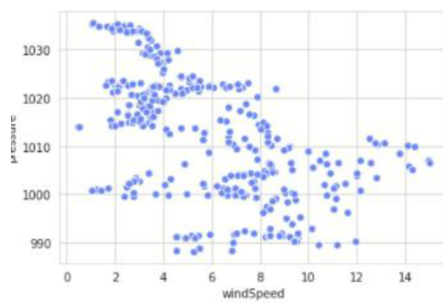
Psychographic Analysis is performed by using histograms and scatter plots

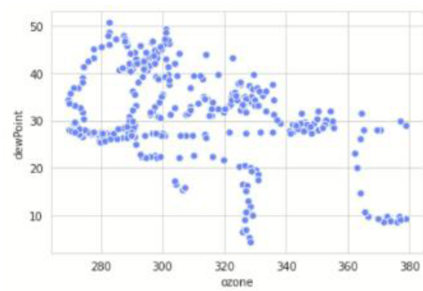
```
[ ]  a = 'temperature,apparentTemperature,precipIntensity,humidity,windSpeed,visibility,temperatureHigh,temperatureLow,icon,dewPoint,pressure,ozone'.split(',')
```

```
[ ]
     fig,([ax0,ax1],[ax2,ax3],[ax4,ax5],[ax6,ax7],[ax8,ax9],[ax10,ax11]) = plt.subplots(ncols=2,nrows=6,figsize=(30,60))

     ax = [ax0,ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9,ax10,ax11]
     for i in range(0,12):

       #sns.set(rc={'figure.figsize':(40,20)})
       sns.histplot(data=df_1,x=a[i],ax=ax[i])
       ax[i].set_title(a[i])
```

```
]  sns.scatterplot(x=df_1['precipIntensity'],y=df_1['humidity'])
   plt.show()
   sns.scatterplot(x=df_1['windSpeed'],y=df_1['pressure'])
   plt.show()
   sns.scatterplot(x=df_1['ozone'],y=df_1['dewPoint'])
   plt.show()
```







Observations

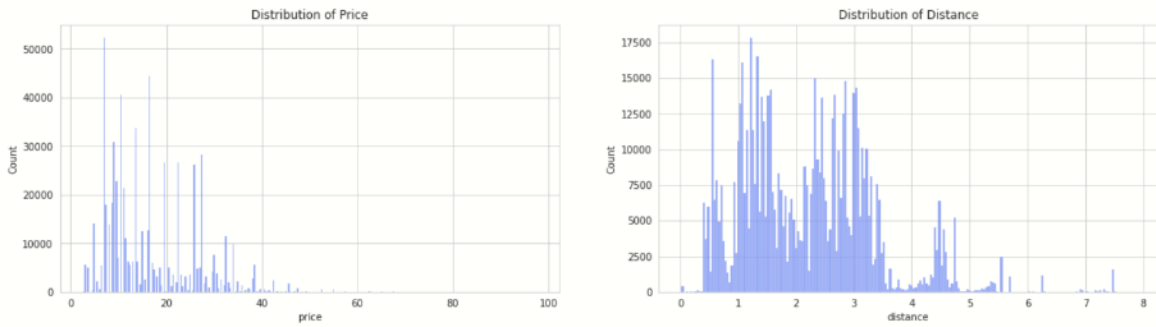
- Temperature is kindly normally distributed with most of the values ranging from 35 degrees to 45 degrees
- PrepIntensity is summed around 0.00 and visibility around 10
- Cloudy day was the most busiest day and surprisingly foggy day was the day when least cabs were booked.
- Precipintensity greater than 0.01 and humidity greater than 0.8 customer is more likely to ride a cab

Behaviour Analysis

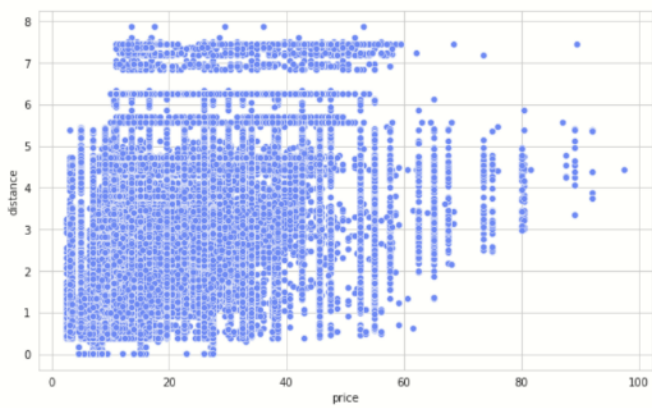
Behaviour analysis is performed by using histograms and scatter plots and

▾ Behaviour Analysis

```
[ ]  fig,([ax0,ax1]) = plt.subplots(ncols=2,figsize=(20,5))
     sns.histplot(x=df_1['price'],ax=ax0)
     ax0.set_title('Distribution of Price')
     sns.histplot(x=df_1['distance'],ax=ax1)
     ax1.set_title('Distribution of Distance')
     plt.show()
```




```
[ ]  plt.figure(figsize=(10,6))
     sns.scatterplot(x=df_1['price'],y = df_1['distance'])
     plt.show()
```

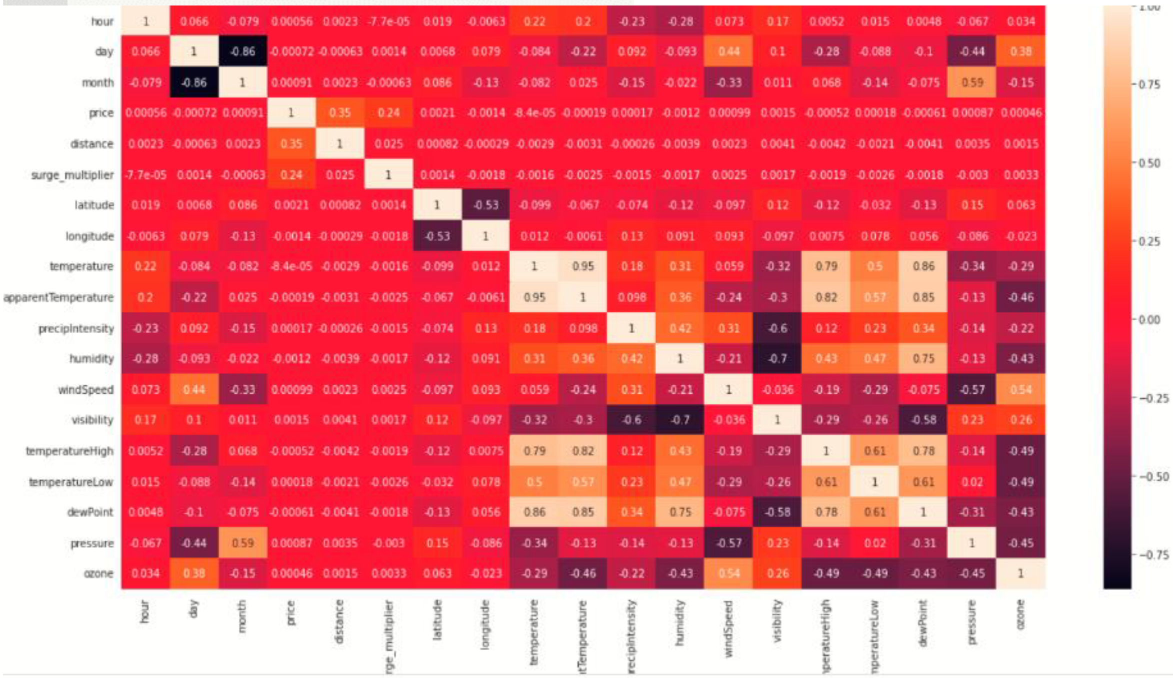


Observations

- ➢ Customers more likely to book a budget friendly cab ranging from 5 to 25
- ➢ Most of the customers books cab for shorter distance ranging from 0.5 to 3.5 units
- ➢ As the distance and price of the cab increases the bookings of the customers decreases

Checking the coorelation between the columns using heatmap

```python
plt.figure(figsize=(20,10))
sns.heatmap(df_1.corr(),annot=True)
```



Observations

- ➢ month and day are highly correlated, can remove month columns if needed
- ➢ Temperature and apparent temperature are both highly correlated about 95 percent
- ➢ Temperature is also corelated with dewpoint and temperaturehigh
- ➢ humidity is corelated with dewpoint and visibility

3.Segment Extraction

Data-driven market segmentation analysis is exploratory by nature. Consumer data sets are typically not well structured. Consumers come in all shapes and forms; a two-dimensional plot of consumers' product preferences typically does not contain clear groups of consumers. Rather, consumer preferences are spread across the entire plot. The combination of exploratory methods and unstructured consumer data means that results from any method used to extract market
segments from such data will strongly depend on the assumptions made on the structure of the segments implied by the method. The result of a market segmentation analysis, therefore, is determined as much by the underlying data as it is by the extraction algorithm chosen. Segmentation methods shape the segmentation solution. Many segmentation methods used to extract market segments are taken from the field of cluster analysis. In that case, market segments correspond to clusters

Clustering

Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean based distance or correlation-based distance. The decision on which similarity measure to use is application-specific. Clustering analysis can be done on the basis of features, where we try to find subgroups of samples based on features, or on the basis of samples, where we try to find subgroups of features based on samples.

```
[ ] pca = PCA(n_components=50)
    pca_df = pca.fit_transform(Scaled_df)
    pca_df = pd.DataFrame(pca_df)
```

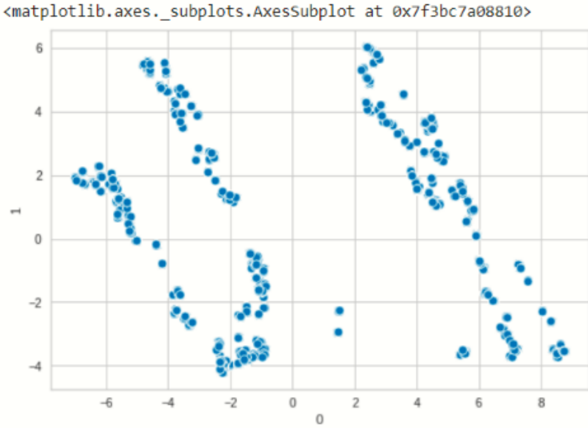
```
pca_df.head()
```

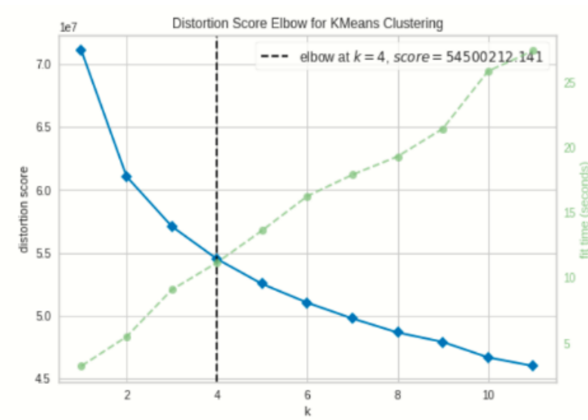
	0	1	2	3	4	5	6	7	8	9	...	40	41	42	43	44	45	46
0	4.227433	2.742441	0.660519	-0.731711	-2.557403	-1.604164	-1.360411	-0.171028	-0.506419	-0.114953	...	0.065333	-0.205501	-0.071416	-0.639130	-0.398105	1.328147	0.785115
1	-7.004690	1.931943	5.466211	2.179112	-0.440189	-1.109221	-0.687044	-1.553265	1.590145	-0.052833	...	-0.144100	-0.246081	-0.228664	-0.830984	-0.359621	1.482782	0.626831
2	-3.346527	-2.716751	0.006718	1.307865	-1.903953	-0.572774	2.780156	-2.446653	2.032484	0.072220	...	-0.122255	-0.098311	-0.032384	-0.420040	-0.396595	1.293240	0.773852
3	-1.392448	-3.623921	-0.939649	2.944187	2.454774	0.136782	2.445027	-2.035909	0.716339	0.071746	...	-0.250796	-0.078337	-0.103195	-0.475629	-0.355824	1.388227	0.764150
4	-2.255829	-4.194597	0.162183	-0.516435	-0.920025	-1.147870	-0.208793	0.097158	1.802554	-0.007875	...	-0.106365	-0.200190	-0.105474	-0.679171	-0.430762	1.390165	0.700502

5 rows × 50 columns

A practical issue encountered when using the K-means algorithm is the choice of the number of clusters, k. A common approach is to create an "Elbow Curve", which is a plot of the distortion (sum of squared distances from the centroid of a cluster) against chosen values of k. Let's create an Elbow Curve for each value of k (1,12).

<matplotlib.axes._subplots.AxesSubplot at 0x7f3bc7a08810>





<matplotlib.axes._subplots.AxesSubplot at 0x7f3bc7961a10>

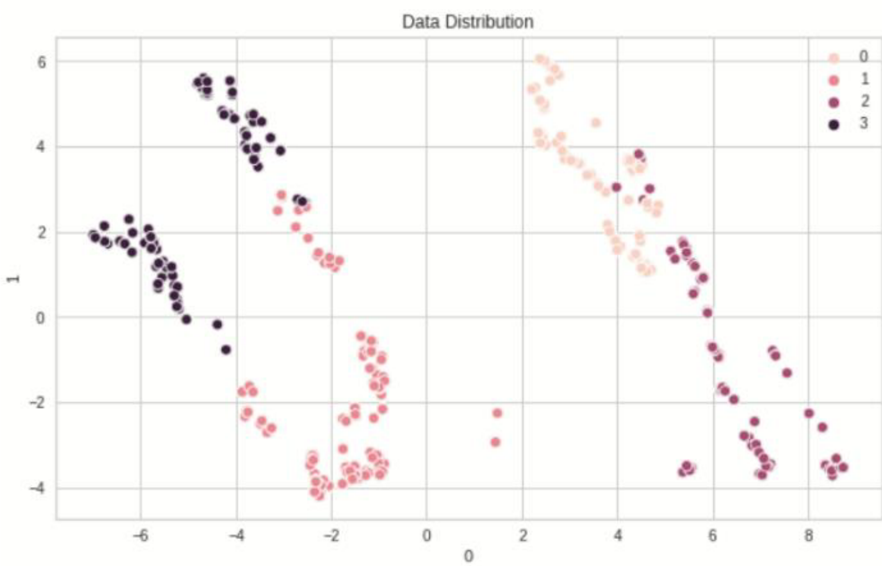
Based on this curve,we will choose k=4. Generate a new model with k=4.

```
[ ] # training the model with 4 clusters
    kmeans = KMeans(n_clusters=4)
    kmeans.fit(pca_df)
```

```
KMeans(n_clusters=4)
```

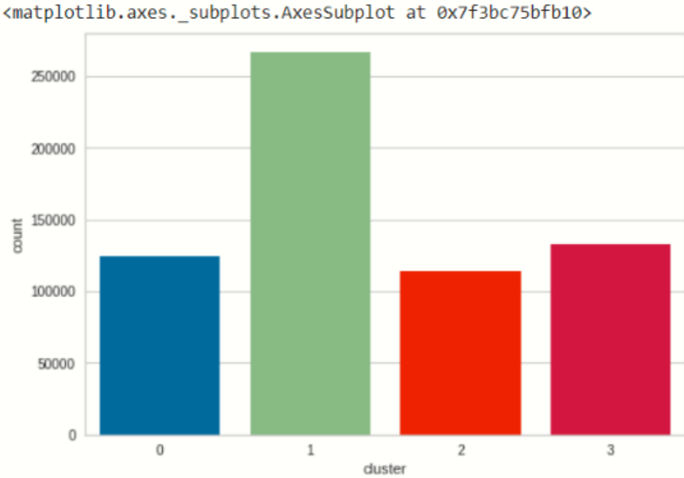
```
[ ] # predicting the clusters
    np.random.seed(42)
    preds = kmeans.predict(pca_df)
```

```
[ ] # plotting the clusters
    plt.figure(figsize=(10,6))
    sns.scatterplot(x=pca_df[0],y=pca_df[1],hue=preds)
    plt.title('Data Distribution')

    plt.show()
```




```
[ ] sns.countplot(x = df['cluster'])
    #plt.savefig('count11.png')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3bc75bfb10>
```



4. Profiling Segments

Once the market segments have been extracted, we need to understand what the four-segment k-means solution means. The first step in this direction is to create a segment profile plot. The segment profile plot makes it easy to see key characteristics of each market segment. It also highlights differences between segments. To ensure the plot is easy to interpret, similar attributes should be positioned close to one another. We achieve this by calculating a hierarchical cluster analysis. Hierarchical cluster analysis used on attributes (rather than consumers) identifies – attribute by attribute – the most similar ones. At the end of step 6, the officials can have a good understanding of the nature of the four market segments in view of the information that was used to create these segments. Apart from that, they know little about the segments.

Identifying Key Characteristics of Market Segments

The aim of the profiling step is to get to know the market segments resulting from the extraction step. Profiling is only required when data-driven market segmentation is used. For commonsense segmentation, the profiles of the segments are predefined. If, for example, age is used as the segmentation variable for the commonsense segmentation, it is obvious that the resulting segments will be age groups. Therefore, Step 6 is not necessary when commonsense segmentation is conducted.

Traditional Approaches to Profiling Market Segments

Data-driven segmentation solutions are usually presented to users (clients, managers) in one of two ways: (1) as high level summaries simplifying segment characteristics to a point where they are misleadingly trivial, or (2) as large tables that provide, for each segment, exact percentages for each segmentation variable. Such tables are hard to interpret, and it is virtually impossible to get a quick overview of the key insights.

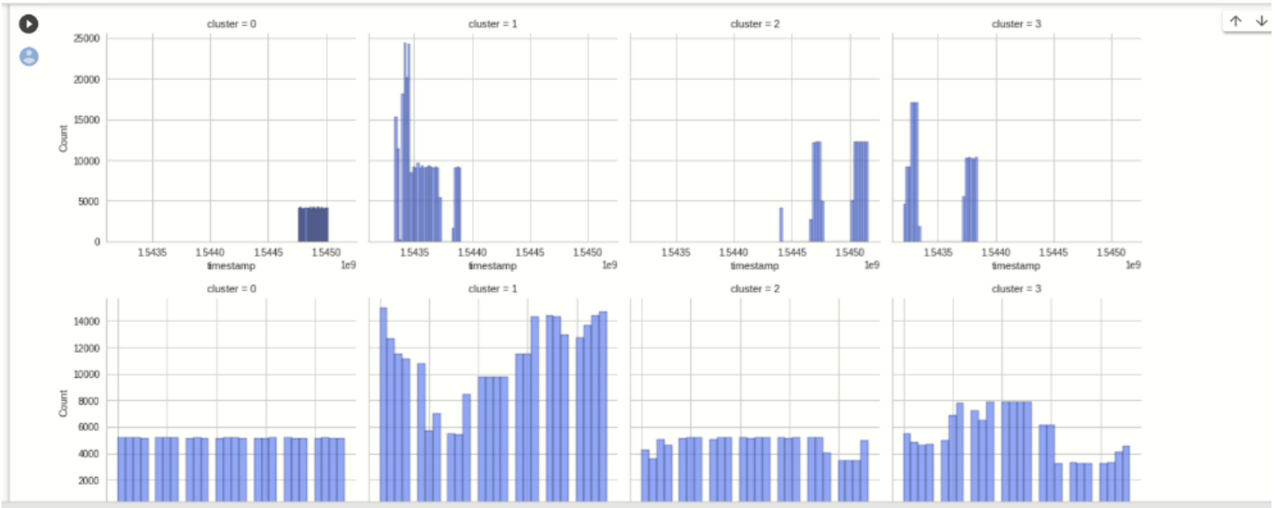
Segment Profiling with Visualisations

Visualisations are useful in the data-driven market segmentation process to inspect, for each segmentation solution, one or more segments in detail. Statistical graphs facilitate the interpretation of segment profiles. They also make it easier to assess the usefulness of a market segmentation solution. The process of segmenting data always leads to a large number of alternative solutions. Selecting one of the possible solutions is a critical decision. Visualisations of solutions assist the data analyst and user with this task.

• Identifying Defining Characteristics of Market Segments : A good way to understand the defining characteristics of each segment is to produce a segment profile plot. The segment profile plot shows – for all segmentation variables – how each market segment differs from the overall sample. The segment profile plot is the direct visual translation of tables.

• Assessing Segment Separation: Segment separation plots are very simple if the number of segmentation variables is low, but become complex as the number of segmentation variables increases. But even in such complex situations, segment separation plots offer data analysts and users a quick overview of the data situation, and the segmentation solution.

```
sns.set_palette('coolwarm')
for i in df.drop(l,axis=1):
    grid = sns.FacetGrid(df,height=4,col='cluster')
    grid = grid.map(sns.histplot,i,bins=30)
plt.show()
```



Performing DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a base algorithm for density-based clustering. It can discover clusters of different shapes and sizes from a large amount of data, which is containing noise and outliers.

DBSCAN{Removing the Noise}

```
[72] # Using the elbow method to find the optimal number of clusters
     from sklearn.cluster import DBSCAN
     dbscan=DBSCAN(eps=3,min_samples=4)
```

```
[74] # Fitting the model

     model=dbscan.fit(Scaled_df)

     labels=model.labels_
```

```
[75] from sklearn import metrics

     #identifying the points which makes up our core points
     sample_cores=np.zeros_like(labels,dtype=bool)

     sample_cores[dbscan.core_sample_indices_]=True

     #Calculating the number of clusters

     n_clusters=len(set(labels))- (1 if -1 in labels else 0)


     print(metrics.silhouette_score(Scaled_df,labels))

     -0.05342759044626029
```

Customizing the Market Mix

The *marketing mix* refers to the set of actions, or tactics, that a company uses to promote its brand or product in the market. The 4Ps make up a typical marketing mix - Price, Product, Promotion and Place**.**

- **Price***:* refers to the value that is put for a product. It depends on costs of production, segment targeted, ability of the market to pay, supply - demand and a host of other direct and indirect factors. There can be several types of pricing strategies, each tied in with an overall business plan

- **Product***:* refers to the item actually being sold. The product must deliver a minimum level of performance; otherwise even the best work on the other elements of the marketing mix won't do any good.

- **Place***:* refers to the point of sale. In every industry, catching the eye of the consumer and making it easy for her to buy it is the main aim of a good distribution or 'place' strategy. Retailers pay a premium for the right location. In fact, the mantra of a successful retail business is 'location, location, location'.

- **Promotion***:* this refers to all the activities undertaken to make the product or service known to the user and trade. This can include advertising, word of mouth, press reports, incentives, commissions and awards to the trade. It can also include consumer schemes, direct marketing, contests and prizes.

Potential Sales in Early Market

For specifically market like India cheap rates can be very good factor for the cab online booking segments .In general studies it is seen 70% people prefer less price over comfort and other factors.
So we can easily occupy 15-30% of market of Ola and Uber in initial state especially in multi-cities as they are very important market with respect to online cab booking system.

Link to GitHub profile with codes and datasets well documented-

Name	GitHub Link
Parth Gupta	Link
Kushagra Sharma	Link
Arya BS	Link
Raunak Shukla	Link
Kovuri Mohan Kumar	Link