Online Vehicle Booking Market Segmentation Analysis

LADI JEEVAN SAI

ML INTERNSHIP

TASK 2

Dataset Link:-

https://drive.google.com/file/d/1_hwV8vWkVMFwB1XiiEoigKpCGqeVJ15g/view?usp=share_link

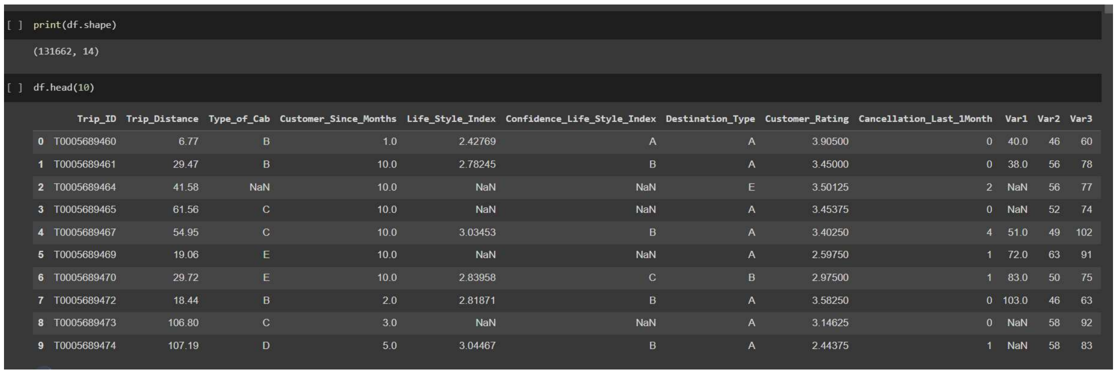
Project Link:-

GitHub - ladijeevansai/Online-vehicle-booking-market-segmentation

<u>Dataset description</u>

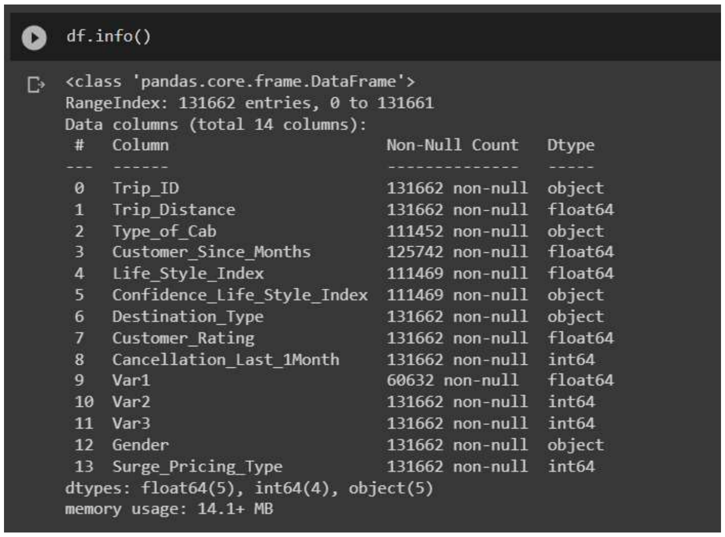
I had collected the data from kaggle ,which consists of 14 attributes described below and consists of 131662 rows.
The attributes are described as follows:

- Trip_ID: A unique identifier for each trip.
- Trip_Distance: The distance in units unspecified that the customer has requested for the trip.
- Type_of_Cab: A categorical variable indicating the category of the cab requested by the customer.
- Customer_Since_Months: A numerical variable indicating the number of months the customer has been using cab services; 0 indicates the current month.
- Life_Style_Index: A proprietary index created by Sigma Cabs that indicates the lifestyle of the customer based on their behavior.
- Confidence_Life_Style_Index: A categorical variable indicating the level of confidence in the Life_Style_Index mentioned above.
- Destination_Type: Sigma Cabs divides any destination into one of 14 categories.
- Customer_Rating: The average lifetime rating of the customer until the present time.
- Cancellation_Last_1Month: The number of trips canceled by the customer in the last month.
- Var1, Var2, and Var3: Continuous variables masked by the company that can be used for modeling purposes.
- Gender: A categorical variable indicating the gender of the customer.
- Surge_Pricing_Type: The target variable indicating the surge pricing type; it can take on 3 different values.



```
[ ] print(df.shape)

(131662, 14)

[ ] df.head(10)
```

	Trip_ID	Trip_Distance	Type_of_Cab	Customer_Since_Months	Life_Style_Index	Confidence_Life_Style_Index	Destination_Type	Customer_Rating	Cancellation_Last_1Month	Var1	Var2	Var3
0	T0005689460	6.77	B	1.0	2.42769	A	A	3.90500	0	40.0	46	60
1	T0005689461	29.47	B	10.0	2.78245	B	A	3.45000	0	38.0	56	78
2	T0005689464	41.58	NaN	10.0	NaN	NaN	E	3.50125	2	NaN	56	77
3	T0005689465	61.56	C	10.0	NaN	NaN	A	3.45375	0	NaN	52	74
4	T0005689467	54.95	C	10.0	3.03453	B	A	3.40250	4	51.0	49	102
5	T0005689469	19.06	E	10.0	NaN	NaN	A	2.59750	1	72.0	63	91
6	T0005689470	29.72	E	10.0	2.83958	C	B	2.97500	1	83.0	50	75
7	T0005689472	18.44	B	2.0	2.81871	B	A	3.58250	0	103.0	46	63
8	T0005689473	106.80	C	3.0	NaN	NaN	A	3.14625	0	NaN	58	92
9	T0005689474	107.19	D	5.0	3.04467	B	A	2.44375	1	NaN	58	83

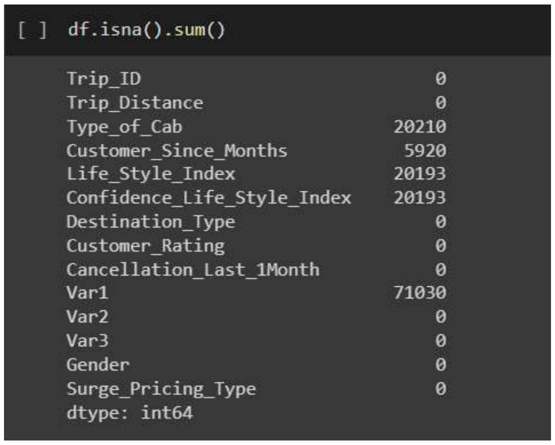


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 131662 entries, 0 to 131661
Data columns (total 14 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Trip_ID                      131662 non-null  object
 1   Trip_Distance                131662 non-null  float64
 2   Type_of_Cab                  111452 non-null  object
 3   Customer_Since_Months        125742 non-null  float64
 4   Life_Style_Index             111469 non-null  float64
 5   Confidence_Life_Style_Index  111469 non-null  object
 6   Destination_Type             131662 non-null  object
 7   Customer_Rating              131662 non-null  float64
 8   Cancellation_Last_1Month     131662 non-null  int64
 9   Var1                         60632 non-null   float64
 10  Var2                         131662 non-null  int64
 11  Var3                         131662 non-null  int64
 12  Gender                       131662 non-null  object
 13  Surge_Pricing_Type           131662 non-null  int64
dtypes: float64(5), int64(4), object(5)
memory usage: 14.1+ MB
```

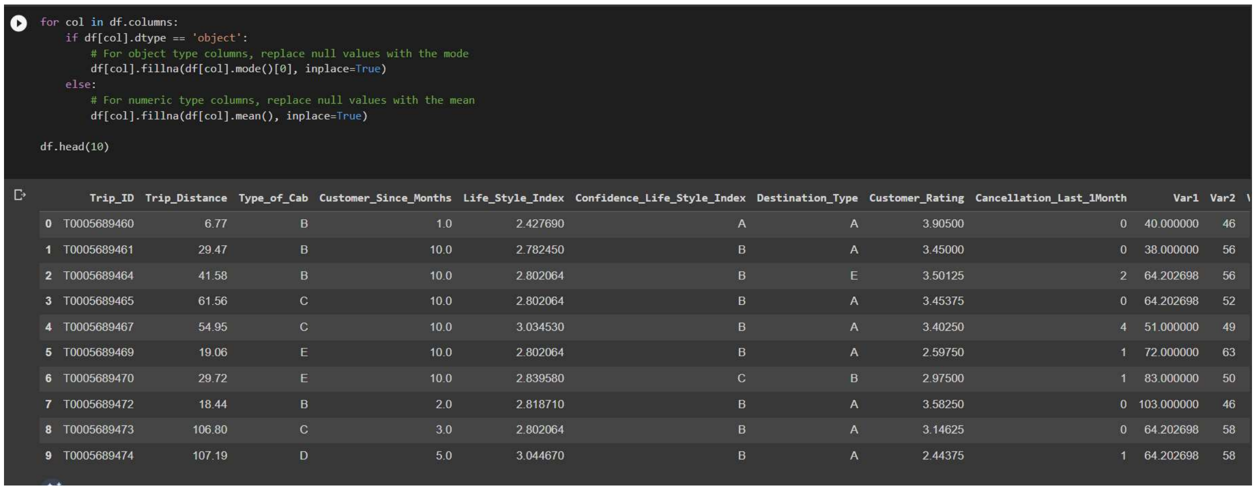
Data Preprocessing

In this step , initially i had replaced all the null values with mean if it is float aor integer data type and replaced by mode if it is categorical attribute



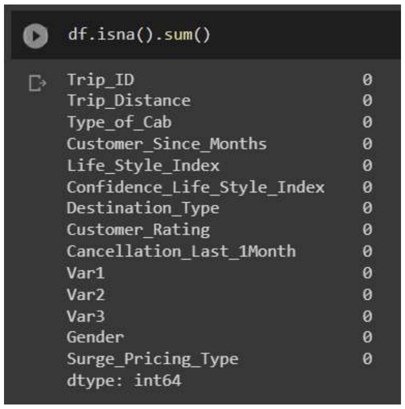
```
df.isna().sum()
```

```
Trip_ID                            0
Trip_Distance                      0
Type_of_Cab                    20210
Customer_Since_Months           5920
Life_Style_Index               20193
Confidence_Life_Style_Index    20193
Destination_Type                   0
Customer_Rating                    0
Cancellation_Last_1Month           0
Var1                           71030
Var2                               0
Var3                               0
Gender                             0
Surge_Pricing_Type                 0
dtype: int64
```



```python
for col in df.columns:
    if df[col].dtype == 'object':
        # For object type columns, replace null values with the mode
        df[col].fillna(df[col].mode()[0], inplace=True)
    else:
        # For numeric type columns, replace null values with the mean
        df[col].fillna(df[col].mean(), inplace=True)

df.head(10)
```

```
       Trip_ID  Trip_Distance  Type_of_Cab  Customer_Since_Months  Life_Style_Index  Confidence_Life_Style_Index  Destination_Type  Customer_Rating  Cancellation_Last_1Month        Var1  Var2  \
0  T0005689460           6.77            B                    1.0          2.427690                            A                 A          3.90500                         0   40.000000    46
1  T0005689461          29.47            B                   10.0          2.782450                            B                 A          3.45000                         0   38.000000    56
2  T0005689464          41.58            B                   10.0          2.802064                            B                 E          3.50125                         2   64.202698    56
3  T0005689465          61.56            C                   10.0          2.802064                            B                 A          3.45375                         0   64.202698    52
4  T0005689467          54.95            C                   10.0          3.034530                            B                 A          3.40250                         4   51.000000    49
5  T0005689469          19.06            E                   10.0          2.802064                            B                 A          2.59750                         1   72.000000    63
6  T0005689470          29.72            E                   10.0          2.839580                            C                 B          2.97500                         1   83.000000    50
7  T0005689472          18.44            B                    2.0          2.818710                            B                 A          3.58250                         0  103.000000    46
8  T0005689473         106.80            C                    3.0          2.802064                            B                 A          3.14625                         0   64.202698    58
9  T0005689474         107.19            D                    5.0          3.044670                            B                 A          2.44375                         1   64.202698    58
```

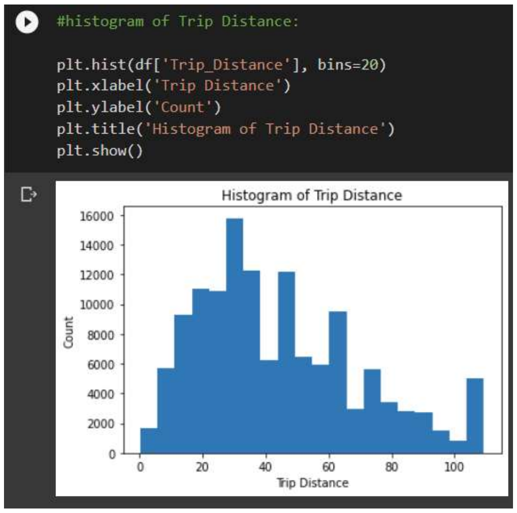


```python
df.isna().sum()
```

```
Trip_ID                        0
Trip_Distance                  0
Type_of_Cab                    0
Customer_Since_Months          0
Life_Style_Index               0
Confidence_Life_Style_Index    0
Destination_Type               0
Customer_Rating                0
Cancellation_Last_1Month       0
Var1                           0
Var2                           0
Var3                           0
Gender                         0
Surge_Pricing_Type             0
dtype: int64
```

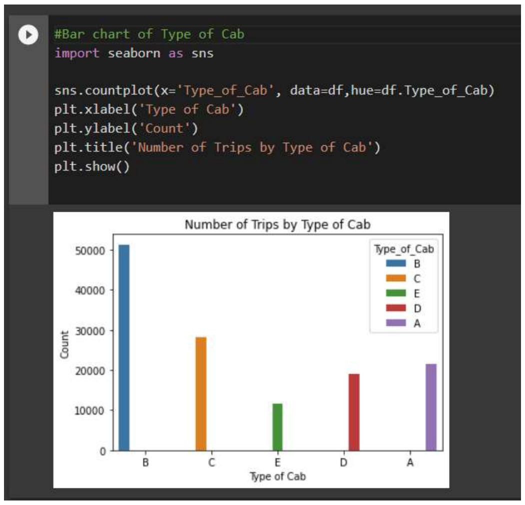
Data visualization:-

Using matplot and seaborn libraries i had tried to visualize several attributes in the dataset



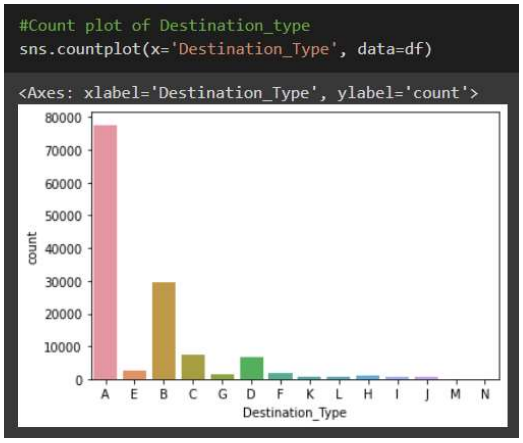
```python
#histogram of Trip Distance:

plt.hist(df['Trip_Distance'], bins=20)
plt.xlabel('Trip Distance')
plt.ylabel('Count')
plt.title('Histogram of Trip Distance')
plt.show()
```

This plot shows about the frequency distribution of the Trip distance attribute.



```
#Bar chart of Type of Cab
import seaborn as sns

sns.countplot(x='Type_of_Cab', data=df,hue=df.Type_of_Cab)
plt.xlabel('Type of Cab')
plt.ylabel('Count')
plt.title('Number of Trips by Type of Cab')
plt.show()
```

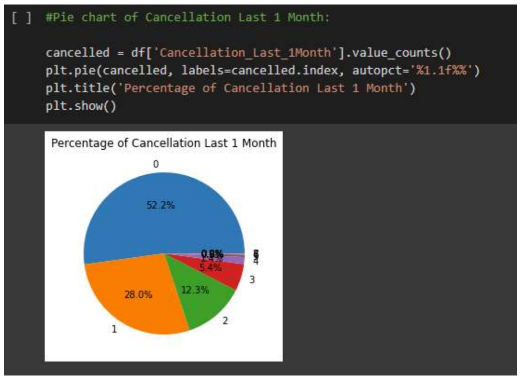
This plot tells that the sigma cabs have ‘B’ type cabs the most and ‘E’ type the least.



```
#Count plot of Destination_type
sns.countplot(x='Destination_Type', data=df)
```

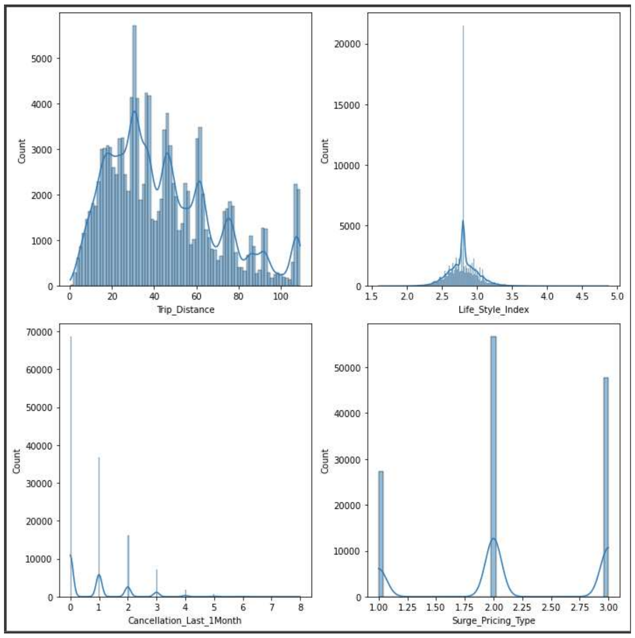
```
<Axes: xlabel='Destination_Type', ylabel='count'>
```

The plot shows that most people prefer Destination type A, while types F, G, K, L, H, I, J, M, and N are very unpopular. This suggests Sigma cabs can supply more cabs at Destinations A and B, while having very few or no cabs at Destinations F, G, K, L, H, I, J, M,N.

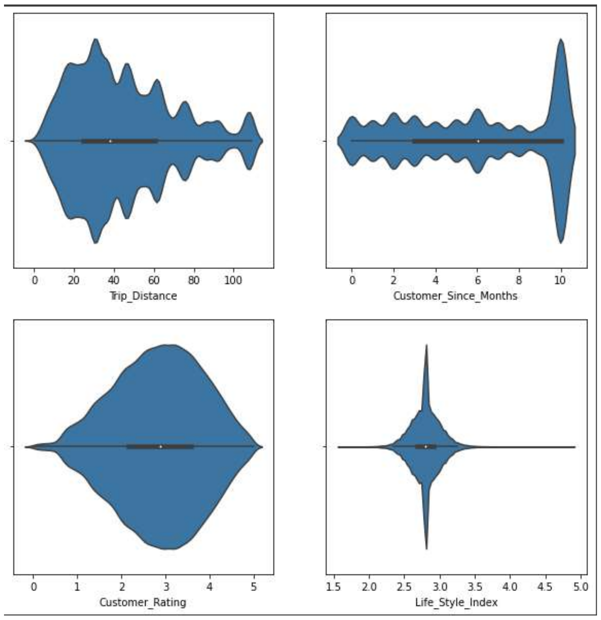


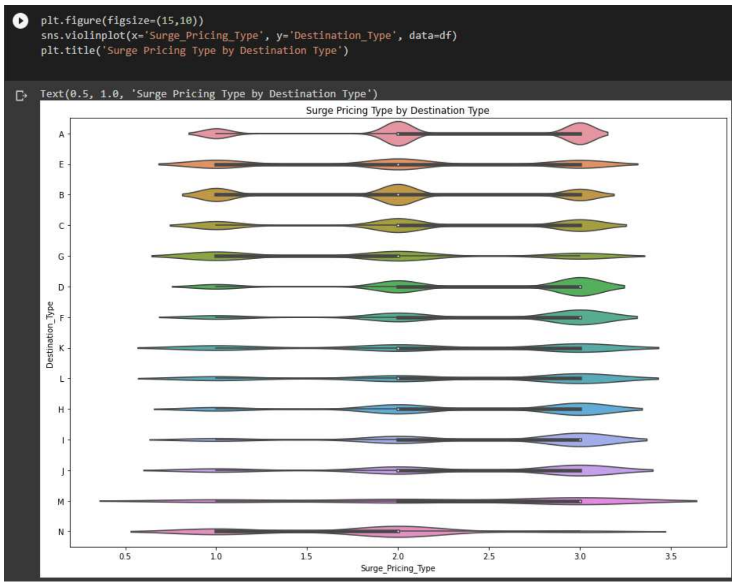
```
[ ]  #Pie chart of Cancellation Last 1 Month:

     cancelled = df['Cancellation_Last_1Month'].value_counts()
     plt.pie(cancelled, labels=cancelled.index, autopct='%1.1f%%')
     plt.title('Percentage of Cancellation Last 1 Month')
     plt.show()
```

This shows us the percentage distribution of the ride cancellations in last month



Based on the plots, it can be inferred that the majority of people who opted for cabs have chosen it for a good distance, with the distance ranging between 20-60. Additionally, the lifestyle index of most people falls between 2.5-3.25, with an average of 2.75. Furthermore, the third plot shows that the cancellation rate is not very high and ranges mostly between 1-3, with around 10,000 people never canceling their ride.



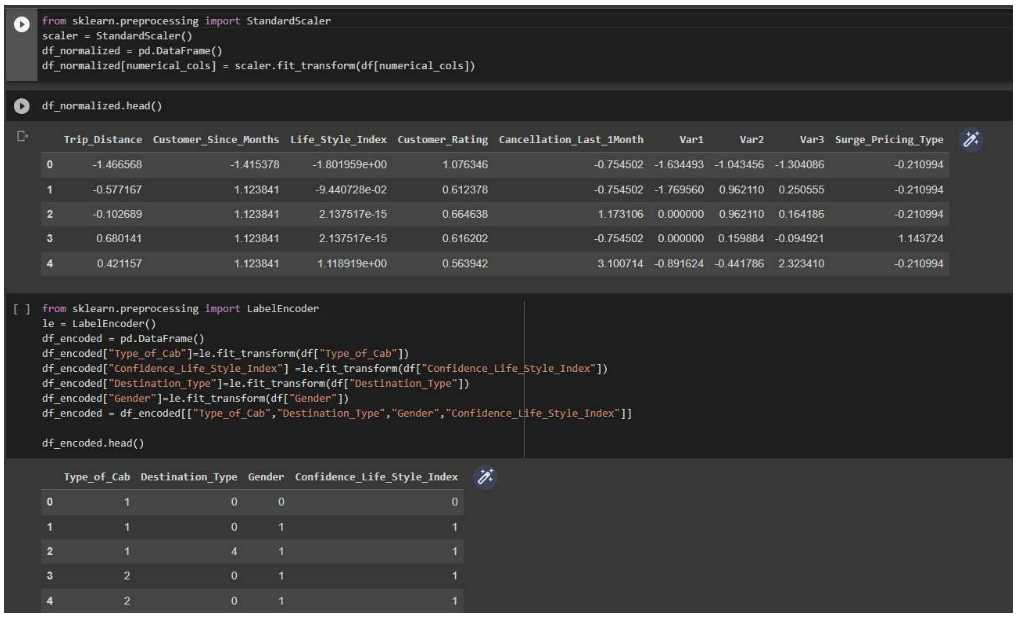


```
plt.figure(figsize=(15,10))
sns.violinplot(x='Surge_Pricing_Type', y='Destination_Type', data=df)
plt.title('Surge Pricing Type by Destination Type')
```

```
Text(0.5, 1.0, 'Surge Pricing Type by Destination Type')
```

The above visualizations are in the form of violin plots, which display the complete distribution of several variables. Specifically, they show the distribution of trip distance, customer service months, customer rating, and lifestyle index. Additionally, there is a violin plot depicting the relationship between surge pricing and destination type, which allows us to identify locations with higher surge rates and provides valuable insights into various aspects of the dataset, including the range and distribution of key variables, as well as patterns and trends in the data.
By analyzing these plots, we can gain a better understanding about the relation between the dataset and make more informed decisions based on this information.

Normalizing and one hot encoding the data



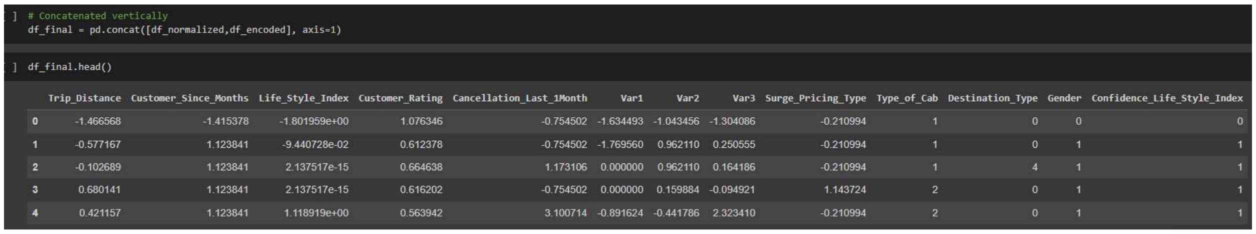
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_normalized = pd.DataFrame()
df_normalized[numerical_cols] = scaler.fit_transform(df[numerical_cols])
```

```
df_normalized.head()
```

	Trip_Distance	Customer_Since_Months	Life_Style_Index	Customer_Rating	Cancellation_Last_1Month	Var1	Var2	Var3	Surge_Pricing_Type
0	-1.466568	-1.415378	-1.801959e+00	1.076346	-0.754502	-1.634493	-1.043456	-1.304086	-0.210994
1	-0.577167	1.123841	-9.440728e-02	0.612378	-0.754502	-1.769560	0.962110	0.250555	-0.210994
2	-0.102689	1.123841	2.137517e-15	0.664638	1.173106	0.000000	0.962110	0.164186	-0.210994
3	0.680141	1.123841	2.137517e-15	0.616202	-0.754502	0.000000	0.159884	-0.094921	1.143724
4	0.421157	1.123841	1.118919e+00	0.563942	3.100714	-0.891624	-0.441786	2.323410	-0.210994

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df_encoded = pd.DataFrame()
df_encoded["Type_of_Cab"]=le.fit_transform(df["Type_of_Cab"])
df_encoded["Confidence_Life_Style_Index"] =le.fit_transform(df["Confidence_Life_Style_Index"])
df_encoded["Destination_Type"]=le.fit_transform(df["Destination_Type"])
df_encoded["Gender"]=le.fit_transform(df["Gender"])
df_encoded = df_encoded[["Type_of_Cab","Destination_Type","Gender","Confidence_Life_Style_Index"]]

df_encoded.head()
```

	Type_of_Cab	Destination_Type	Gender	Confidence_Life_Style_Index
0	1	0	0	0
1	1	0	1	1
2	1	4	1	1
3	2	0	1	1
4	2	0	1	1



```
# Concatenated vertically
df_final = pd.concat([df_normalized,df_encoded], axis=1)
```

```
df_final.head()
```

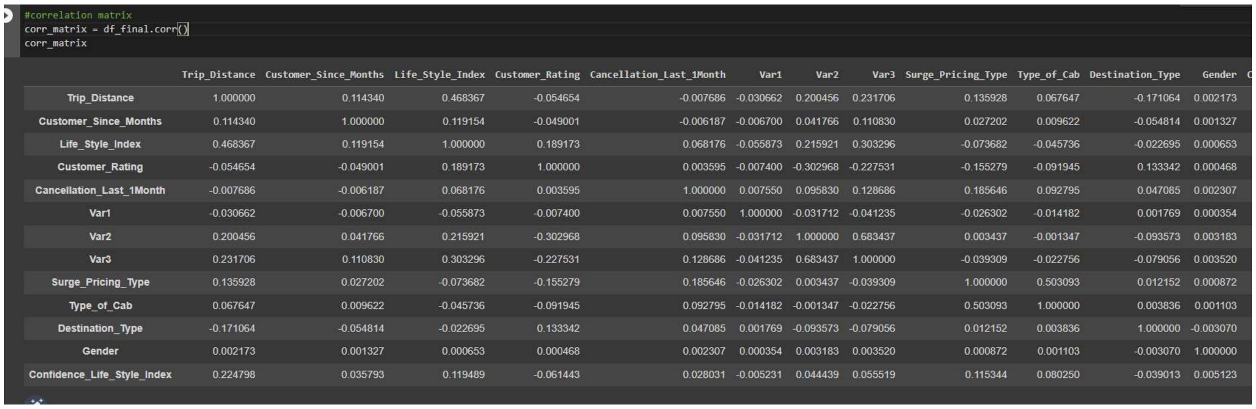
	Trip_Distance	Customer_Since_Months	Life_Style_Index	Customer_Rating	Cancellation_Last_1Month	Var1	Var2	Var3	Surge_Pricing_Type	Type_of_Cab	Destination_Type	Gender	Confidence_Life_Style_Index
0	-1.466568	-1.415378	-1.801959e+00	1.076346	-0.754502	-1.634493	-1.043456	-1.304086	-0.210994	1	0	0	0
1	-0.577167	1.123841	-9.440728e-02	0.612378	-0.754502	-1.769560	0.962110	0.250555	-0.210994	1	0	1	1
2	-0.102689	1.123841	2.137517e-15	0.664638	1.173106	0.000000	0.962110	0.164186	-0.210994	1	4	1	1
3	0.680141	1.123841	2.137517e-15	0.616202	-0.754502	0.000000	0.159884	-0.094921	1.143724	2	0	1	1
4	0.421157	1.123841	1.118919e+00	0.563942	3.100714	-0.891624	-0.441786	2.323410	-0.210994	2	0	1	1

In the above code i had followed few steps to preprocess the data:

1. Separated the numerical and categorical attributes into separate arrays.
2. Normalized the numerical attributes (integer and float) using StandardScaler, which is a common method of normalization that scales the data to have a mean of 0 and a standard deviation of 1.
3. Encoded your categorical attributes into numerical values, which is a common method of transforming non-numeric data into a format that can be used by machine learning algorithms.
4. Merged the two dataframes back together to create a single, uniform, and normalized dataset.

This preprocessing pipeline can help improve the performance of your machine learning models by reducing the impact of differences in scale and format between different types of attributes. Normalizing numerical attributes can make them more comparable, while encoding categorical attributes can make them more amenable to machine learning algorithms.

Removing Outliers and dropping unnecessary attributes:-



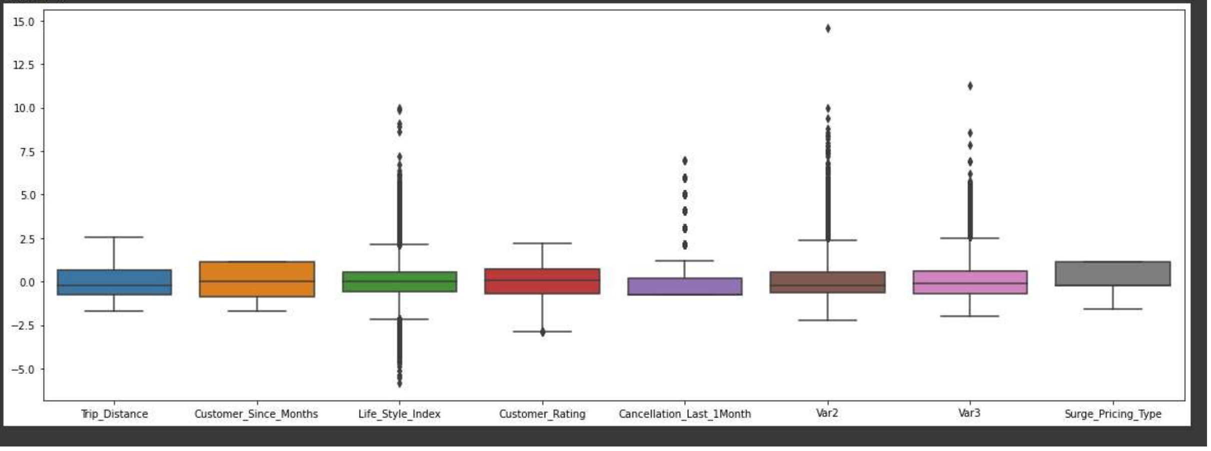
```
#correlation matrix
corr_matrix = df_final.corr()
corr_matrix
```

	Trip_Distance	Customer_Since_Months	Life_Style_Index	Customer_Rating	Cancellation_Last_1Month	Var1	Var2	Var3	Surge_Pricing_Type	Type_of_Cab	Destination_Type	Gender
Trip_Distance	1.000000	0.114340	0.468367	-0.054654	-0.007686	-0.030662	0.200456	0.231706	0.135928	0.067647	-0.171064	0.002173
Customer_Since_Months	0.114340	1.000000	0.119154	-0.049001	-0.006187	-0.006700	0.041766	0.110830	0.027202	0.009622	-0.054814	0.001327
Life_Style_Index	0.468367	0.119154	1.000000	0.189173	0.068176	-0.055873	0.215921	0.303296	-0.073682	-0.045736	-0.022695	0.000653
Customer_Rating	-0.054654	-0.049001	0.189173	1.000000	0.003595	-0.007400	-0.302968	-0.227531	-0.155279	-0.091945	0.133342	0.000468
Cancellation_Last_1Month	-0.007686	-0.006187	0.068176	0.003595	1.000000	0.007550	0.095830	0.128686	0.185646	0.092795	0.047085	0.002307
Var1	-0.030662	-0.006700	-0.055873	-0.007400	0.007550	1.000000	-0.031712	-0.041235	-0.026302	-0.014182	0.001769	0.000354
Var2	0.200456	0.041766	0.215921	-0.302968	0.095830	-0.031712	1.000000	0.683437	0.003437	-0.001347	-0.093573	0.003183
Var3	0.231706	0.110830	0.303296	-0.227531	0.128686	-0.041235	0.683437	1.000000	-0.039309	-0.022756	-0.079056	0.003520
Surge_Pricing_Type	0.135928	0.027202	-0.073682	-0.155279	0.185646	-0.026302	0.003437	-0.039309	1.000000	0.503093	0.012152	0.000872
Type_of_Cab	0.067647	0.009622	-0.045736	-0.091945	0.092795	-0.014182	-0.001347	-0.022756	0.503093	1.000000	0.003836	0.001103
Destination_Type	-0.171064	-0.054814	-0.022695	0.133342	0.047085	0.001769	-0.093573	-0.079056	0.012152	0.003836	1.000000	-0.003070
Gender	0.002173	0.001327	0.000653	0.000468	0.002307	0.000354	0.003183	0.003520	0.000872	0.001103	-0.003070	1.000000
Confidence_Life_Style_Index	0.224798	0.035793	0.119489	-0.061443	0.028031	-0.005231	0.044439	0.055519	0.115344	0.080250	-0.039013	0.005123

```
df_numeric = df_final[numerical_cols]
df_numeric = df_numeric.drop('Var1', axis=1)
```

```
# Draw boxplots for all numerical columns
fig = plt.figure(figsize=(20,7))
sns.boxplot(data=df_numeric)
```

```
<Axes: >
```



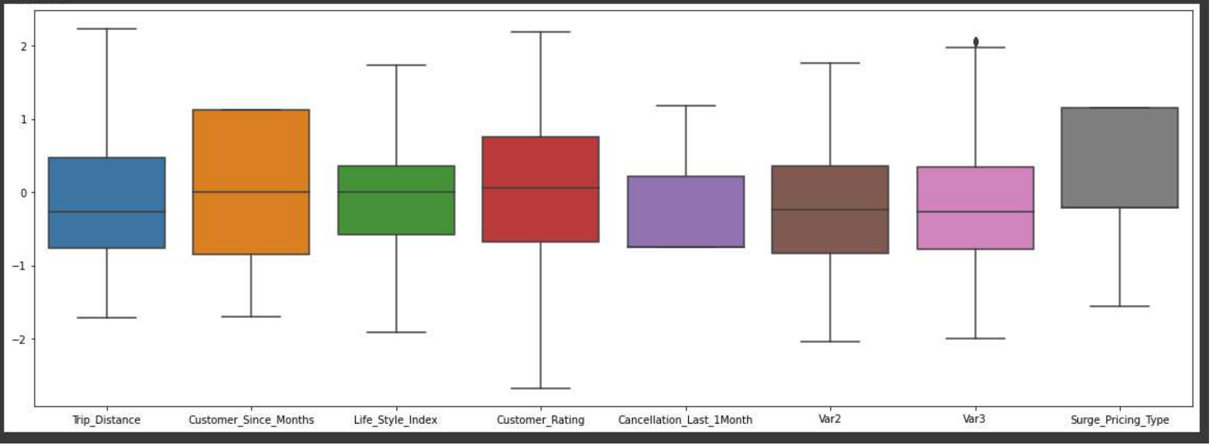
Removing outliers

df_N_NO= df_numerical data frame with no outliers

```
Q1 = df_numeric.quantile(0.25)
Q3 = df_numeric.quantile(0.70)
IQR = Q3 - Q1
df_N_NO = df_numeric[~((df_numeric < (Q1 - 1.5 * IQR)) |(df_numeric > (Q3 + 1.5 * IQR))).any(axis=1)]
```

```
# Draw boxplots for all numerical columns
fig = plt.figure(figsize=(20,7))
sns.boxplot(data=df_N_NO)
```

```
<Axes: >
```



```
print(df_numeric.shape)
```

```
(131662, 8)
```

```
print(df_N_NO.shape)
```

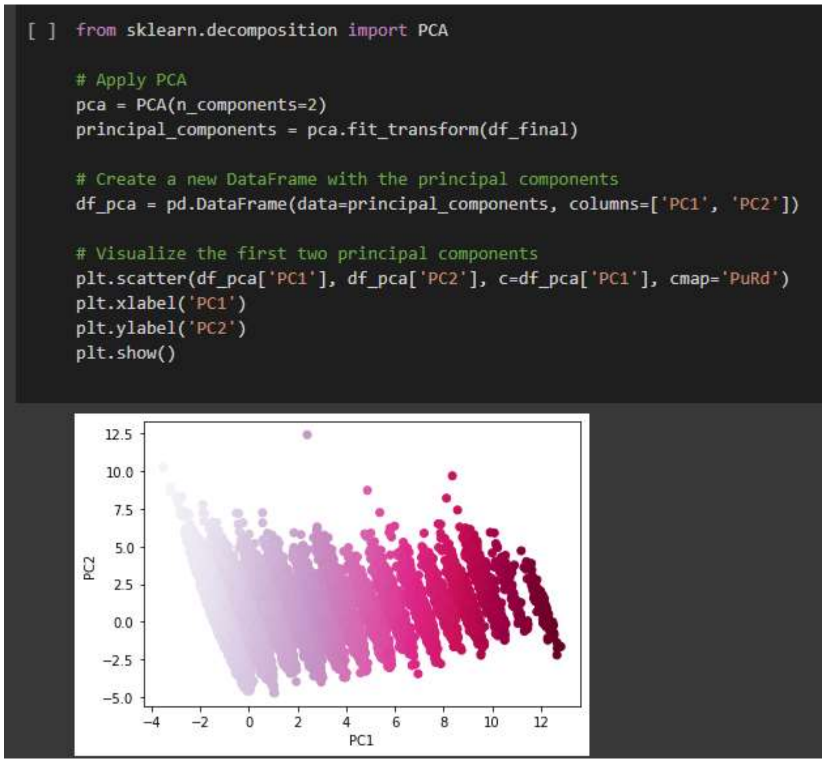
```
(101994, 8)
```

- I have created a correlation matrix to analyze the relationships between attributes, and have decided to exclude the attribute var1 as it has the lowest correlation with most of the other variables.

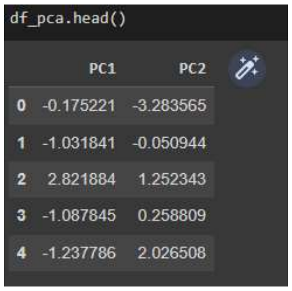
- Then I had plotted box plots to examine the numerical attributes and found that there were a significant number of outliers. After removing the outliers, I had observed that the number of rows in the data set reduced by approximately 30,000, and then confirmed the absence of outliers by plotting the box plots again.

Applying PCA for dimensionality reduction

1. Two Principal components



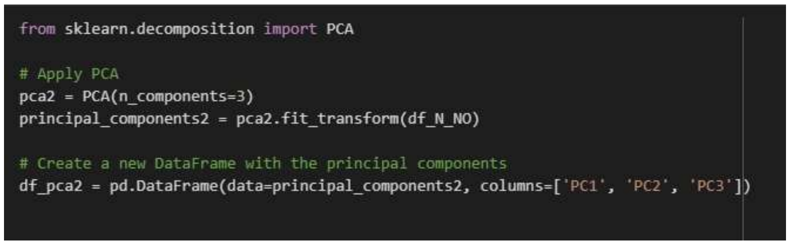
```
[ ]  from sklearn.decomposition import PCA

     # Apply PCA
     pca = PCA(n_components=2)
     principal_components = pca.fit_transform(df_final)

     # Create a new DataFrame with the principal components
     df_pca = pd.DataFrame(data=principal_components, columns=['PC1', 'PC2'])

     # Visualize the first two principal components
     plt.scatter(df_pca['PC1'], df_pca['PC2'], c=df_pca['PC1'], cmap='PuRd')
     plt.xlabel('PC1')
     plt.ylabel('PC2')
     plt.show()
```



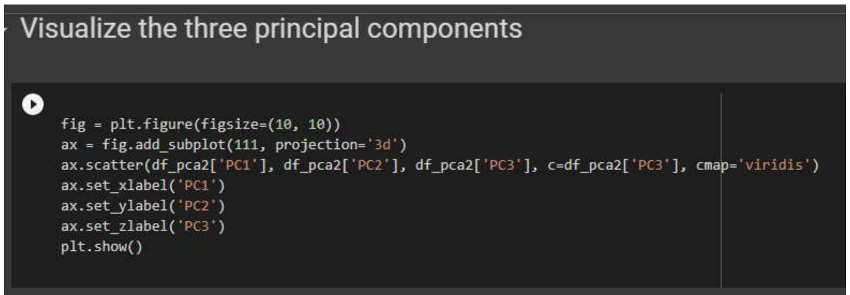
```
df_pca.head()
```

	PC1	PC2
0	-0.175221	-3.283565
1	-1.031841	-0.050944
2	2.821884	1.252343
3	-1.087845	0.258809
4	-1.237786	2.026508

2. Three Principal components

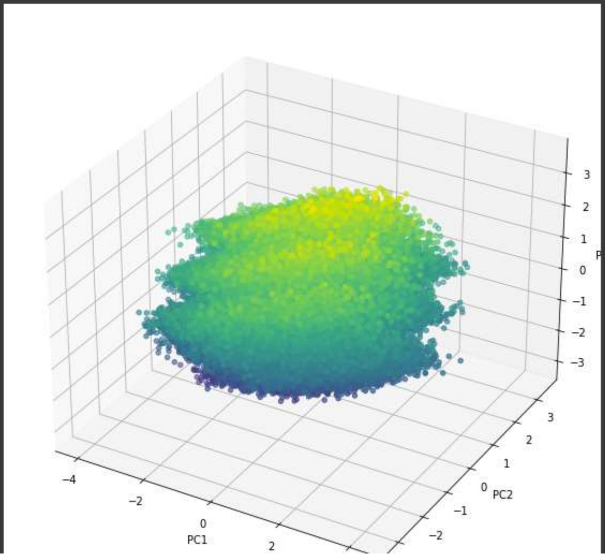


```
from sklearn.decomposition import PCA

# Apply PCA
pca2 = PCA(n_components=3)
principal_components2 = pca2.fit_transform(df_N_NO)

# Create a new DataFrame with the principal components
df_pca2 = pd.DataFrame(data=principal_components2, columns=['PC1', 'PC2', 'PC3'])
```



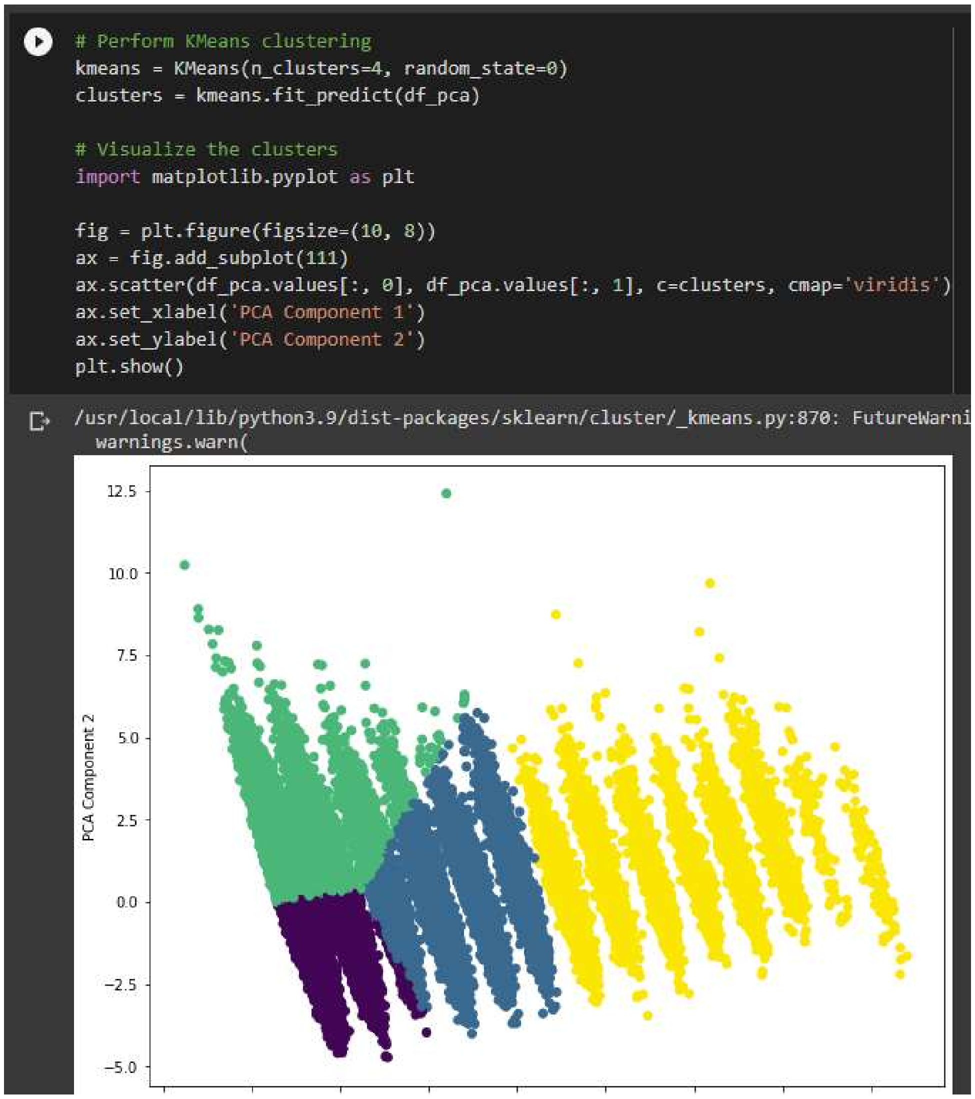
Visualize the three principal components

```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_pca2['PC1'], df_pca2['PC2'], df_pca2['PC3'], c=df_pca2['PC3'], cmap='viridis')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.show()
```



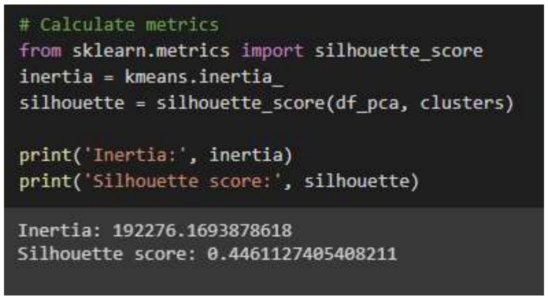
I had used PCA to reduce the dimensionality of the dataset. As there were 14 attributes and 131662 rows. By applying PCA, we can reduce the number of attributes (columns) while retaining most of the variability in the data. This can be helpful for clustering, and classification. Additionally, reducing the number of attributes can also help in reducing the computational cost of algorithms.

Segmentation using the data upon which PCA is applied



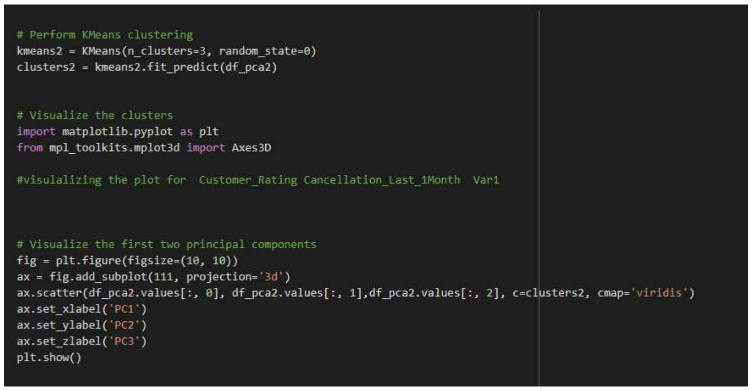
```
# Perform KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=0)
clusters = kmeans.fit_predict(df_pca)

# Visualize the clusters
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111)
ax.scatter(df_pca.values[:, 0], df_pca.values[:, 1], c=clusters, cmap='viridis')
ax.set_xlabel('PCA Component 1')
ax.set_ylabel('PCA Component 2')
plt.show()
```

```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarni
  warnings.warn(
```



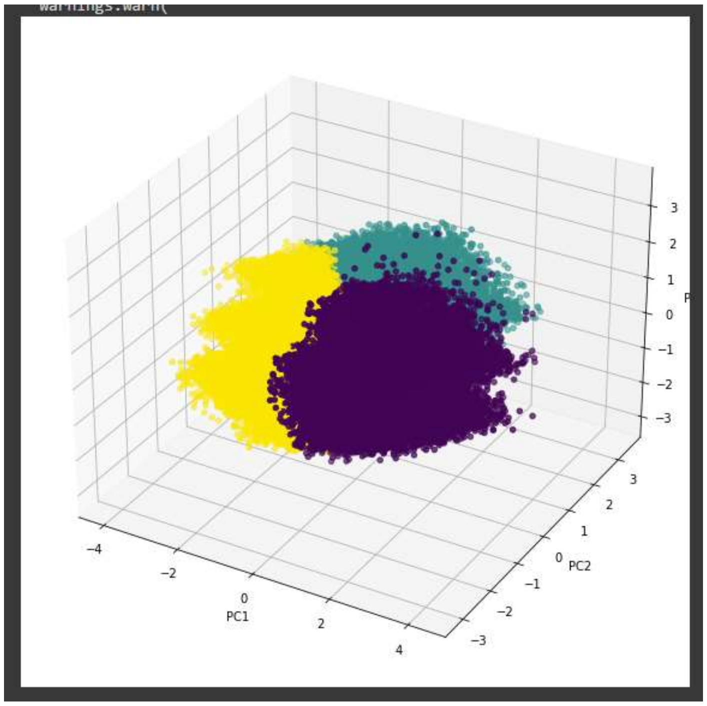
```
# Calculate metrics
from sklearn.metrics import silhouette_score
inertia = kmeans.inertia_
silhouette = silhouette_score(df_pca, clusters)

print('Inertia:', inertia)
print('Silhouette score:', silhouette)
```

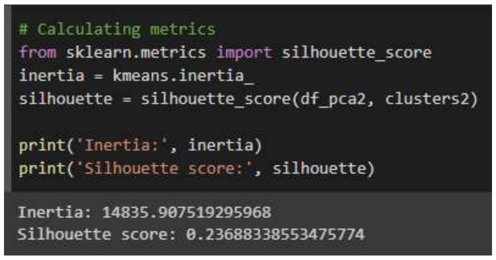
```
Inertia: 192276.1693878618
Silhouette score: 0.4461127405408211
```



```
# Perform KMeans clustering
kmeans2 = KMeans(n_clusters=3, random_state=0)
clusters2 = kmeans2.fit_predict(df_pca2)


# Visualize the clusters
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#visulalizing the plot for  Customer_Rating Cancellation_Last_1Month  Var1



# Visualize the first two principal components
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_pca2.values[:, 0], df_pca2.values[:, 1],df_pca2.values[:, 2], c=clusters2, cmap='viridis')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.show()
```

```
  warnings.warn(
```





```
# Calculating metrics
from sklearn.metrics import silhouette_score
inertia = kmeans.inertia_
silhouette = silhouette_score(df_pca2, clusters2)

print('Inertia:', inertia)
print('Silhouette score:', silhouette)
```

```
Inertia: 14835.907519295968
Silhouette score: 0.23688338553475774
```

<u>Use of K Means clustering for market segmentation</u>

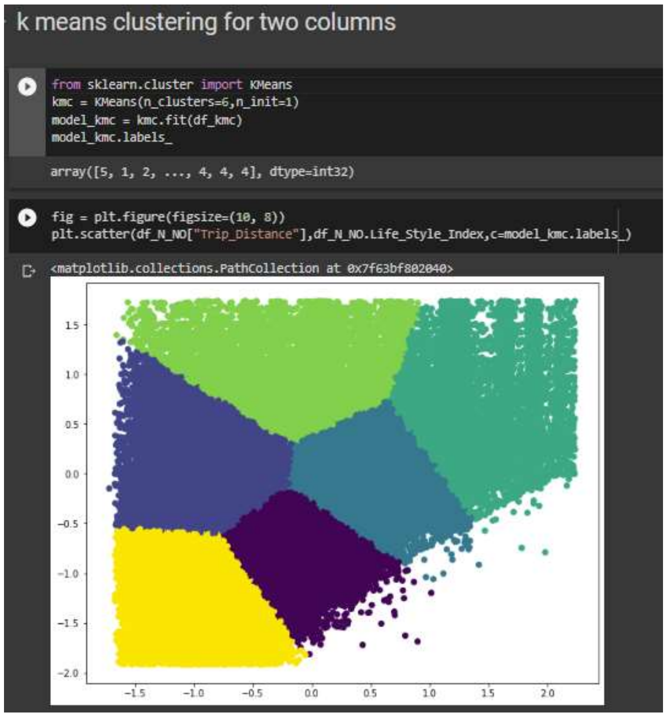
KMeans clustering is a popular unsupervised learning algorithm used for market segmentation. The primary goal of market segmentation is to divide customers into groups or clusters with similar characteristics so that companies can create targeted marketing strategies and improve customer experience.

KMeans clustering algorithm can be used for market segmentation on the above data by grouping customers with similar characteristics such as trip distance, type of cab, customer rating, etc. into clusters. The algorithm aims to minimize the distance between the data points within each cluster while maximizing the distance between the clusters. The result is a set of clusters that represent the different segments of customers.

Once the clusters are created, companies can tailor their marketing efforts to each segment, creating targeted campaigns that address the unique needs and preferences of each group. This approach can lead to increased customer loyalty, higher customer satisfaction, and ultimately, increased revenue for the company.

Overall, KMeans clustering can be a powerful tool for market segmentation, allowing companies to gain a deeper understanding of their customers and create more effective marketing strategies.

<u>Segmentation using the data upon which PCA is applied</u>



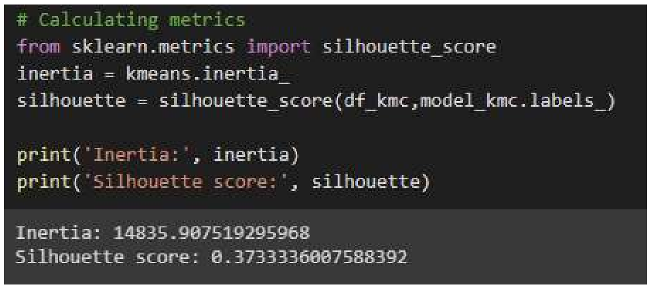
k means clustering for two columns

```
from sklearn.cluster import KMeans
kmc = KMeans(n_clusters=6,n_init=1)
model_kmc = kmc.fit(df_kmc)
model_kmc.labels_
```

```
array([5, 1, 2, ..., 4, 4, 4], dtype=int32)
```

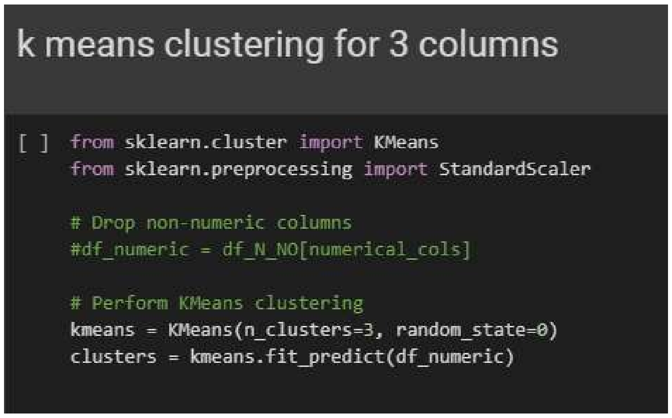
```
fig = plt.figure(figsize=(10, 8))
plt.scatter(df_N_NO["Trip_Distance"],df_N_NO.Life_Style_Index,c=model_kmc.labels_)
```

```
<matplotlib.collections.PathCollection at 0x7f63bf802040>
```



```
# Calculating metrics
from sklearn.metrics import silhouette_score
inertia = kmeans.inertia_
silhouette = silhouette_score(df_kmc,model_kmc.labels_)

print('Inertia:', inertia)
print('Silhouette score:', silhouette)
```

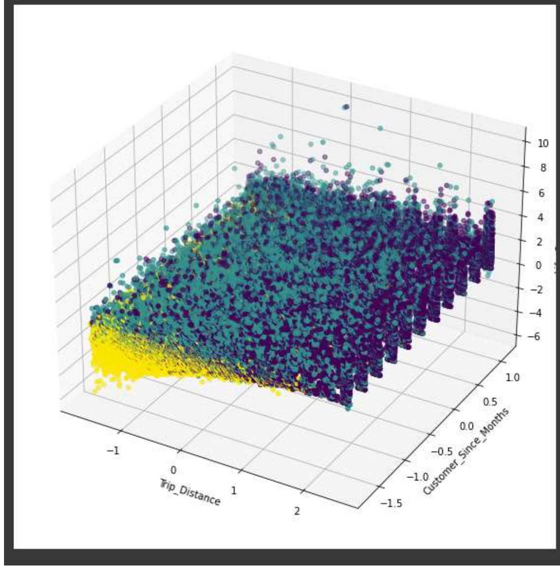
```
Inertia: 14835.907519295968
Silhouette score: 0.3733336007588392
```



k means clustering for 3 columns

```
[ ] from sklearn.cluster import KMeans
    from sklearn.preprocessing import StandardScaler

    # Drop non-numeric columns
    #df_numeric = df_N_NO[numerical_cols]

    # Perform KMeans clustering
    kmeans = KMeans(n_clusters=3, random_state=0)
    clusters = kmeans.fit_predict(df_numeric)
```

```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(df_numeric.values[:, 0], df_numeric.values[:, 1], df_numeric.values[:, 2], c=clusters, cmap='viridis')
ax.set_xlabel(numerical_cols[0])
ax.set_ylabel(numerical_cols[1])
ax.set_zlabel(numerical_cols[2])
plt.show()
```



Conclusion:-

Based on the analysis of the dataset consisting of 131,662 rows with 14 attributes , several insights can be drawn. Firstly, there are outliers present in the dataset , which are be removed to improve the accuracy of the model. Secondly, PCA is used to reduce the dimensionality of the dataset and identify the most important variables. Then by using the elbow method, it became possible to identify three as the optimal number of clusters for KMeans clustering. So these can be used for market segmentation to target customers based on their trip distance, type of cab requested, customer rating, lifestyle index, confidence in lifestyle index, destination type, cancellation rate, and surge pricing type. Overall, the use of PCA and KMeans clustering can help the companies to identify patterns and clusters in the data, which can be used to make informed decisions in marketing and business strategies.