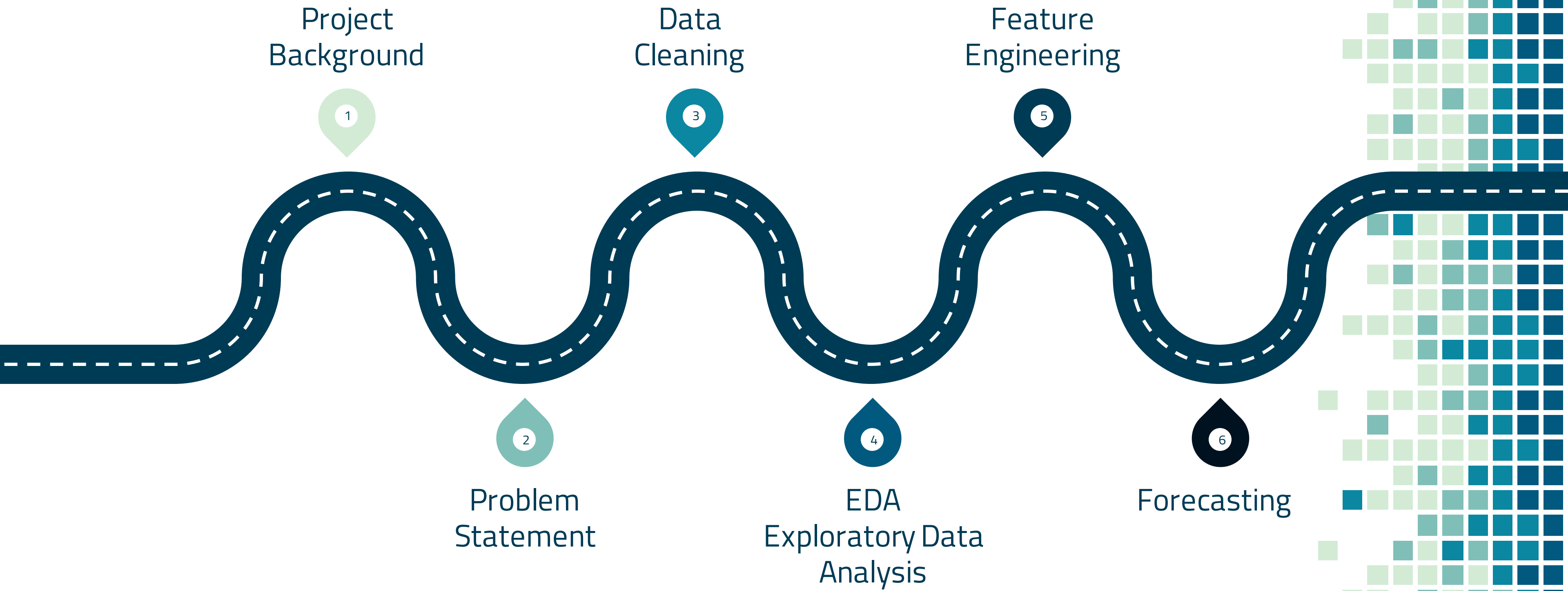
ROADMAP


Project Background
1
Problem Statement
2
Data Cleaning
3
EDA Exploratory Data Analysis
4
Feature Engineering
5
Forecasting
6

1. Project Background

Project Background

Scholastic Travel Company (STC)

This is an educational tourism firm, which has the ability to coordinate the numerous details associated. The contract renewal opportunities would begin for customers who had gone on an STC trip in 2012.

Based on past experiences models could be constructed to predict whether or not a customer would book again in 2013.with this model marketing strategy can be adjusted to save cost and improve yield.

2.

Problem Statement

Problem Statement

Prediction Task and Available Data

Predict which customers would book with STC in the 2013-14 school year.

For training the model we can use the data from 2012-13 school year. The sample size is nearly 2400 groups.

Due to the extensive numbers of columns we took the approach of selecting them based on their type (Numerical &...) over the next slides we will perform data cleaning, data processing, data visualization as well as method selection for our forecasting.

The methods which we used are Random Forest, Decision Tree, KNN, Logistic Regression & SVC. We will calculate the accuracy and we will select the best one accordingly.

3.
Data Preprocessing

Data Cleaning

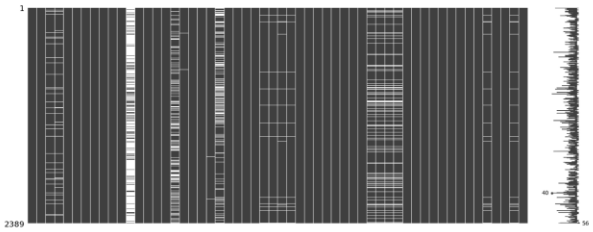
1. Missing Data

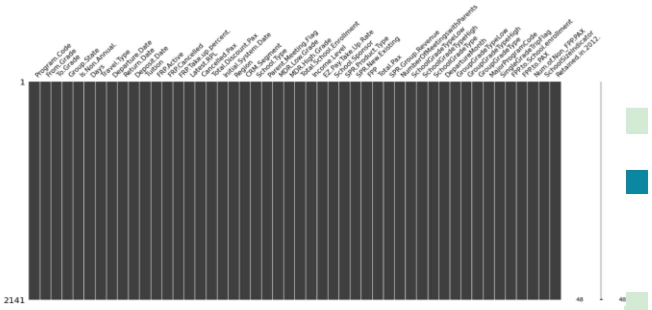
ID	0	Total.Discount.Pax	0	NumberOfMeetingswithParents	0
Program.Code	0	Initial.System.Date	8	FirstMeeting	337
From.Grade	127	Poverty.Code	599	LastMeeting	337
To.Grade	150	Region	0	DifferenceTraveltoFirstMeeting	337
Group.State	0	CRM.Segment	4	DifferenceTraveltoLastMeeting	337
Is.Non.Annual.	0	School.Type	0	SchoolGradeTypeLow	0
Days	0	Parent.Meeting.Flag	0	SchoolGradeTypeHigh	0
Travel.Type	0	MDR.Low.Grade	68	SchoolGradeType	0
Departure.Date	0	MDR.High.Grade	68	DepartureMonth	0
Return.Date	0	Total.School.Enrollment	91	GroupGradeTypeLow	0
Deposit.Date	0	Income.Level	62	GroupGradeTypeHigh	0
Special.Pay	1919	EZ.Pay.Take.Up.Rate	0	GroupGradeType	0
Tuition	0	School.Sponsor	0	MajorProgramCode	0
FRP.Active	0	SPR.Product.Type	0	SingleGradeTripFlag	0
FRP.Cancelled	0	SPR.New.Existing	0	FPP.to.School.enrollment	91
FRP.Take.up.percent.	0	FPP	0	FPP.to.PAX	0
Early.RPL	673	Total.Pax	0	Num.of.Non_FPP.PAX	0
Latest.RPL	19	SPR.Group.Revenue	0	SchoolSizeIndicator	91
Cancelled.Pax	0			Retained.in.2012.	0

Drop

```
df1 = df1.dropna(axis = 0)
```

- All the missing values are eliminated





Data Cleaning

2.Duplicates

```
df1.duplicated()
```

```
0       False
1       False
2       False
4       False
5       False
        ...
2381    False
2382    False
2384    False
2385    False
2388    False
Length: 2141, dtype: bool
```

Drop

- Based on the length comparison, there is no duplicate.

```
df1.reset_index(inplace=True)
```

```
df1.drop_duplicates()
```

0	0	HS
1	1	HC
2	2	HD
3	4	HD
4	5	HC
...	...	...
2136	2381	SC
2137	2382	SC
2138	2384	HC
2139	2385	HD
2140	2388	HD

2141 rows × 49 columns

Data Cleaning

Step 3: Fix structural errors. ...

Step 4: Filter unwanted outliers. ...

Step 5: Validate and QA.

Encoding objects to numerical data

```python
def object_to_int(dataframe_series):
    if dataframe_series.dtype=='object':
        dataframe_series = LabelEncoder().fit_transform(dataframe_series)
    return dataframe_series
```

```python
df2 = df1.apply(lambda x: object_to_int(x))
df2.head()
```

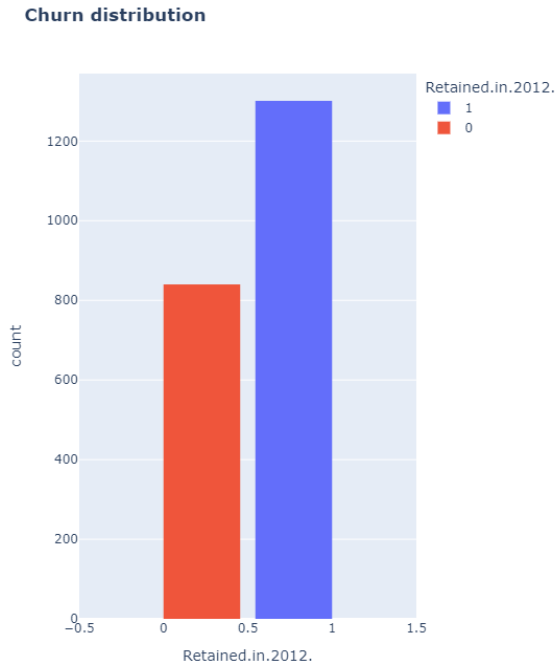
	index	Program.Code	From.Grade	To.Grade	Group.State	Is.Non.Annual.	Days	Travel.Type	Departure.Date	Return.Date	...	GroupGradeTypeLow	GroupGra
0	0	13	4.0	4.0	4	0	1	0	0	0	...	2	
1	1	4	8.0	8.0	3	0	7	0	0	3	...	3	
2	2	5	8.0	8.0	7	0	3	0	1	1	...	3	
3	4	5	6.0	8.0	7	0	6	3	2	3	...	3	
4	5	4	10.0	12.0	16	0	4	0	3	2	...	1	

5 rows × 49 columns

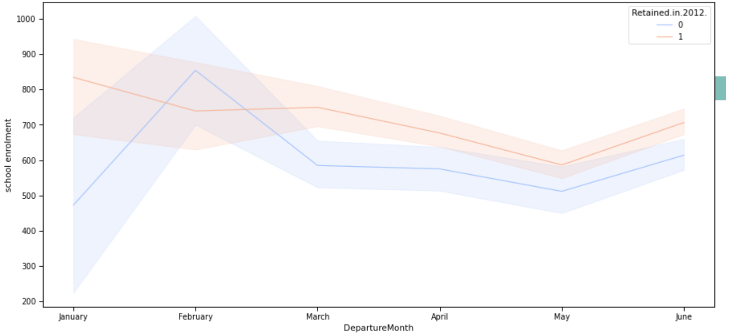
4.

EDA
Exploratory Data Analysis

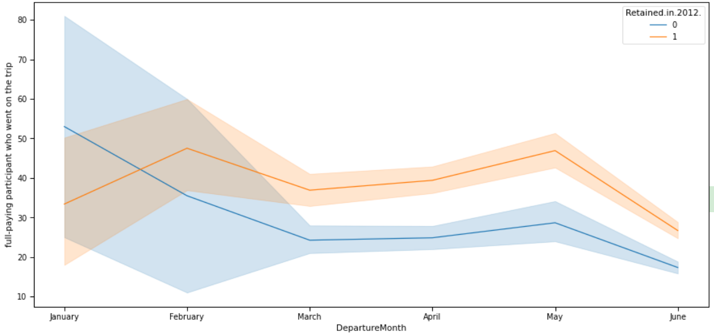
Distribution of predicted participation in 2012



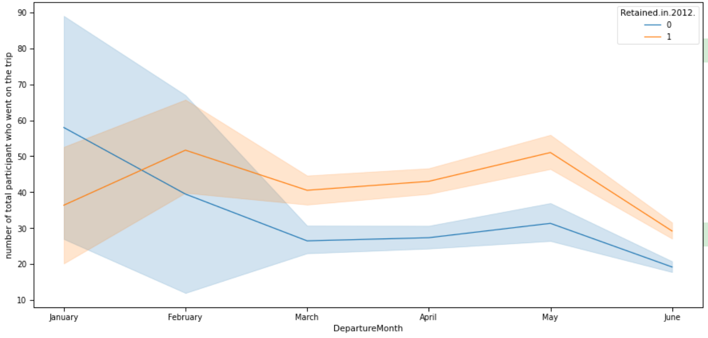
School Enrolment VS Departure Month



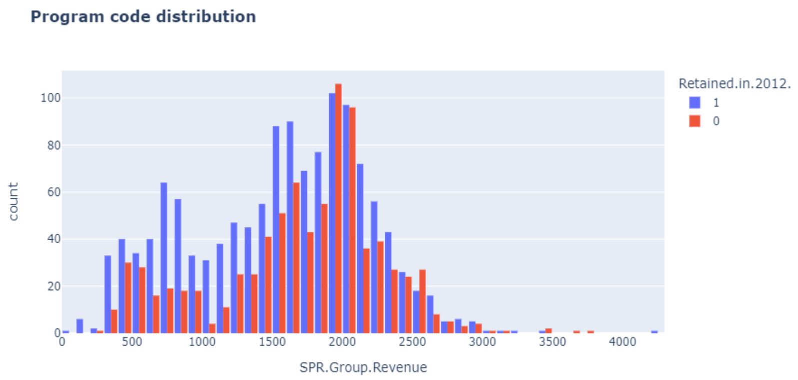
FPP VS Departure Month

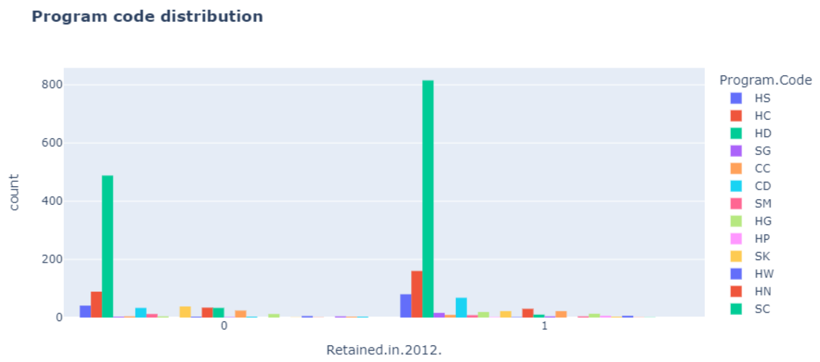


Total.Pax VS Departure Month

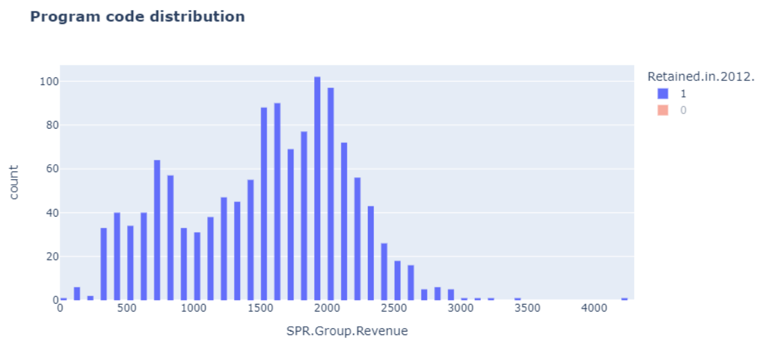


HD program will participate Most at school trips

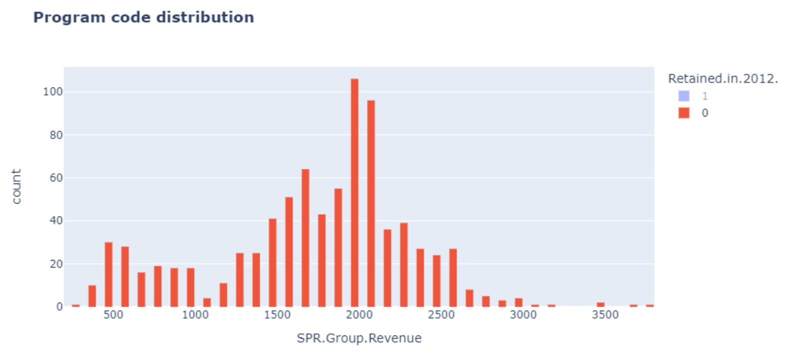


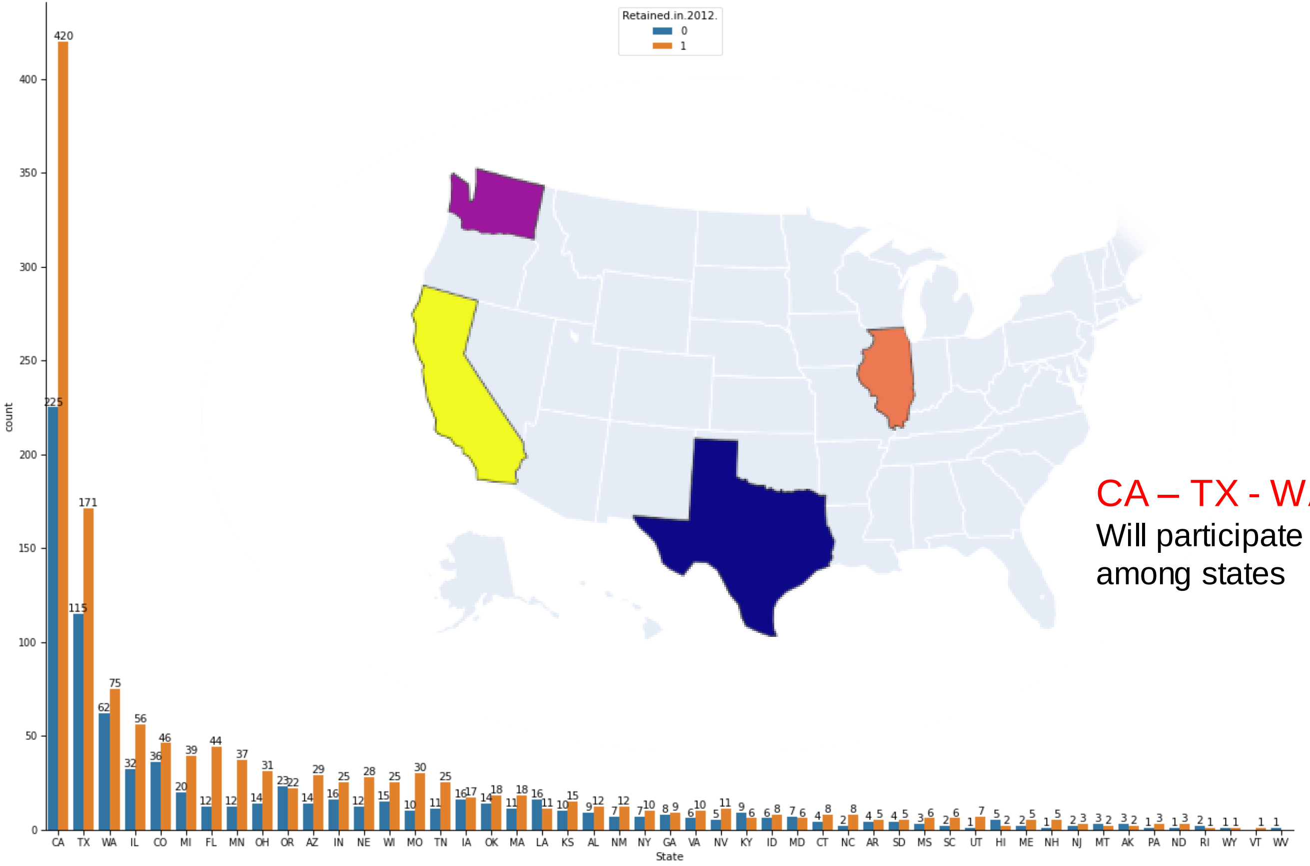


Retained=01



Retained=0



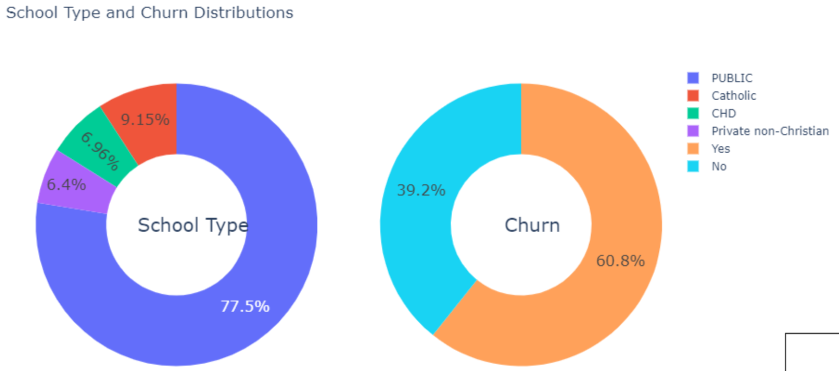


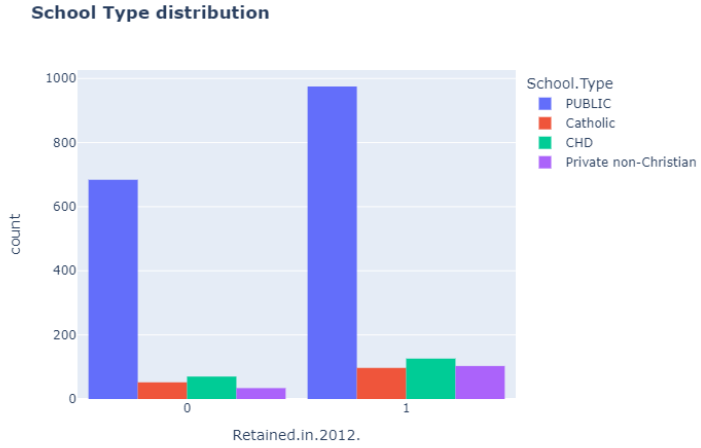
CA – TX - WA - IL

Will participate the most among states

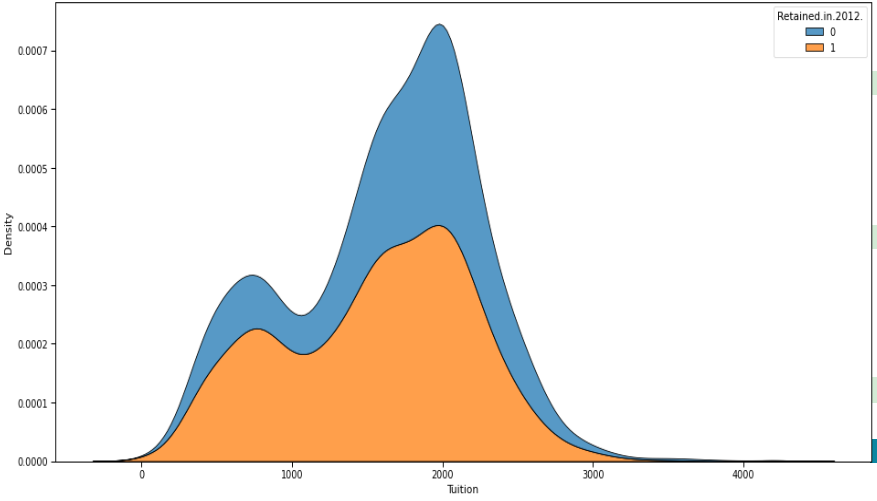
Distribution of Retain and Churn based on the states

Public School Students will participate the most in school trip





Highly paid tuition student tend to participate less in school trip



5.
Feature Engineering

Correlation

Target Value Correlation

In this section we have calculated the correlation with the target value which is Retained column.

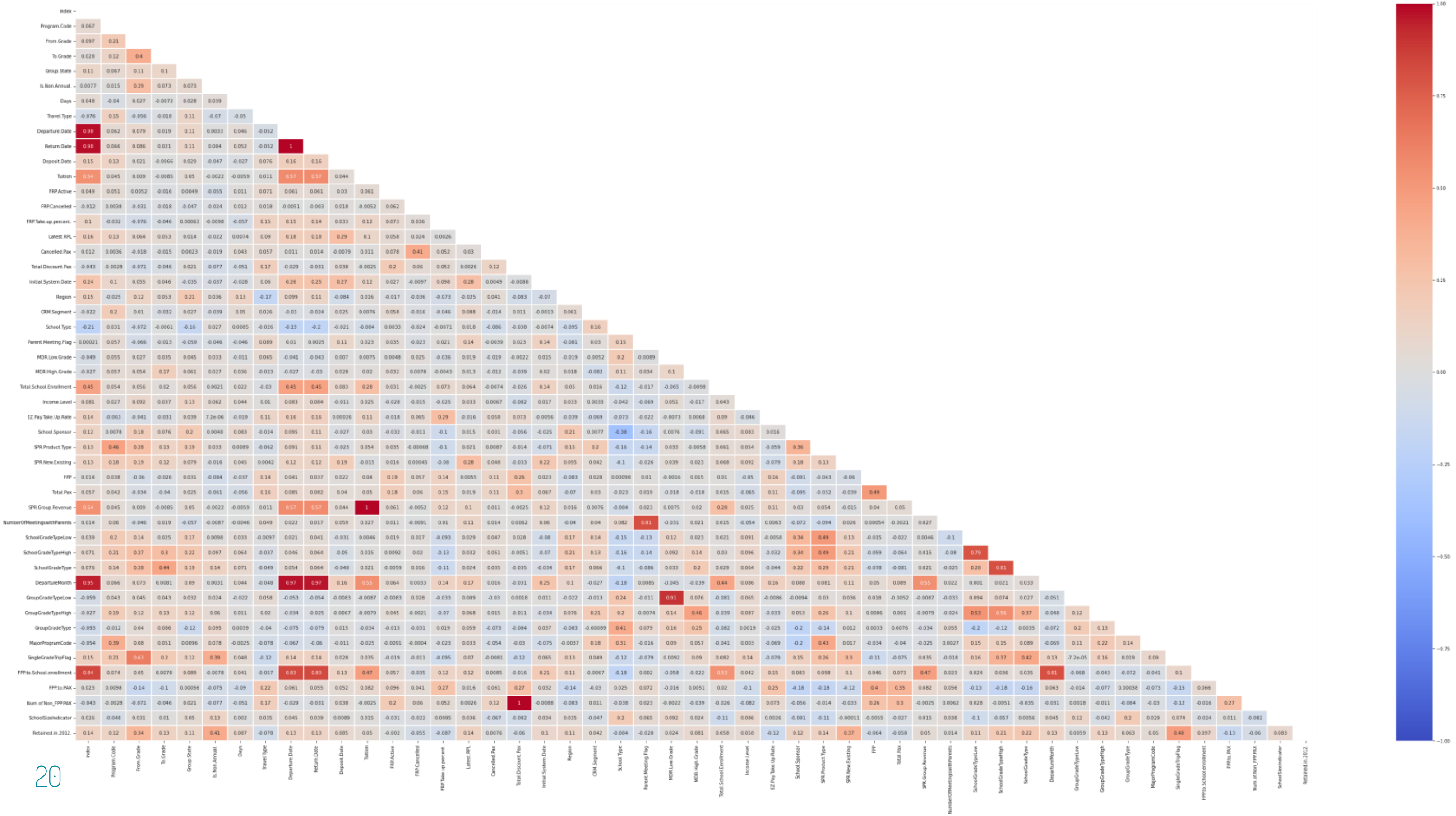
Total Values Correlation

In the next slide we have calculated all columns correlation together. In this strategy we will consider before having the first Retained forecast made by the company team. This approach allows us to consider new columns for our forecasting.

```
df1.corr()['Retained.in.2012.']
```

```
index                          -0.142795
From.Grade                      0.085849
To.Grade                       -0.194510
Is.Non.Annual.                 -0.410288
Days                           -0.037605
Tuition                        -0.114570
FRP.Active                      0.249512
FRP.Cancelled                   0.061124
FRP.Take.up.percent.           -0.023139
Cancelled.Pax                   0.042182
Total.Discount.Pax              0.204881
CRM.Segment                    -0.001991
Parent.Meeting.Flag            -0.027961
MDR.High.Grade                 -0.122477
Total.School.Enrollment         0.114613
EZ.Pay.Take.Up.Rate            -0.009721
School.Sponsor                  0.120232
FPP                             0.255944
Total.Pax                       0.254632
SPR.Group.Revenue              -0.114570
NumberOfMeetingswithParents    -0.063132
SingleGradeTripFlag             0.483699
FPP.to.School.enrollment        0.061485
FPP.to.PAX                      0.156437
Num.of.Non_FPP.PAX              0.204881
Retained.in.2012.               1.000000
Name: Retained.in.2012., dtype: float64
```

Total Values Correlation

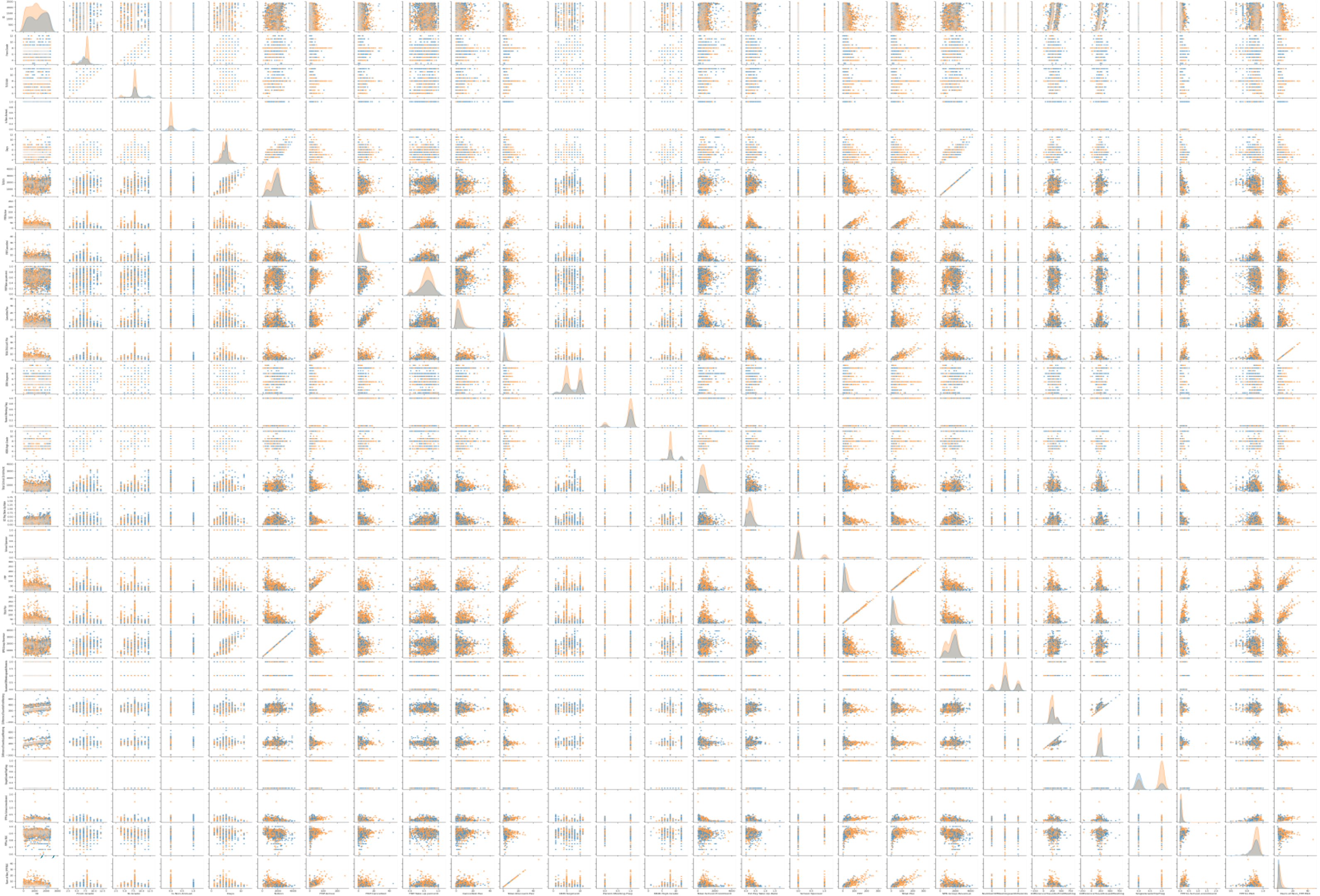


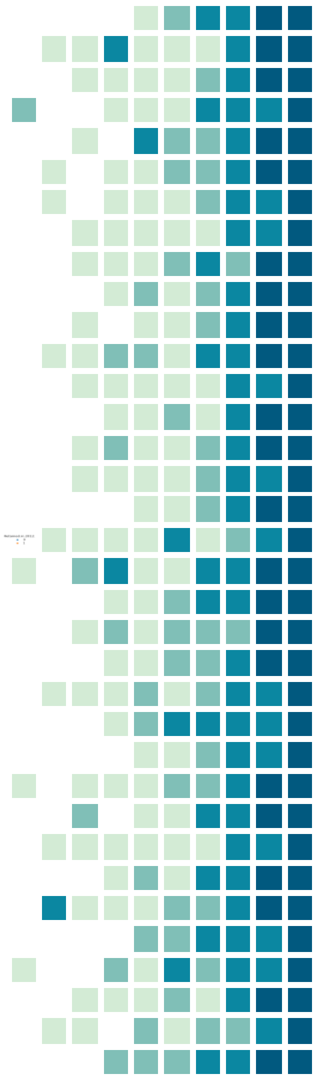
Correlation greater than 0.3

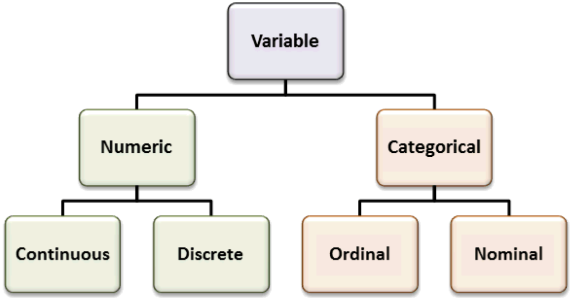
We have considered correlation for all columns which are greater than 0.3, these columns with high value are the fundamental data for forecasting.

```
target_corr=abs(corr)
positive_corr_target=target_corr[target_corr>(0.3)]
positive_corr_target
```

	index	Program.Code	From.Grade	To.Grade	Group.State	Is.Non.Annual.	Days	Travel.Type	Departure.Date	Return.Date	...
index	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.980376	0.982706	...
Program.Code	NaN	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
From.Grade	NaN	NaN	1.000000	0.395149	NaN	NaN	NaN	NaN	NaN	NaN	...
To.Grade	NaN	NaN	0.395149	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	...
Group.State	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	...
Is.Non.Annual.	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	NaN	NaN	NaN	...
Days	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	...
Travel.Type	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	...
Departure.Date	0.980376	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.998517	...
Return.Date	0.982706	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.998517	1.000000	...
Deposit.Date	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Tuition	0.544229	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.568451	0.568583	...
FRP.Active	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
FRP.Cancelled	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
FRP.Take.up.percent.	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
Latest.RPL	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...








```
numerical_cols = ['Days', 'Tuition', 'FRP.Active','FRP.Cancelled','FRP.Take.up.percent.','Cancelled.Pax','Total.Discount.Pax','To
```

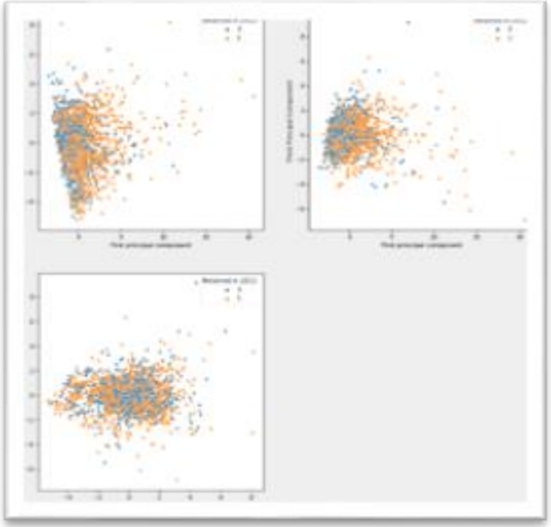
```
cat_nom=['Program.Code','Group.State','Travel.Type','Region','CRM.Segment','School.Type','Income.Level','SPR.Product.Type','SPR.I
```

```
cat_ord=['SchoolGradeTypeLow','SchoolGradeTypeHigh','GroupGradeTypeHigh','SchoolSizeIndicator']
```

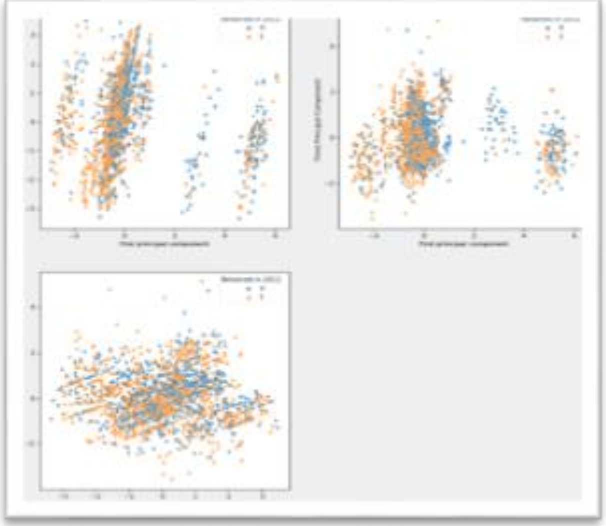
PCA Method

Principal Component Analysis (PCA) is a linear dimensionality reduction technique that can be utilized for extracting information from a high-dimensional space by projecting it into a lower-dimensional sub-space.

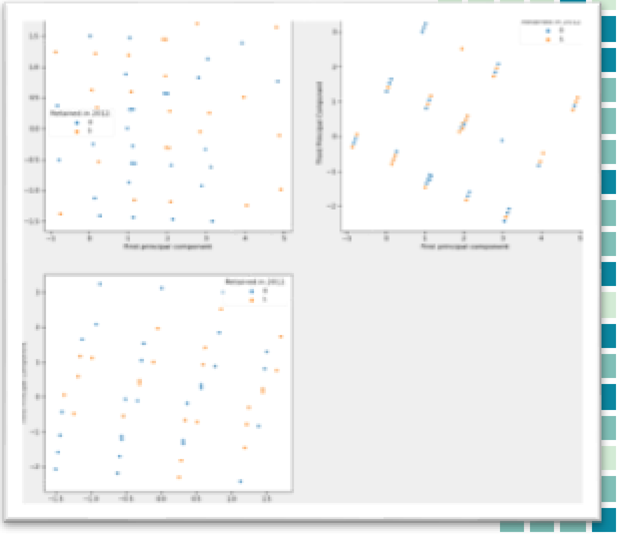
Numerical Values



Categorical Nominal



Categorical Ordinal



Random Forest
Set Accuracy & Importance

Numerical Values

```
Ran_Frst_num = RandomForestClassifier(criterion = 'gini')
Ran_Frst_num.fit(x_train_num, y_train_num)

print('test set accuracy: ', round(Ran_Frst_num.score(x_te
prediction_test_num = Ran_Frst_num.predict(x_test_num)
print (metrics.accuracy_score(y_test_num, prediction_test_
```

```
test set accuracy:  65.79
```

	column	Importance
7	Total.School.Enrollment	0.134040
1	Tuition	0.100596
10	Total.Pax	0.099381
11	SPR.Group.Revenue	0.098916
8	FPP	0.091825
2	FRP.Active	0.090517
4	FRP.Take.up.percent.	0.087160
9	EZ.Pay.Take.Up.Rate	0.081571
5	Cancelled.Pax	0.064434
3	FRP.Cancelled	0.053996
0	Days	0.035950
6	Total.Discount.Pax	0.031841
12	Num.of.Non_FPP.PAX	0.029772

Categorical Nominal

```
Ran_Frst_nom = RandomForestClassifier(criterion = 'gini')
Ran_Frst_nom.fit(x_train_nom, y_train_nom)

print('test set accuracy: ', round(Ran_Frst_nom.score(x_t
prediction_test_nom = Ran_Frst_nom.predict(x_test_nom)
print (metrics.accuracy_score(y_test_nom, prediction_test
```

```
test set accuracy:  68.12
```

	column	Importance
6	Income.Level	0.256318
1	Group.State	0.171280
8	SPR.New.Existing	0.115795
9	DepartureMonth	0.103796
4	CRM.Segment	0.090081
0	Program.Code	0.082118
3	Region	0.069634
5	School.Type	0.058236
2	Travel.Type	0.025623
10	MajorProgramCode	0.015835
7	SPR.Product.Type	0.011283

Categorical Ordinal

```
Ran_Frst_ord = RandomForestClassifier(criterion = 'gini')
Ran_Frst_ord.fit(x_train_ord, y_train_ord)

print('test set accuracy: ', round(Ran_Frst_ord.score(x_te
prediction_test_ord = Ran_Frst_ord.predict(x_test_ord)
print (metrics.accuracy_score(y_test_ord, prediction_test_
```

```
test set accuracy:  67.03
```

	column	Importance
3	SchoolSizeIndicator	0.397511
1	SchoolGradeTypeHigh	0.229594
0	SchoolGradeTypeLow	0.207750
2	GroupGradeTypeHigh	0.165145

Random Forest Performance Measures

Numerical Values

```
print(classification_report(y_test_num, prediction_test_num))
```

	precision	recall	f1-score	support
0	0.57	0.51	0.54	252
1	0.71	0.75	0.73	391
accuracy			0.66	643
macro avg	0.64	0.63	0.63	643
weighted avg	0.65	0.66	0.65	643

Categorical Ordinal

```
print(classification_report(y_test_ord, prediction_test_ord))
```

	precision	recall	f1-score	support
0	0.61	0.44	0.51	252
1	0.69	0.82	0.75	391
accuracy			0.67	643
macro avg	0.65	0.63	0.63	643
weighted avg	0.66	0.67	0.66	643

Categorical Nominal

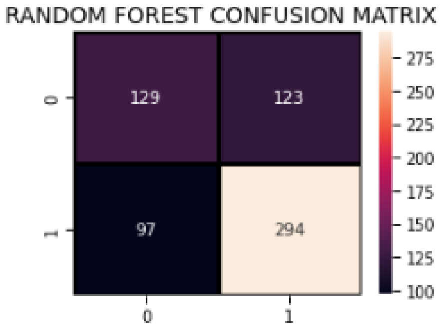
```
: print(classification_report(y_test_nom, prediction_test_nom))
```

	precision	recall	f1-score	support
0	0.61	0.54	0.57	252
1	0.72	0.77	0.75	391
accuracy			0.68	643
macro avg	0.66	0.66	0.66	643
weighted avg	0.68	0.68	0.68	643

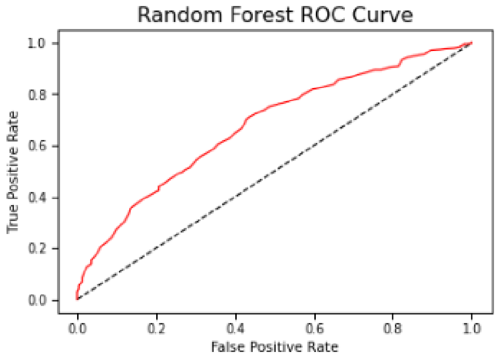
Random Forest – Numerical

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_num, prediction_test_num),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```



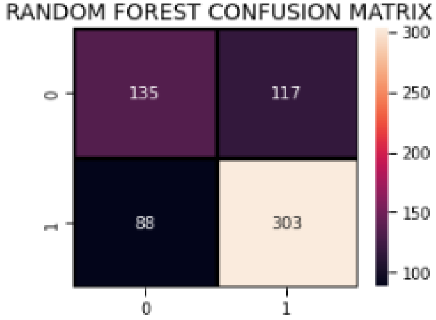

```
y_rfpred_prob = Ran_Frst_num.predict_proba(x_test_num)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test_num, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



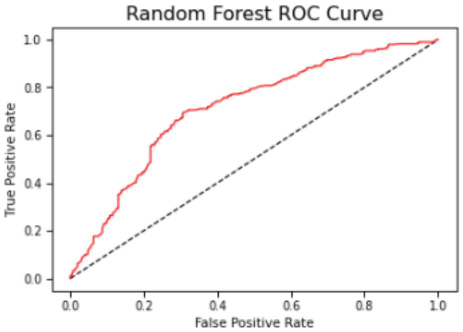
Random Forest – Categorical nominal

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_nom, prediction_test_nom),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```



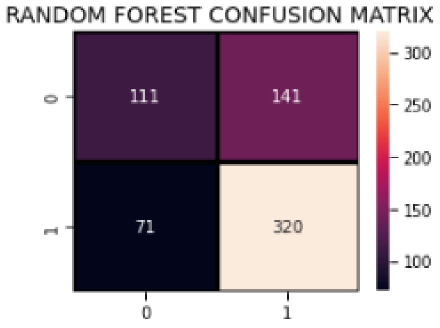

```
y_rfpred_prob = Ran_Frst_nom.predict_proba(x_test_nom)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test_nom, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



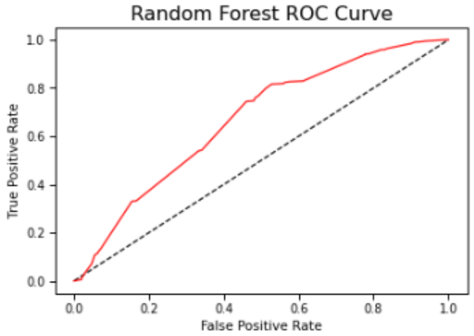
Random Forest – Categorical Ordinal

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_ord, prediction_test_ord),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```




```
y_rfpred_prob = Ran_Frst_ord.predict_proba(x_test_ord)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test_ord, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



Logistic Regression
Set Accuracy & Importance

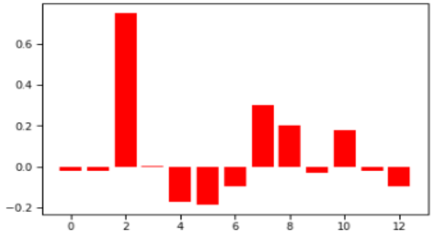
Numerical Values

```
lr_model_num = LogisticRegression()
lr_model_num.fit(x_train_num,y_train_num)
accuracy_lr = lr_model_num.score(x_test_num,y_test_num)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.6625194401244168
```

```
plt.bar([x for x in range(len(importance))], impor
plt.show()
```

```
Feature: 0, Score: -0.01946
Feature: 1, Score: -0.01988
Feature: 2, Score: 0.74870
Feature: 3, Score: 0.00295
Feature: 4, Score: -0.17115
Feature: 5, Score: -0.18250
Feature: 6, Score: -0.09463
Feature: 7, Score: 0.30149
Feature: 8, Score: 0.20295
Feature: 9, Score: -0.02853
Feature: 10, Score: 0.17827
Feature: 11, Score: -0.01988
Feature: 12, Score: -0.09463
```



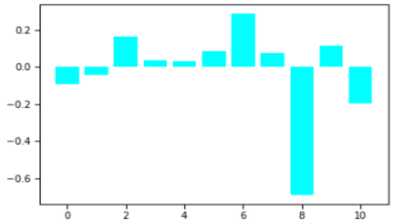
Categorical Nominal

```
lr_model_nom = LogisticRegression()
lr_model_nom.fit(x_train_nom,y_train_nom)
accuracy_lr = lr_model_nom.score(x_test_nom,y_test_nom)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.7200622083981337
```

```
importance = lr_model_nom.coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], impor
plt.show()
```

```
Feature: 0, Score: -0.09206
Feature: 1, Score: -0.04159
Feature: 2, Score: 0.16464
Feature: 3, Score: 0.03596
Feature: 4, Score: 0.03006
Feature: 5, Score: 0.08643
Feature: 6, Score: 0.28856
Feature: 7, Score: 0.07723
Feature: 8, Score: -0.68954
Feature: 9, Score: 0.11609
Feature: 10, Score: -0.19579
```



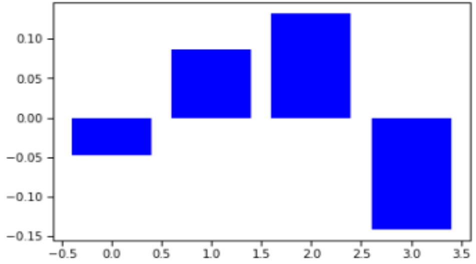
Categorical Ordinal

```
lr_model_ord = LogisticRegression()
lr_model_ord.fit(x_train_ord,y_train_ord)
accuracy_lr = lr_model_ord.score(x_test_ord,y_test_ord)
print("Logistic Regression accuracy is :",accuracy_lr)
```

```
Logistic Regression accuracy is : 0.6018662519440124
```

```
importance = lr_model_ord.coef_[0]
# summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], import
plt.show()
```

```
Feature: 0, Score: -0.04817
Feature: 1, Score: 0.08621
Feature: 2, Score: 0.13236
Feature: 3, Score: -0.14117
```



Logistic Regression Performance Measures

Numerical Values

```
lr_pred_num= lr_model_num.predict(x_test_num)
report = classification_report(y_test_num,lr_pred_num)
print(report)
```

	precision	recall	f1-score	support
0	0.60	0.43	0.50	252
1	0.69	0.81	0.75	391
accuracy			0.66	643
macro avg	0.64	0.62	0.62	643
weighted avg	0.65	0.66	0.65	643

Categorical Ordinal

```
lr_pred_ord= lr_model_ord.predict(x_test_ord)
report = classification_report(y_test_ord,lr_pred_ord)
print(report)
```

	precision	recall	f1-score	support
0	0.44	0.06	0.10	252
1	0.61	0.95	0.74	391
accuracy			0.60	643
macro avg	0.53	0.51	0.42	643
weighted avg	0.54	0.60	0.49	643

Categorical Nominal

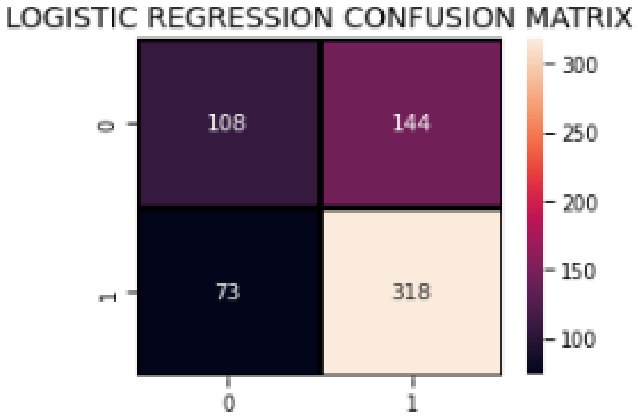
```
lr_pred_nom= lr_model_nom.predict(x_test_nom)
report = classification_report(y_test_nom,lr_pred_nom)
print(report)
```

	precision	recall	f1-score	support
0	0.67	0.55	0.61	252
1	0.74	0.83	0.78	391
accuracy			0.72	643
macro avg	0.71	0.69	0.69	643
weighted avg	0.72	0.72	0.71	643

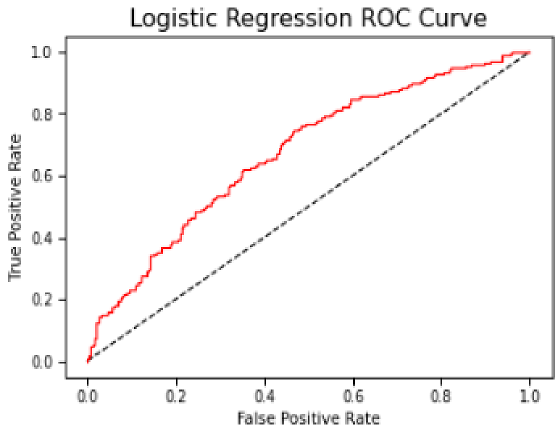
Logistic Regression – Numerical

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_num, l
                annot=True,fmt = "d",linec

plt.title("LOGISTIC REGRESSION CONFUSION M
plt.show()
```



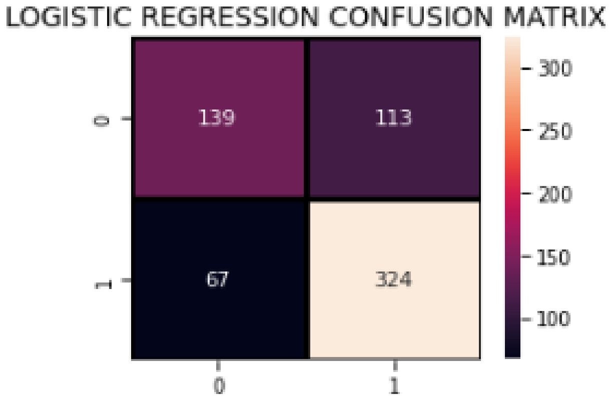

```
y_pred_prob = lr_model_num.predict_proba(x_test_num)[:,1]
fpr, tpr, thresholds = roc_curve(y_test_num, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```



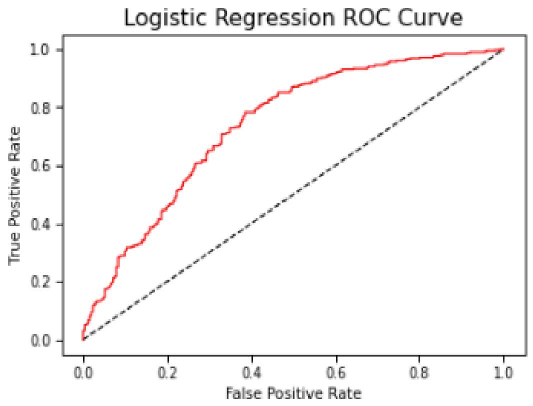
Logistic Regression – Categorical nominal

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_nom,
            annot=True,fmt = "d",line

plt.title("LOGISTIC REGRESSION CONFUSION
plt.show()
```

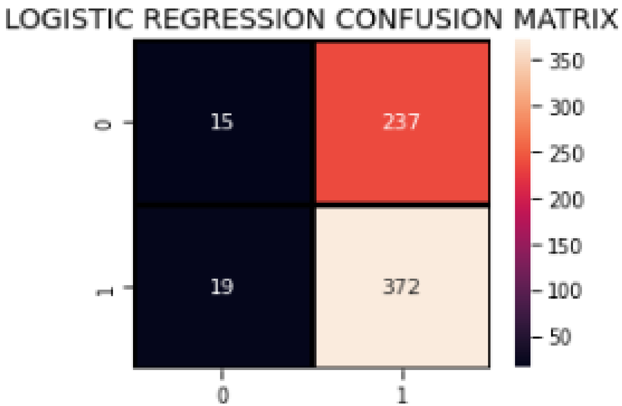



```
y_pred_prob = lr_model_nom.predict_proba(x_test_nom)[:,1]
fpr, tpr, thresholds = roc_curve(y_test_nom, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```

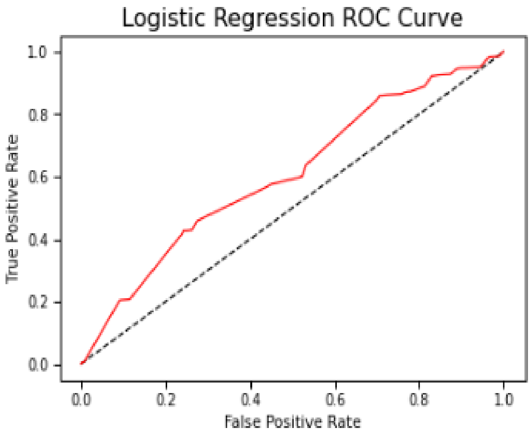


Logistic Regression – Categorical Ordinal

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_ord,
                annot=True,fmt = "d",line
plt.title("LOGISTIC REGRESSION CONFUSION
plt.show()
```




```
y_pred_prob = lr_model_ord.predict_proba(x_test_ord)[:,1]
fpr, tpr, thresholds = roc_curve(y_test_ord, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```



Decision Tree Classifier

Numerical Values

```
dt_model_num = DecisionTreeClassifier()
dt_model_num.fit(x_train_num,y_train_num)
predictdt_y = dt_model_num.predict(x_test_num)
accuracy_dt = dt_model_num.score(x_test_num,y_test_num)
print("Decision Tree accuracy is :",accuracy_dt)

Decision Tree accuracy is : 0.5972006220839814
```

	column	Importance
7	Total.School.Enrollment	0.141710
8	FPP	0.140401
9	EZ.Pay.Take.Up.Rate	0.112105
4	FRP.Take.up.percent.	0.092000
11	SPR.Group.Revenue	0.088798
1	Tuition	0.086640
5	Cancelled.Pax	0.080409
2	FRP.Active	0.076043
10	Total.Pax	0.051245
3	FRP.Cancelled	0.040121
0	Days	0.035062
6	Total.Discount.Pax	0.033305
12	Num.of.Non_FPP.PAX	0.022163

Categorical Nominal

```
dt_model_nom = DecisionTreeClassifier()
dt_model_nom.fit(x_train_nom,y_train_nom)
predictdt_y = dt_model_nom.predict(x_test_nom)
accuracy_dt = dt_model_nom.score(x_test_nom,y_test_nom)
print("Decision Tree accuracy is :",accuracy_dt)

Decision Tree accuracy is : 0.6314152410575428
```

	column	Importance
6	Income.Level	0.245462
1	Group.State	0.185536
8	SPR.New.Existing	0.135755
4	CRM.Segment	0.099613
9	DepartureMonth	0.091706
3	Region	0.072737
5	School.Type	0.066883
0	Program.Code	0.064012
2	Travel.Type	0.026414
7	SPR.Product.Type	0.008015
10	MajorProgramCode	0.003867

Categorical Ordinal

```
dt_model_ord = DecisionTreeClassifier()
dt_model_ord.fit(x_train_ord,y_train_ord)
predictdt_y = dt_model_ord.predict(x_test_ord)
accuracy_dt = dt_model_ord.score(x_test_ord,y_test_nom
print("Decision Tree accuracy is :",accuracy_dt)

Decision Tree accuracy is : 0.6702954898911353
```

	column	Importance
3	SchoolSizeIndicator	0.361570
0	SchoolGradeTypeLow	0.328902
2	GroupGradeTypeHigh	0.164866
1	SchoolGradeTypeHigh	0.144662

KNN Classification

Numerical Values

```
knn_model_num = KNeighborsClassifier(n_neighbors = 3)
knn_model_num.fit(x_train_num,y_train_num)
predicted_y = knn_model_num.predict(x_test_num)
accuracy_knn = knn_model_num.score(x_test_num,y_test_num)
print("KNN accuracy:",accuracy_knn)
```

```
KNN accuracy: 0.6516329704510109
```

Categorical Nominal

```
knn_model_nom = KNeighborsClassifier(n_neighbors = 11)
knn_model_nom.fit(x_train_nom,y_train_nom)
predicted_y = knn_model_nom.predict(x_test_nom)
accuracy_knn = knn_model_nom.score(x_test_nom,y_test_nom)
print("KNN accuracy:",accuracy_knn)
```

```
KNN accuracy: 0.7200622083981337
```

Categorical Ordinal

```
knn_model_ord = KNeighborsClassifier(n_neighbors = 11)
knn_model_ord.fit(x_train_ord,y_train_ord)
predicted_y = knn_model_ord.predict(x_test_ord)
accuracy_knn = knn_model_ord.score(x_test_ord,y_test_ord)
print("KNN accuracy:",accuracy_knn)
```

```
KNN accuracy: 0.6671850699844479
```

Selected Columns Based On Analysis

By considering different methods of classification and prediction with their amount of accuracy, we tried to filter the most important columns which are effective on our target value.

Numerical

- Total school enrollment
- FPP
- Total Pax
- SPR Group Revenue
- Income Level

Nominal

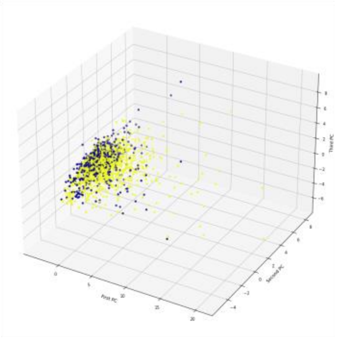
- SPR New Existing
- Group State
- Is Non Annual
- Single Grade Trip Flag
- Departure Month

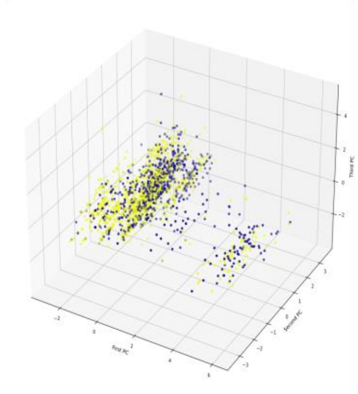
Ordinal

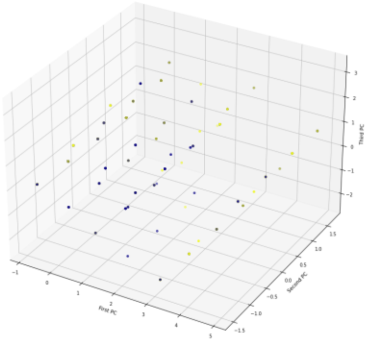
- SchoolGradeTypeHigh

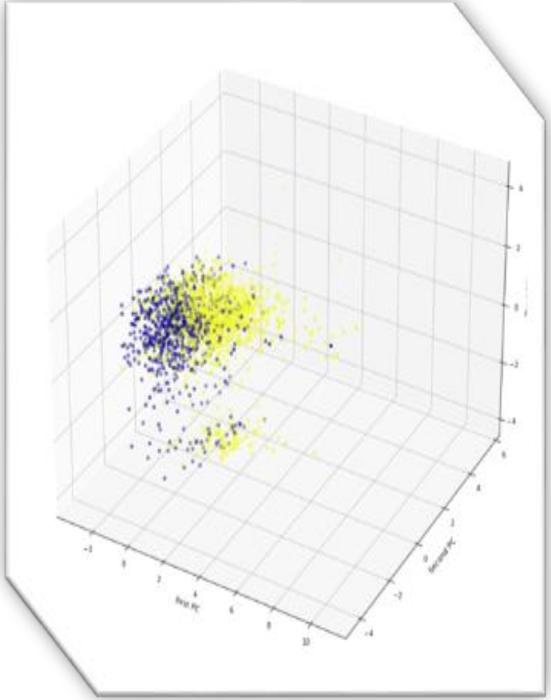
* These columns are selected by analyzing the pair plot and the correlations

PCA Method









Random Forest Performance Measures

```
Ran_Frst_slt = RandomForestClassifier(criterion = 'gini')
Ran_Frst_slt.fit(x_train_slt, y_train_slt)

print('test set accuracy: ', round(Ran_Frst_slt.score(x_test_slt, y_test_slt)*100, 2))
prediction_test_slt = Ran_Frst_slt.predict(x_test_slt)
print (metrics.accuracy_score(y_test_slt, prediction_test_slt))
```

```
test set accuracy:  83.83
```

```
print(classification_report(y_test_slt, prediction_test_slt))
```

```
              precision    recall  f1-score   support

           0       0.80      0.79      0.79       252
           1       0.87      0.87      0.87       391

    accuracy                           0.84       643
   macro avg       0.83      0.83      0.83       643
weighted avg       0.84      0.84      0.84       643
```

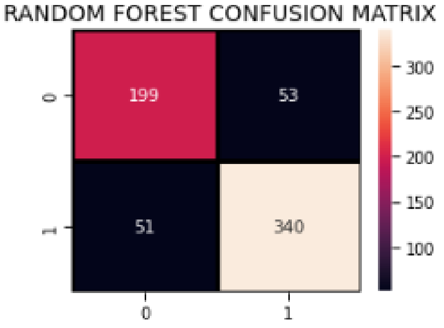
```
df_result = pd.DataFrame()
df_result['column'] = x_train_slt.columns
df_result['Importance'] = Ran_Frst_slt.feature_importances_
df_result.sort_values('Importance',ascending=False)
```

	column	Importance
3	SPR.Group.Revenue	0.130503
0	Total.School.Enrollment	0.130295
8	SingleGradeTripFlag	0.128867
1	FPP	0.117846
7	Is.Non.Annual.	0.110747
2	Total.Pax	0.105142
4	Income.Level	0.078639
5	SPR.New.Existing	0.077950
6	Group.State	0.063173
9	DepartureMonth	0.037317
10	SchoolGradeTypeHigh	0.019520

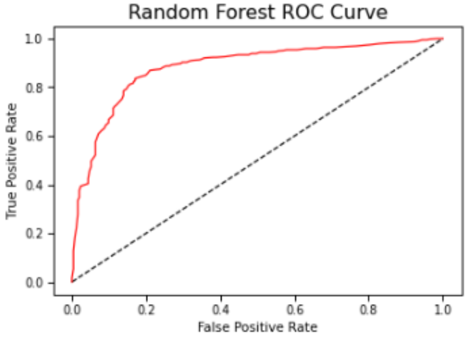
Random Forest

```python
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_slt, prediction_test_slt),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title(" RANDOM FOREST CONFUSION MATRIX",fontsize=14)
plt.show()
```




```python
y_rfpred_prob = Ran_Frst_slt.predict_proba(x_test_slt)[:,1]
fpr_rf, tpr_rf, thresholds = roc_curve(y_test_slt, y_rfpred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr_rf, tpr_rf, label='Random Forest',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve',fontsize=16)
plt.show();
```



Decision Tree Performance Measures

```
dt_model_slt = DecisionTreeClassifier()
dt_model_slt.fit(x_train_slt,y_train_slt)
predictdt_y = dt_model_slt.predict(x_test_slt)
accuracy_dt = dt_model_slt.score(x_test_slt,y_test_slt)
print("Decision Tree accuracy is :",accuracy_dt)
```

```
Decision Tree accuracy is : 0.7045101088646968
```

```
print(classification_report(y_test_slt, predictdt_y))
```

```
              precision    recall  f1-score   support

           0       0.62      0.65      0.63       252
           1       0.77      0.74      0.75       391

    accuracy                           0.70       643
   macro avg       0.69      0.70      0.69       643
weighted avg       0.71      0.70      0.71       643
```

```
df_result = pd.DataFrame()
df_result['column'] = x_train_slt.columns
df_result['Importance'] = dt_model_slt.feature_importances_
df_result.sort_values('Importance',ascending=False)
```

	column	Importance
8	SingleGradeTripFlag	0.220323
0	Total.School.Enrollment	0.169439
3	SPR.Group.Revenue	0.129071
2	Total.Pax	0.107700
7	Is.Non.Annual.	0.094287
4	Income.Level	0.080431
1	FPP	0.068919
9	DepartureMonth	0.045399
6	Group.State	0.043543
5	SPR.New.Existing	0.037825
10	SchoolGradeTypeHigh	0.003063

KNN
Performance Measures

```
knn_model_slt = KNeighborsClassifier(n_neighbors = 11)
knn_model_slt.fit(x_train_slt,y_train_slt)
predicted_y = knn_model_slt.predict(x_test_slt)
accuracy_knn = knn_model_slt.score(x_test_slt,y_test_slt)
print("KNN accuracy:",accuracy_knn)
```

```
KNN accuracy: 0.8258164852255054
```

Logistic Regression Performance Measures

```
lr_model_slt = LogisticRegression()
lr_model_slt.fit(x_train_slt,y_train_slt)
accuracy_lr = lr_model_slt.score(x_test_slt,y_test_slt)
print("Logistic Regression accuracy is :",accuracy_lr)
```

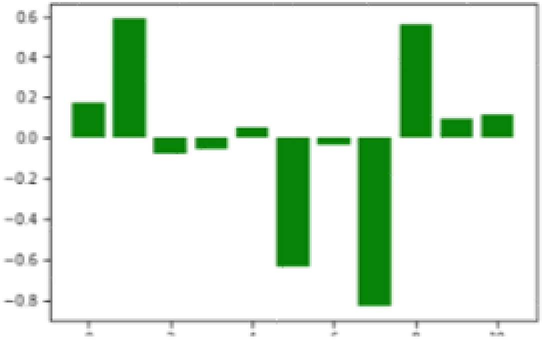
```
Logistic Regression accuracy is : 0.8304821150855366
```

```
lr_pred_slt= lr_model_slt.predict(x_test_slt)
report = classification_report(y_test_slt,lr_pred_slt)
print(report)
```

	precision	recall	f1-score	support
0	0.81	0.73	0.77	252
1	0.84	0.89	0.86	391
accuracy			0.83	643
macro avg	0.83	0.81	0.82	643
weighted avg	0.83	0.83	0.83	643

```
importance = lr_model_slt.coef_[0]
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
plt.bar([x for x in range(len(importance))], impor
plt.show()
```

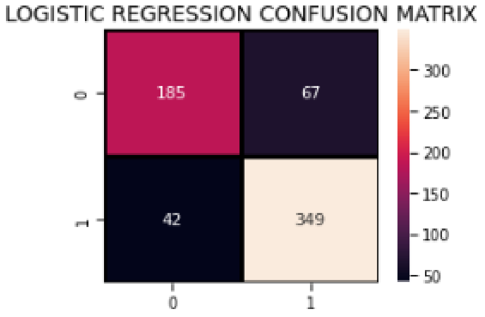
```
Feature: 0, Score: 0.17513
Feature: 1, Score: 0.58996
Feature: 2, Score: -0.07515
Feature: 3, Score: -0.05454
Feature: 4, Score: 0.04857
Feature: 5, Score: -0.63537
Feature: 6, Score: -0.03346
Feature: 7, Score: -0.82938
Feature: 8, Score: 0.55794
Feature: 9, Score: 0.09406
Feature: 10, Score: 0.11328
```



Logistic Regression

```
plt.figure(figsize=(4,3))
sns.heatmap(confusion_matrix(y_test_slt, lr_pred_slt),
                annot=True,fmt = "d",linecolor="k",linewidths=3)

plt.title("LOGISTIC REGRESSION CONFUSION MATRIX",fontsize=14)
plt.show()
```




```
model1=LogisticRegression().fit(x_train_slt,y_train_slt)
y_pred_prob = lr_model_slt.predict_proba(x_test_slt)[:,1]
fpr, tpr, thresholds = roc_curve(y_test_slt, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--' )
plt.plot(fpr, tpr, label='Logistic Regression',color = "r")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Logistic Regression ROC Curve',fontsize=16)
plt.show();
```

SVC
Performance Measures

```
svc_model = SVC(random_state = 1)
svc_model.fit(x_train_slt,y_train_slt)
predict_y = svc_model.predict(x_test_slt)
accuracy_svc = svc_model.score(x_test_slt,y_test_slt)
print("SVM accuracy is :",accuracy_svc)
```

```
SVM accuracy is : 0.8273716951788491
```

WHAT IS SVC?

A support vector machine is a supervised machine learning algorithm that can be used for both classification and regression tasks. The Support vector machine classifier works by finding the hyperplane that maximizes the margin between the two classes.

```
print(classification_report(y_test_slt, predict_y))
```

	precision	recall	f1-score	support
0	0.83	0.70	0.76	252
1	0.83	0.91	0.86	391
accuracy			0.83	643
macro avg	0.83	0.81	0.81	643
weighted avg	0.83	0.83	0.82	643

6. *Forecasting*

83.83%

Based on below comparison Random Forest method is showing the highest accuracy thus our forecasting is based on it.

Method	Accuracy
Random Forest	83.83%
Decision Tree	70.45%
KNN	82.58%
Logistic Regression	83.04%
SVC	82.73%

Forecasting

```
prediction=pd.DataFrame(Ran_Frst_slt.predict(x_train_slt))
prediction
```

	0
0	0
1	1
2	1
3	0
4	0
...	...
1493	0
1494	0
1495	0
1496	1
1497	0

1498 rows × 1 columns