PGP DSE
CAPSTONE PROJECT

FINAL REPORT

TRIP PRICING WITH TAXI MOBILITY ANALYTICS

Done by:

Vignesh. S
Abhijith S Varma
Siva Kumar. G
Ram Prakash. V
Vignesh .M

Mentor:

Ms. Vibha Santhanam

TABLE OF CONTENTS:

BUSINESS UNDERSTANDING:

We are predicting the surge price type for Sigma Cabs. Previously surge price was given by service providers, from that information they have captured surge price type, we are building a predictive model based on that surge price type, so that they can fix the fare beforehand.

DATA AND FINDINGS:

I. DATA:

STEP 1: Read the dataset.

```
[ ] df = pd.read_csv('sigma_cabs.csv')

[ ] df.head()
```

	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
0	T0005689460	6.77	B	1	2.42769	A	A	3.90500	0	40.0
1	T0005689461	29.47	B	10	2.78245	B	A	3.45000	0	38.0
2	T0005689464	41.58	NaN	10	NaN	NaN	E	3.50125	2	NaN
3	T0005689465	61.56	C	10	NaN	NaN	A	3.45375	0	NaN
4	T0005689467	54.95	C	10	3.03453	B	A	3.40250	4	51.0

STEP 2: Understand the dataset by checking

- Info
- Nunique
- Describe
- Shape

```
<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
```

VARIABLE CATEGORIZATION:

- No of rows: 131662
- Total Features/Columns :14
- Numerical Features :06
- Categorical Features:08
- Target: Surge_Pricing_Type (Multiclassification)

II. FINDINGS:

- F1- weighted score improved after changing the null value imputation method

  Before the new null value imputation method:

  F1-Weighted-Score =0.69

  After the new null value imputation method:

  F1-Weighted-Score =0.80

- Model Building with smote and without smote

BEFORE SMOTE:

```
[ ]  print(metrics.classification_report(ytest,ypred))
```

	precision	recall	f1-score	support
1.0	0.83	0.72	0.77	7201
2.0	0.79	0.86	0.82	14869
3.0	0.80	0.78	0.79	11651
accuracy			0.80	33721
macro avg	0.81	0.79	0.79	33721
weighted avg	0.80	0.80	0.80	33721

INFERENCE:

Scores of class1 was generally inconsistent in comparison with other classes.

AFTER SMOTE:

```
[ ]  print(metrics.classification_report(ytest,ypred))
```

	precision	recall	f1-score	support
1.0	0.86	0.78	0.82	12033
2.0	0.76	0.85	0.80	16937
3.0	0.82	0.77	0.79	14365
accuracy			0.80	43335
macro avg	0.81	0.80	0.80	43335
weighted avg	0.81	0.80	0.80	43335

INFERENCE:
Scores of class 1 became inline with scores of other classes

SCALING:

- Model building with Standard scaler and Min Max scaler.

INFERENCE:
There is not much difference in the performance of the models.
So, we took data with Min max scaling for further model building.

OVERVIEW OF THE FINAL PROCESS:

SALIENT FEATURES:

- This a Multi-classification dataset.
- This is a commercial dataset based on taxi surge type
- There is some confidentiality involved in this data such as Masked variables and Masked target.

Data Preprocessing

- Data type conversion
- Missing value Treatment
- Outlier Treatment
- Feature selection using Multicollinearity
- Scaling
- Encoding using dummies
- Treating class imbalance

Modelling:

- Base Models can be built using KNN.
- Furthermore, we and did one standalone model (Decision Tree) and three Ensemble models (Random Forest, Adaboost and Xgboost) using cross-validation method.
- After that we did stacking model using three of the best models from the above (Random Forest, Adaboost and Xgboost) and bagging model of that particular stacking model.
- Finally, we did Grid search hyper parameter tuning on the best model (stacking model mentioned above).

Evaluation:

- We use cross-validation method to obtain the average f1-weighted score.
- From this cross-validation method, we evaluate model by analyzing the following metrics:
  F1-score (Weighted Fi-score)
- Finally, we check whether the model is under fitting or overfitting by evaluating the Bias and Variance error.
- If the Bias and variance error in optimal level we draw inferences from the model.

Deployment:

- We are deploying the model as a web application using Flask, HTML5 and CSS.

STEP BY STEP WALKTHROUGH OF THE SOLUTION:

STEP 1: Read the dataset

STEP 2: Understand the dataset by checking

- Info
- Nunique
- Describe
- Shape

STEP 3: Data Preprocessing

- Data type conversion
- Missing value Treatment
- Outlier Treatment
- Feature selection using Multicollinearity
- Scaling
- Encoding using dummies
- Treating class imbalance

STEP4: MODEL BUILDING

- **Base Models** can be built using KNN.
- Furthermore, we and did one standalone model (Decision Tree) and three Ensemble models (Random Forest, Adaboost and Xgboost) using cross-validation method.
- After that we did stacking model using three of the best models from the above (Random Forest, Adaboost and Xgboost) and bagging model of that particular stacking model.
- Finally, we did Grid search hyper parameter tuning on the best model (Random Forest model mentioned above).

STEP 5: MODEL DEPLOYMENT

- We are deploying the model as a web application using Flask, HTML5 and CSS

MODEL EVALUATION:

- We use cross-validation method to obtain the average f1-weighted score.
- From this cross-validation method, we evaluate model by analyzing the following metrics:
  F1-score (Weighted Fi-score)
- Finally, we check whether the model is under fitting or overfitting by evaluating the Bias and Variance error.
- If the Bias and variance error in optimal level we draw inferences from the model.

Final model-Random Forest:

- The best model was stacking model but as the stacking model and bagging model is based on Random Forest.
- we are proceeding with hyper parameter tuning for Random Model.
- Random Forest is an ensemble modelling technique which uses bagging of multiple Decision trees and majority of the individual decision tree's prediction will be taken as the prediction for random forest.

Objective:

- Objective is to build a predictive model on the surge pricing type.

Parameters:

These are all the parameters for the final model

- *n_estimators*
- *criterion*
- *max_depth*
- *scoring*
- *cv*

We Evaluate the success by F1-weighted score.

We improved the robustness of the model by balancing the class weights of the target variables which is surge pricing type.

COMPARISION TO BENCHMARK:

KNN model is the Base model
Random Forest is the Final Model.

	Models	F1-Weighted	Bias Error	Variance Error
5	Stacking	0.806707	0.193293	0.0
6	Bagging	0.806663	0.193337	0.0
2	RandomForest	0.803783	0.196217	0.0
4	XGBoost	0.801025	0.198975	0.0
3	AdaBoosting	0.786245	0.213755	0.0
0	KNN	0.749143	0.250857	0.0
1	DecisionTree	0.713592	0.286408	0.0

BASE MODEL-K NEAREST NEIGHBOUR

```
kf = KFold(n_splits=5,shuffle=True,random_state=7)
scores = cross_val_score(estimator=knn,X=x,y=y,scoring='f1_weighted',cv=kf)
score_knn = np.mean(scores)
bias_knn = 1-np.mean(score_knn)
var_knn = np.std(score_knn)/np.mean(score_knn)

print('Average f1-weighted score = ',np.mean(score_knn))
print('Bias error(in %) = ',bias_knn * 100 , '%')
print('Variance error(in %) = ',var_knn * 100,'%')
```

```
Average f1-weighted score =  0.7491425413096751
Bias error(in %) =  25.085745869032493 %
Variance error(in %) =  0.0 %
```

FINAL MODEL -RANDOM FOREST:

```
[ ]  rf = RandomForestClassifier()
     kf = KFold(n_splits=5,shuffle=True,random_state=7)
     scores = cross_val_score(estimator=rf,X=x,y=y,scoring='f1_weighted',cv=kf)
     score_rf = np.mean(scores)
     bias_rf = 1-np.mean(score_rf)
     var_rf = np.std(score_rf)/np.mean(score_rf)

     print('Average f1-weighted score = ',np.mean(score_rf))
     print('Bias error(in %) = ',bias_rf * 100 , '%')
     print('Variance error(in %) = ',var_rf * 100,'%')

     Average f1-weighted score =  0.8036601430444581
     Bias error(in %) =  19.63398569555419 %
     Variance error(in %) =  0.0 %
```

INFERENCE:

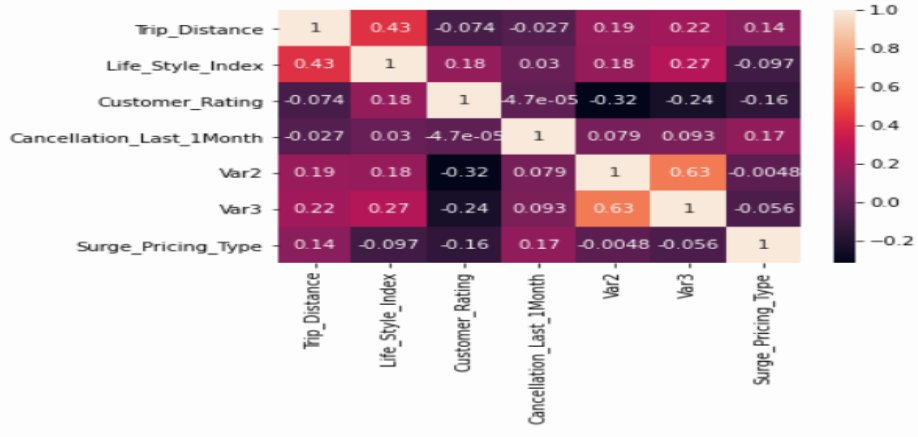
- Yes, we improved significantly on the benchmark.
- With comparison with the base model the f1-weighted score is improved with (6% efficiency) and the bias error is also reduced (6%).

VISUALIZATION:

- Multicollinearity
- Univariant Analysis
- Bivariant Analysis
- Multi Variant Analysis
- TABLEAU Analysis.

checking for multicollinearity

```
[ ]  sns.heatmap(df2.corr(),annot=True)

     <matplotlib.axes._subplots.AxesSubplot at 0x7fbc6b00e110>
```

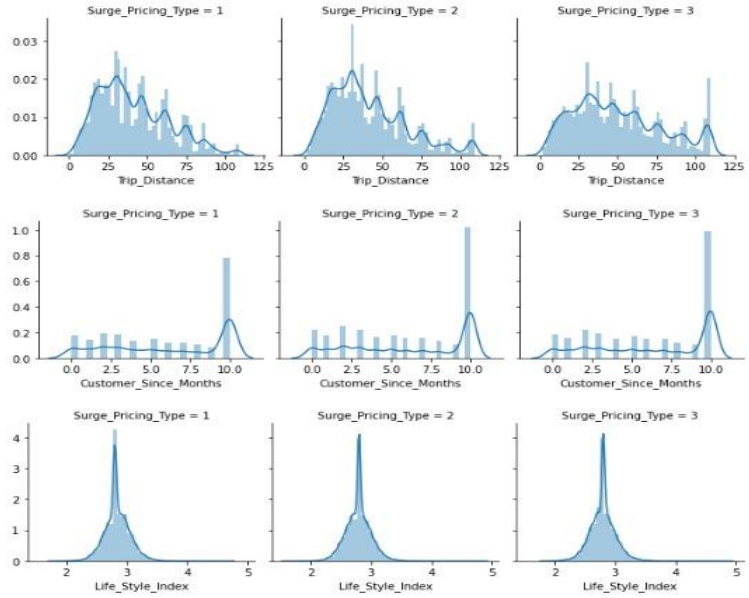


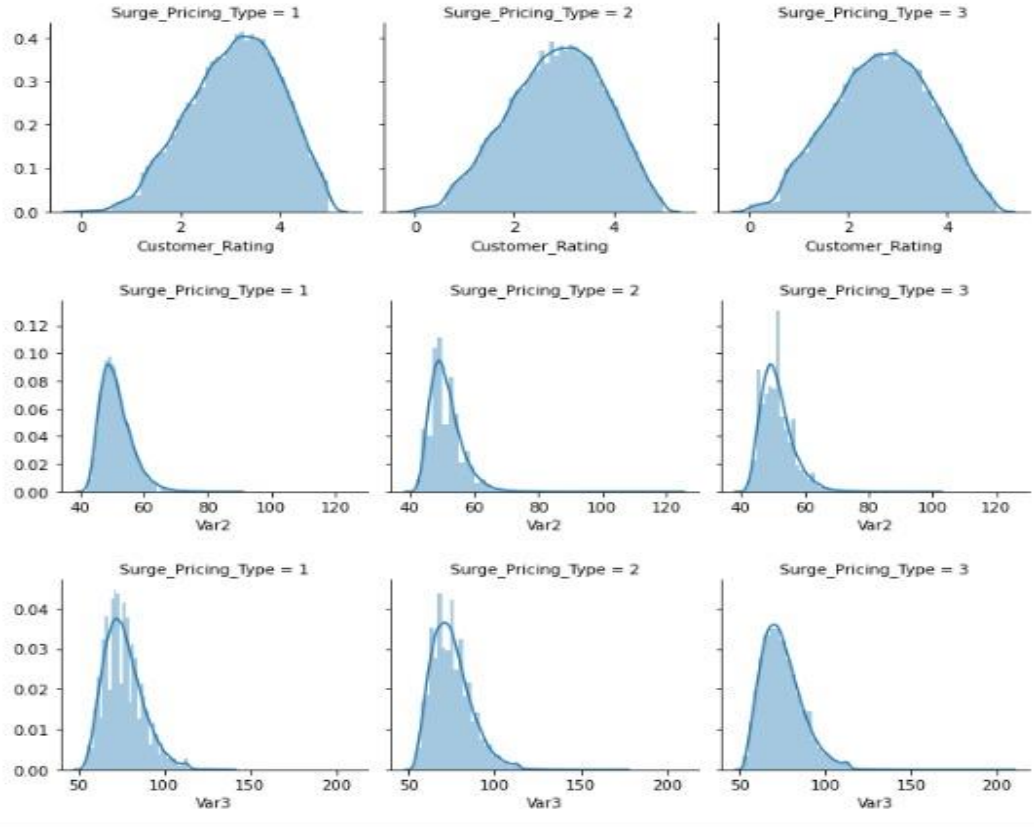

```
[ ]  # due to multi-collinearity between var2 and var3, we are dropping var2
```

UNIVARIANT ANALYSIS:NUMERICAL

In [ ]: # Distribution comparision of numerical features based on surge pricing type classes using distribution plot in facet grid

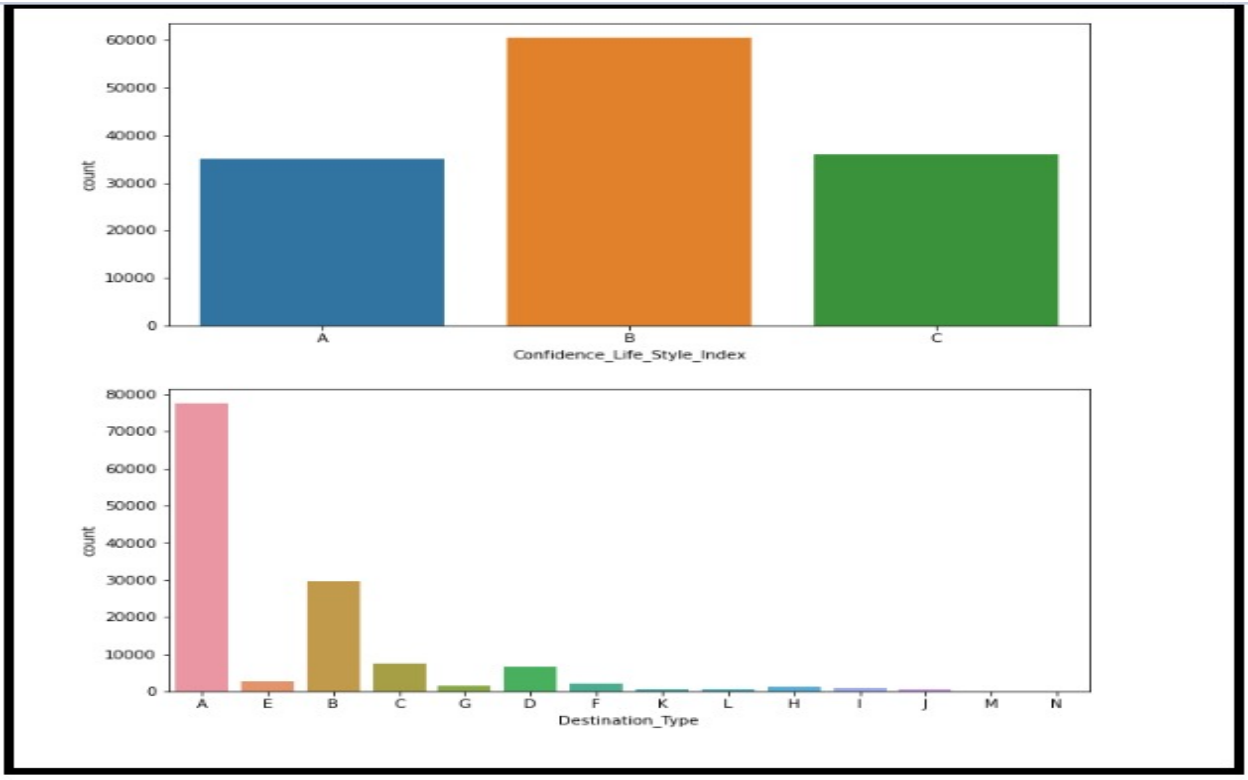
In [95]:
```
for i in num_features.columns:
    g = sns.FacetGrid(df, col='Surge_Pricing_Type')
    g.map(sns.distplot, i)
```





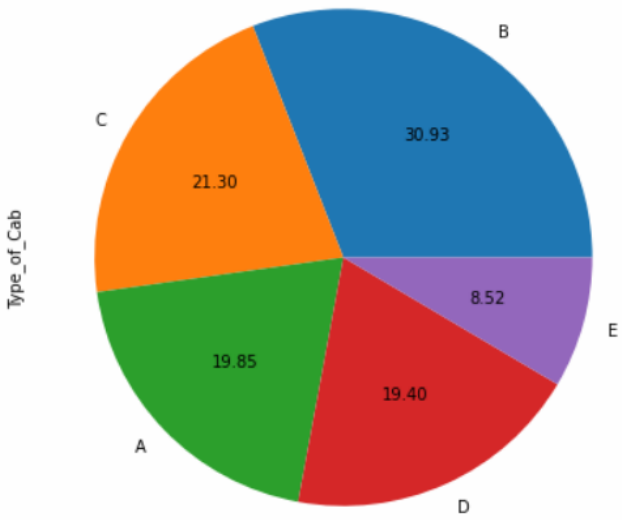

```
# Inference:

1.customer trip distance is maximum around 20 km to 40 km
2.Customer maximum trip distance is 120 km
3.Life style index of most customers is 2.5 to 3.4
```

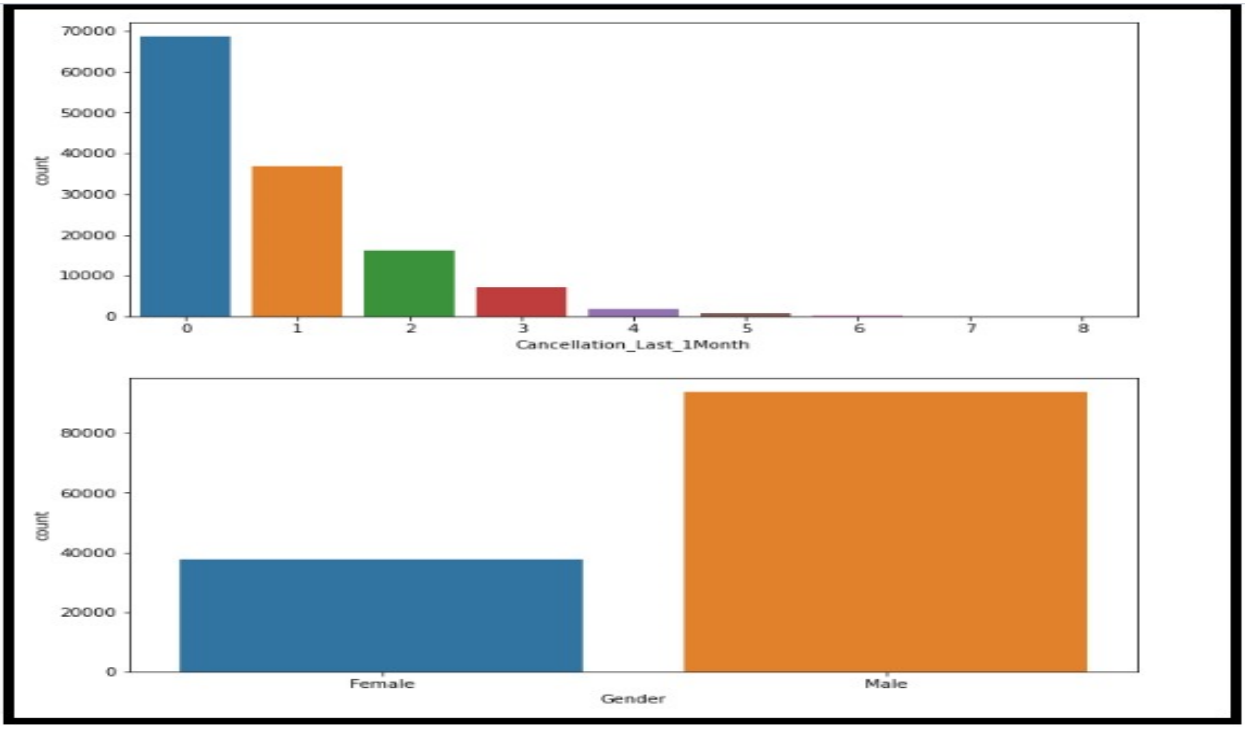
UNIVARIANT-CATEGORICAL




```
plt.figure(figsize=[18,7])
df2["Type_of_Cab"].value_counts(normalize=True).plot(kind="pie",autopct="%0.2f")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f82e270f5d0>
```

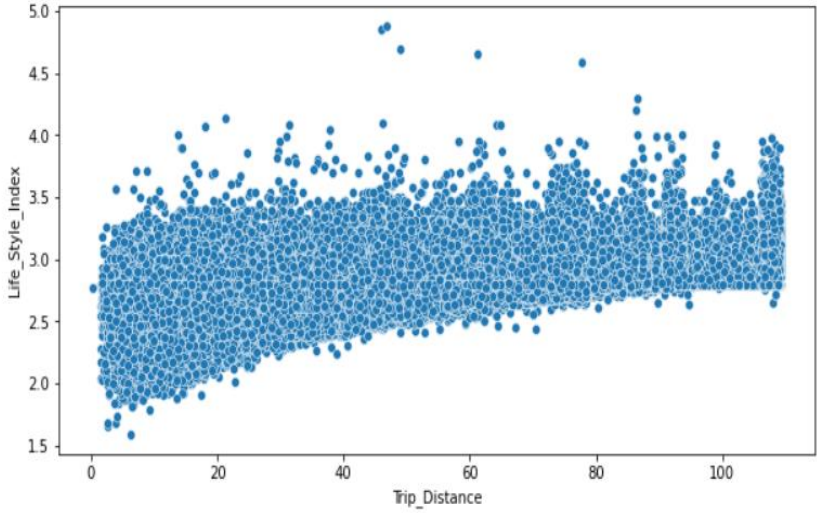




BIVARIANT ANALYSIS:

In [177]:
```
# comparing relation of trip distance and life_style_index
```

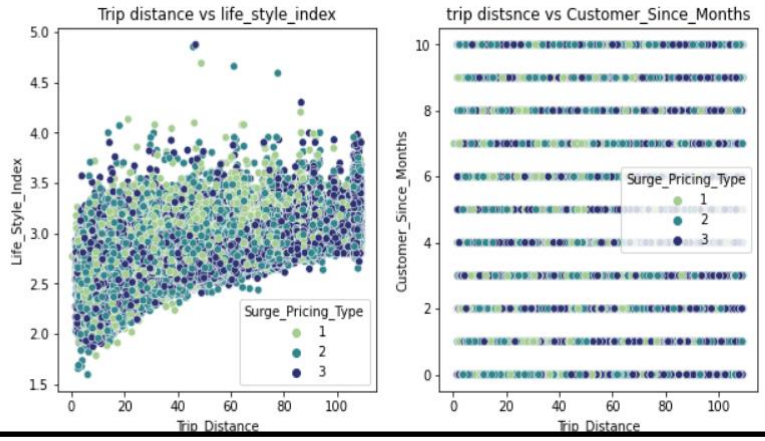
In [180]:
```
sns.scatterplot(df['Trip_Distance'],df["Life_Style_Index"])
plt.show()
```




```
# Inference

Trip distance and life style index is linearly correlated,as life_style_index increases trip distance of
customer is also long.
```

MULTIVARIANT ANALYSIS:

In [184]:
```
fig,(ax1,ax2)=plt.subplots(1,2)

ax1=plt.subplot(1,2,1)
plt.title("Trip distance vs life_style_index")
sns.scatterplot(y="Life_Style_Index",x="Trip_Distance",hue="Surge_Pricing_Type",data=df,ax=ax1,palette='crest')


ax2=plt.subplot(1,2,2)
plt.title("trip distsnce vs Customer_Since_Months")
sns.scatterplot(y="Customer_Since_Months",x="Trip_Distance",hue="Surge_Pricing_Type",data=df,ax=ax2,palette='crest')
plt.show()
```



In [185]:
```
sns.scatterplot(data=df,x="Trip_Distance",y='Life_Style_Index',hue="Type_of_Cab",size='Customer_Rating')
plt.show()
```

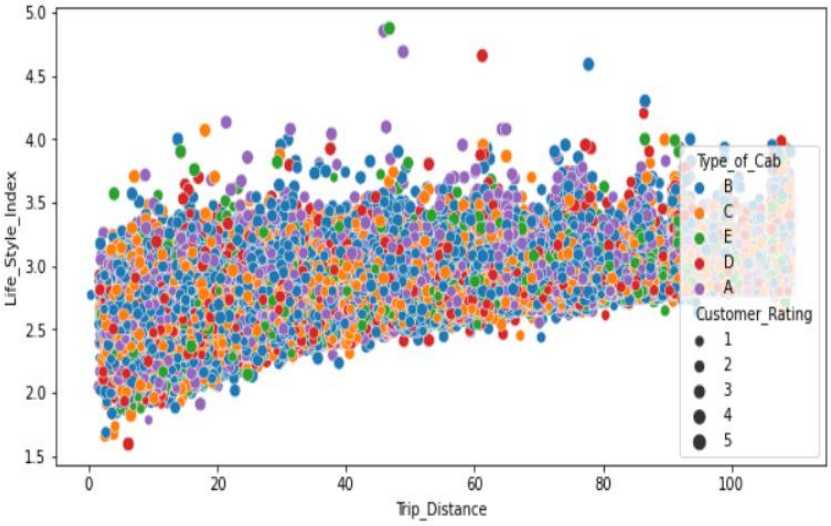
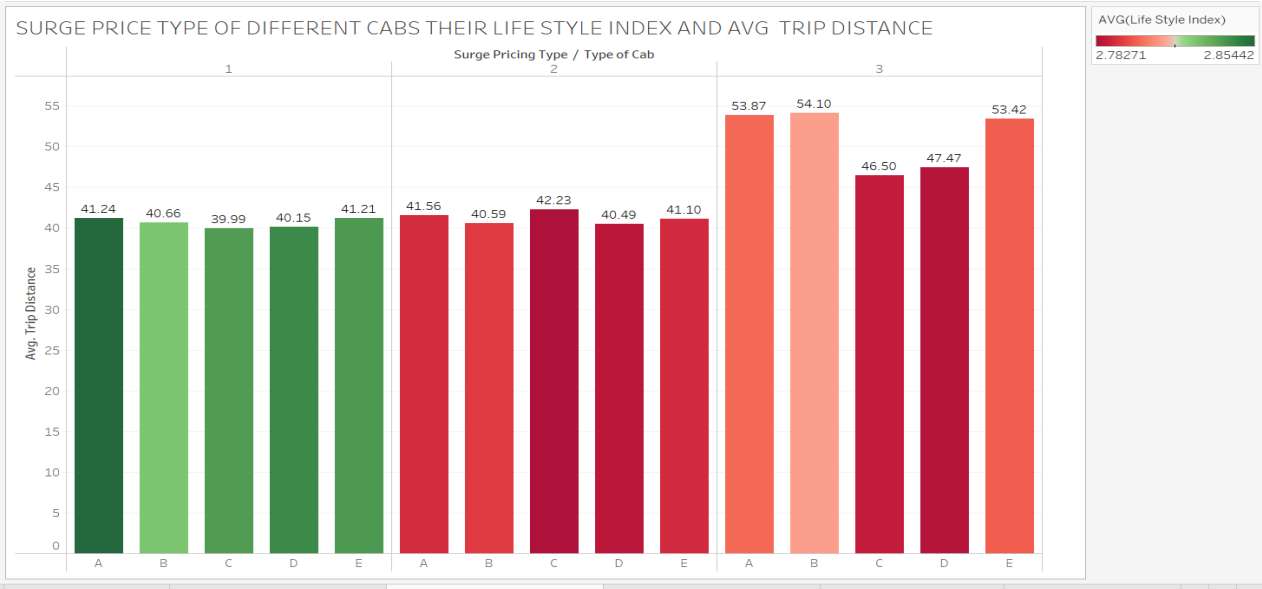
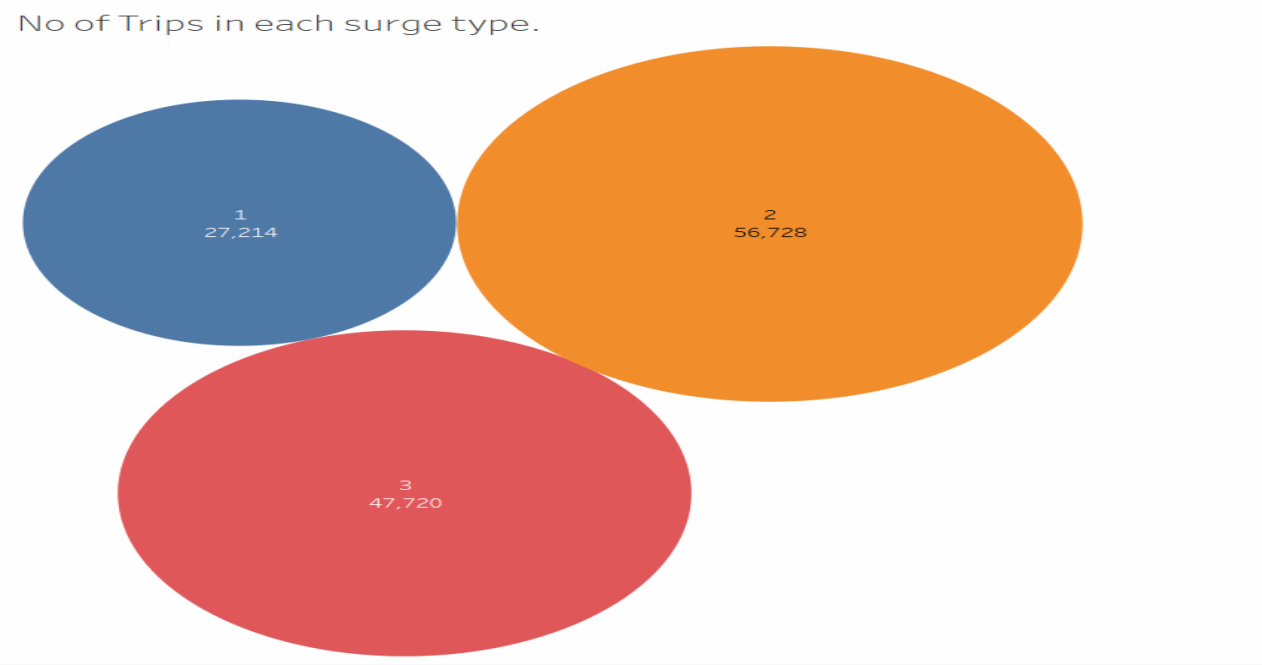


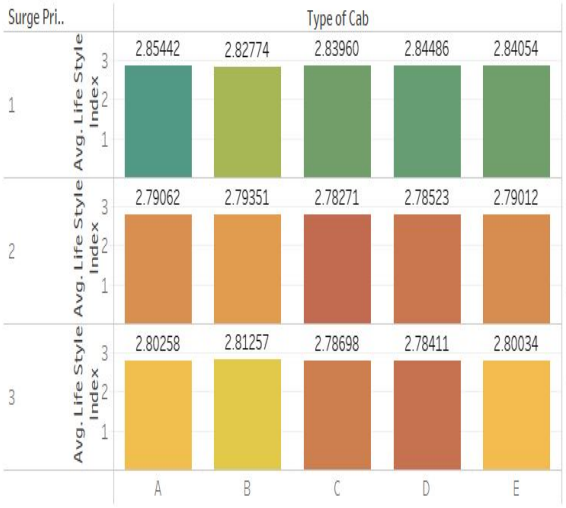
TABLEAU ANALYSIS:



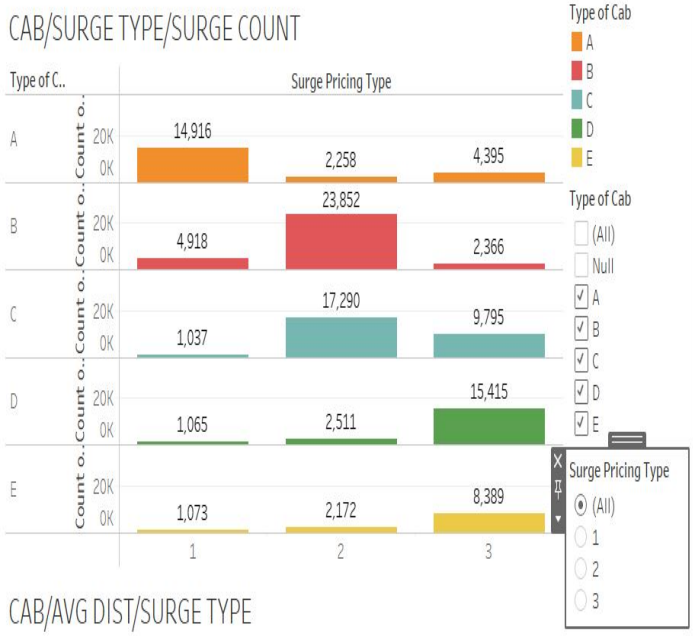


- Surge_type1 is having less trip_distance but average life style index of the customer is high so its assumed that they are capable to pay high surge price even though the rate is high and their aim to reach destination on time.
- Surge_type_2 is having almost same trip distance but their life_style_index is comparatively lower so they avoid high surge price.
- Even though Surge_type_3 is having higher distance since their life_style index is medium they are capable of paying slightly higher prices.

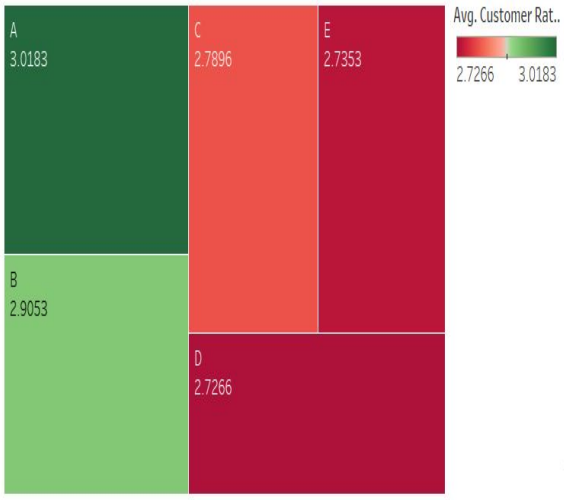
CAB/AVG LIFE STYLE INDEX



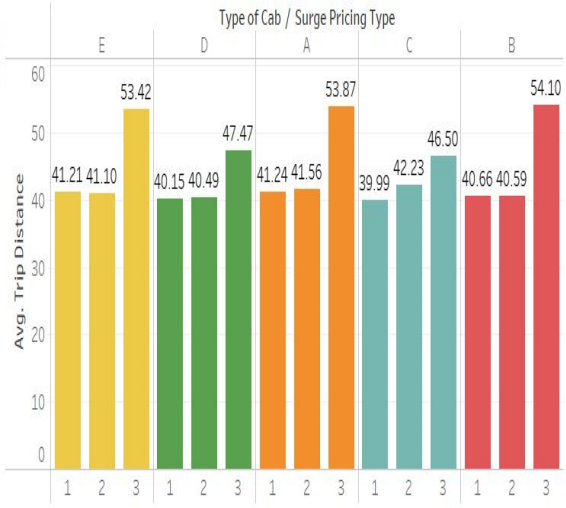
CAB/SURGE TYPE/SURGE COUNT



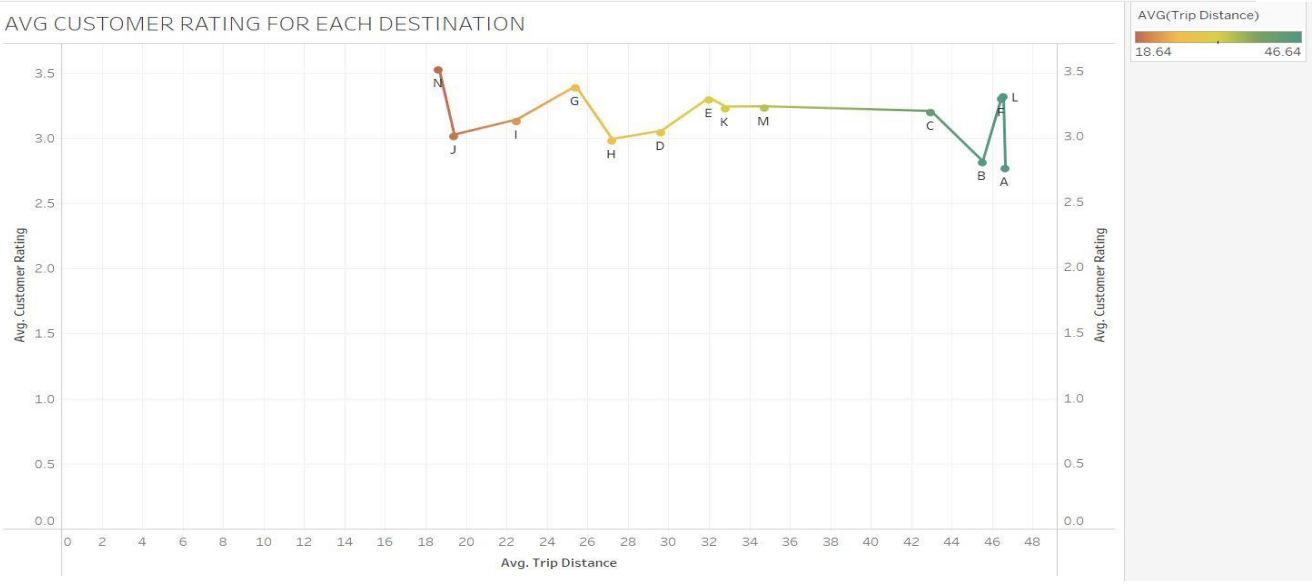
CAB/AVG RATING



CAB/AVG DIST/SURGE TYPE



AVG CUSTOMER RATING FOR EACH DESTINATION



BUSINESS INSIGHTS:

Our project is for commercial purpose. We are helping Sigma Cabs, a startup cab aggregator to predict the surge price type with the data they have provided

IMPLICATIONS:

- Our f1-Weighted score is only 80% so while in production the model can emit some false predictions which can affect the business.
- In order to get more accurate predictions, we need more data for the model to train.

LIMITATIONS:

- Var1, Var2, Var3 is masked variables so we don't know exactly what role it plays in the data.
- Target Variable is also masked due to confidentiality issues.
- There is considerable multicollinearity between var2 and var3 which can affect the performance of the model in production.
- With general data cleaning the model performance was poor.

CLOSING REFLECTIONS:

- We got exposure real world data and its implications.
- We learnt about the process of cab aggregation.
- We learnt about the Simple imputer library in sklearn.
- After doing in depth analysis in Tableau, we learned more about the target variable which is surge pricing type.

MODEL DEPLOYMENT:

Deployment is done using Flask, HTML, CSS.

We have attached the deployment file along with this report for your reference.

FUTURE SCOPE:

- We will ask to the client for more information regarding the price, Traffic level at the time of the trip.
- We will use more advanced modelling techniques in order to make the prediction more accurate.