In [159…

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import hstack
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

PreProcessing

Load Data after EDA

In [160…

```
data = pd.read_csv('data_after_eda.csv')
data.head()
```

Out[160]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductR
0	0	0.0	0	0.0	
1	0	0.0	0	0.0	
2	0	0.0	0	0.0	
3	0	0.0	0	0.0	
4	0	0.0	0	0.0	

In [161…

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12330 entries, 0 to 12329
Data columns (total 13 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Administrative           12330 non-null  int64
 1   Administrative_Duration  12330 non-null  float64
 2   Informational            12330 non-null  int64
 3   Informational_Duration   12330 non-null  float64
 4   ProductRelated           12330 non-null  float64
 5   ProductRelated_Duration  12330 non-null  float64
 6   BounceRates              12330 non-null  float64
 7   ExitRates                12330 non-null  float64
 8   PageValues               12330 non-null  float64
 9   SpecialDay               12330 non-null  float64
 10  Month                    12330 non-null  object
 11  VisitorType              12330 non-null  object
 12  target                   12330 non-null  int64
dtypes: float64(8), int64(3), object(2)
memory usage: 1.2+ MB
```

We are observing Categorical and Numerical features in our data. We will normalize the numerical features and perform onhot encoding on categorical features

Train Test Data Split

In [162…

```
#Devide data as X and y
y = data['target'].values
X = data.drop(['target'],axis=1)

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.33,stratif

print("Train=",X_train.shape, y_train.shape)
print("Test=",X_test.shape, y_test.shape)
```

```
Train= (8261, 12) (8261,)
Test= (4069, 12) (4069,)
```

Handling Categorical features

In [163…

```
#Collecting feature names
top_feature_names =[]


vectorizer = CountVectorizer()

#Converting school state
vectorizer.fit(X_train['VisitorType'].values) # fit has to happen only on t
top_feature_names.extend(vectorizer.get_feature_names())

# we use the fitted CountVectorizer to convert the text to vector
X_train_visitor_ohe = vectorizer.transform(X_train['VisitorType'].values)
X_test_visitor_ohe = vectorizer.transform(X_test['VisitorType'].values)

#Converting school state
vectorizer.fit(X_train['Month'].values) # fit has to happen only on train d
top_feature_names.extend(vectorizer.get_feature_names())

# we use the fitted CountVectorizer to convert the text to vector
X_train_month_ohe = vectorizer.transform(X_train['Month'].values)
X_test_month_ohe = vectorizer.transform(X_test['Month'].values)
```

Handling Numerical features

In [164…

```
normalizer = Normalizer()
def transform_numerical_features(feature):
    X_train[feature] = normalizer.fit_transform(X_train[feature].values.res
    X_test[feature] =  normalizer.fit_transform(X_test[feature].values.resh
```

In [165…

```
for column in X_train.select_dtypes(['int64','float64']):
    transform_numerical_features(column)
    top_feature_names.append(column)
```

Drop Month and VisitorType from Train and Test

In [166…

```
X_train.drop(['Month','VisitorType'],axis=1,inplace=True)
X_test.drop(['Month','VisitorType'],axis=1,inplace=True)
```

In [167…

```
X_train.head()
X_test.head()
```

Out[167]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	Produ
12021	0.000000	0.000000	0.0	0.0	
8710	0.000000	0.000000	0.0	0.0	
8749	0.003952	0.018872	0.0	0.0	
11731	0.000000	0.000000	0.0	0.0	
6585	0.011856	0.006822	0.0	0.0	

In [168…

```
print("Total No.Of features for Feature selection=",len(top_feature_names))
```

```
Total No.Of features for Feature selection= 23
```

Merge pre-processed data

In [169…

```
X_tr = hstack((X_train,X_train_visitor_ohe,X_train_month_ohe ))
X_te = hstack((X_test,X_test_visitor_ohe,X_test_month_ohe ))
```

In [170…

```
print(X_tr.shape)
print(X_te.shape)
```

```
(8261, 23)
(4069, 23)
```

Modeling

Modeling using Decision Tree

In [171…

```
dtClassifier = DecisionTreeClassifier(class_weight={1:8,0:2})
#review range
parameters ={'max_depth':[1, 5, 10, 50],'min_samples_split':[5, 10, 100, 50
clf = GridSearchCV(dtClassifier,parameters,cv=3,scoring='roc_auc',return_tr
clf.fit(X_tr,y_train)
```

Out[171]:

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(class_weight={0: 2, 1:
8}),
             param_grid={'max_depth': [1, 5, 10, 50],
                         'min_samples_split': [5, 10, 100, 500]},
             return_train_score=True, scoring='roc_auc')
```

In [172…

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results.sort_values('rank_test_score')[0:3]
```

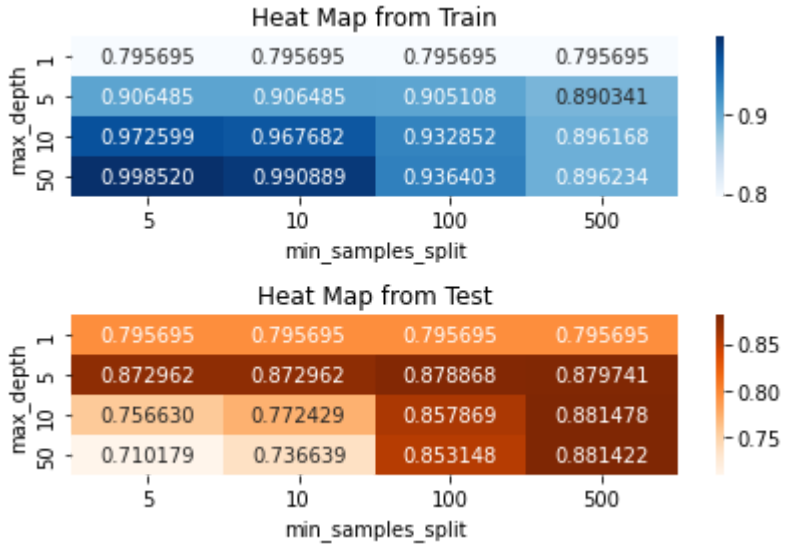
Out[172]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_depth	par
11	0.010266	0.000290	0.000901	0.000005	10	
15	0.010298	0.000262	0.000957	0.000099	50	
7	0.008215	0.000119	0.001022	0.000049	5	

HeatMap for HyperParameter Analysis Visualization

In [173…

```
##Heatmap
#https://stackoverflow.com/questions/39041865/three-variables-as-heatmap
import seaborn as sns
import matplotlib.pyplot as plt

max_depth=results['param_max_depth']
min_samples_split=results['param_min_samples_split']
train_score = results['mean_train_score']
test_score = results['mean_test_score']

fig, ax= plt.subplots(nrows=2)
#https://stackoverflow.com/questions/41659188/how-to-adjust-subplot-size-in
plt.rcParams['figure.figsize']=(7,7)

train_roc_auc = pd.DataFrame({'max_depth':max_depth,'min_samples_split':min_
test_roc_auc = pd.DataFrame({'max_depth':max_depth,'min_samples_split':min_

train_roc_auc = train_roc_auc.pivot('max_depth', 'min_samples_split', 'trai
ax[0] = sns.heatmap(train_roc_auc,annot=True, fmt="f",ax=ax[0],cmap='Blues'

# labels, title
ax[0].set_xlabel('min_samples_split');
ax[0].set_ylabel('max_depth');
ax[0].set_title('Heat Map from Train');


test_roc_auc = test_roc_auc.pivot('max_depth', 'min_samples_split', 'test_s
ax[1] = sns.heatmap(test_roc_auc,annot=True, fmt="f",ax=ax[1],cmap='Oranges

# labels, title
ax[1].set_xlabel('min_samples_split');
ax[1].set_ylabel('max_depth');
ax[1].set_title('Heat Map from Test');

fig.tight_layout()
```



Fetch Best Params for Modeling

In [174…

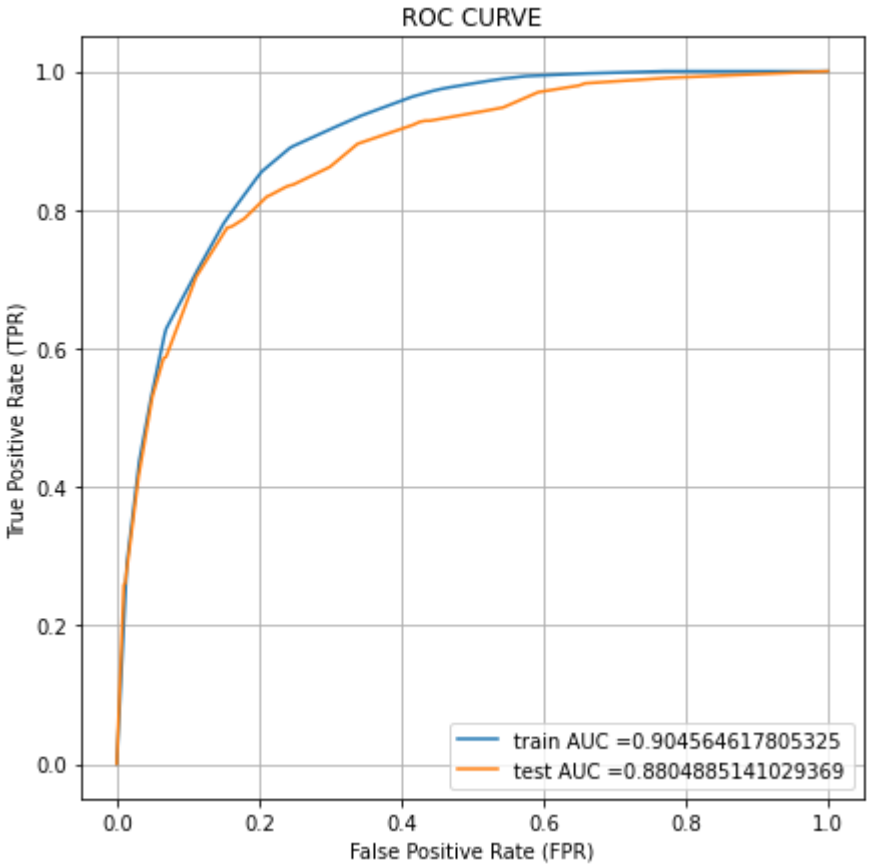
```
##Fetch best permaters
max_depth= clf.best_params_['max_depth']
min_samples_split= clf.best_params_['min_samples_split']
print('Best depth=',max_depth,' Best min samples=',min_samples_split)
```

```
Best depth= 10  Best min samples= 500
```

Decision Tree Modeling using Best Params

In [175…

```
#Pass actual alpha and compute predictions
#https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.Mult
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_cur

from sklearn.metrics import roc_curve, auc

dTclf = DecisionTreeClassifier(class_weight={1:8,0:2},max_depth=max_depth,m
dTclf.fit(X_tr,y_train)

#https://discuss.analyticsvidhya.com/t/what-is-the-difference-between-predi
y_train_pred = dTclf.predict_proba(X_tr)[:,1]
y_test_pred = dTclf.predict_proba(X_te)[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)

#Plot roc curve, AUC cureve
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)
plt.legend()
plt.xlabel("False Positive Rate (FPR)")
plt.ylabel("True Positive Rate (TPR)")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



In [176…

```
best_auc_dt = str(auc(test_fpr, test_tpr))
```

Confusion Matrix

In [177…

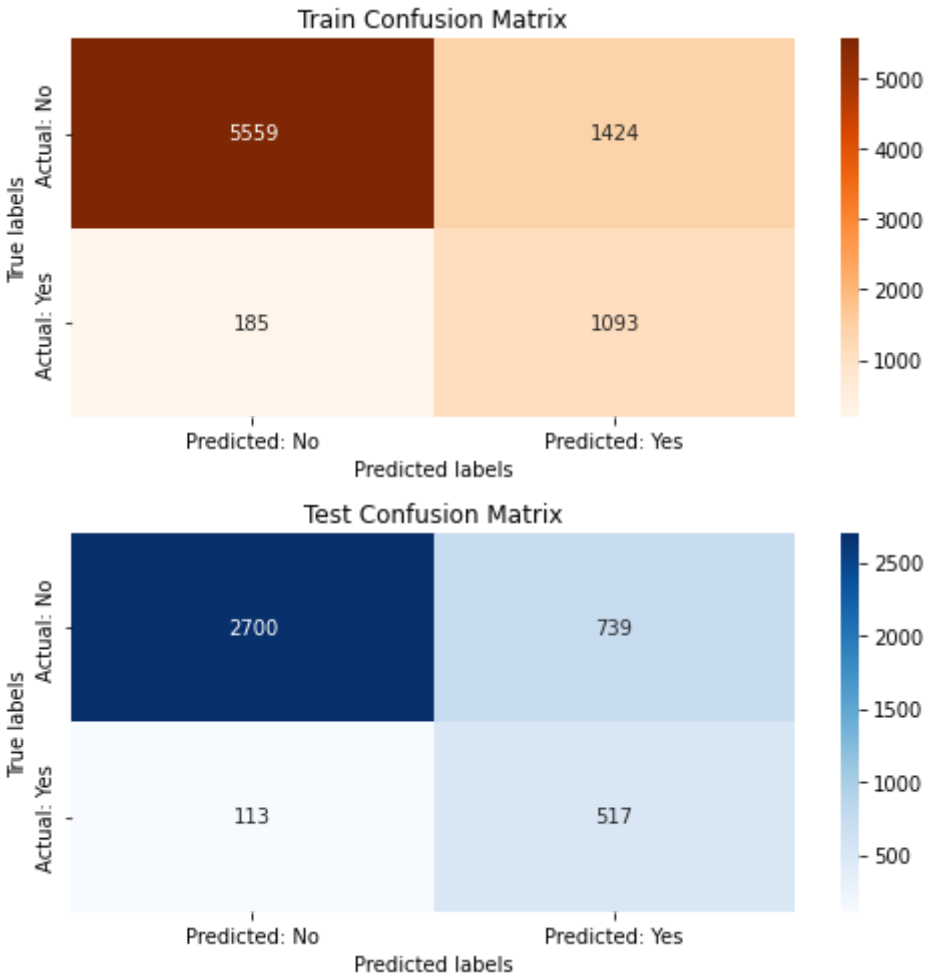
```
import numpy as np
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for thresh
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


print("="*50)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
y_test_preicted = predict_with_best_t(y_test_pred, best_t)
train_conf = confusion_matrix(y_train, predict_with_best_t(y_train_pred, be
test_conf = confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_
```

```
==================================================
the maximum value of tpr*(1-fpr) 0.6808382396147855 for threshold 0.49
```

In [178…

```
#https://stackoverflow.com/questions/61748441/how-to-fix-the-values-display
```

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax= plt.subplots(nrows=2)
#https://stackoverflow.com/questions/41659188/how-to-adjust-subplot-size-in
plt.rcParams['figure.figsize']=(7,7)

sns.heatmap(train_conf, annot=True,fmt="d",cmap='Oranges',ax=ax[0]);
# labels, title and ticks
ax[0].set_xlabel('Predicted labels');
ax[0].set_ylabel('True labels');
ax[0].set_ylim(2.0, 0)
ax[0].set_title('Train Confusion Matrix');
ax[0].xaxis.set_ticklabels(['Predicted: No','Predicted: Yes']);
ax[0].yaxis.set_ticklabels(['Actual: No','Actual: Yes']);

sns.heatmap(test_conf, annot=True,fmt="d",cmap='Blues',ax=ax[1]);
# labels, title and ticks
ax[1].set_xlabel('Predicted labels');ax[1].set_ylabel('True labels');
ax[1].set_ylim(2.0, 0)
ax[1].set_title('Test Confusion Matrix');
ax[1].xaxis.set_ticklabels(['Predicted: No','Predicted: Yes']);
ax[1].yaxis.set_ticklabels(['Actual: No','Actual: Yes']);


fig.tight_layout()
```



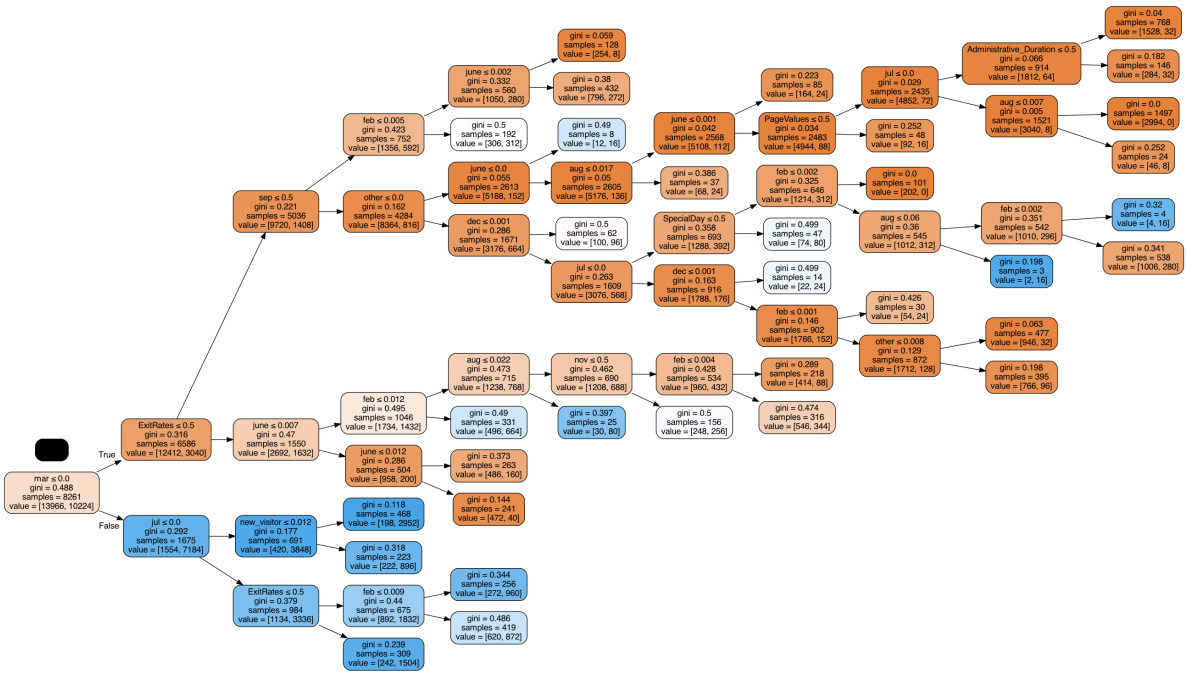
Visualize DecisionTree

In [179…

```
#https://stackoverflow.com/questions/61901365/modulenotfounderror-no-module
from six import StringIO
```

```
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(dTclf, out_file=dot_data, filled=True, rounded=True, specia
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[179]:



Combine test data with predicted and filter False Positive DataPoints

In [180…

```
test_data = X_test
test_data['y_actual'] = list(y_test)
test_data['y_pred'] = y_test_preicted

test_data.head(5)
```

Out[180]:

	Administrative	Administrative_Duration	Informational	Informational_Duration	Prodı
12021	0.000000	0.000000	0.0	0.0	
8710	0.000000	0.000000	0.0	0.0	
8749	0.003952	0.018872	0.0	0.0	
11731	0.000000	0.000000	0.0	0.0	
6585	0.011856	0.006822	0.0	0.0	

Fetch Important Features from DT

In [181…

```
#https://stackoverflow.com/questions/36967666/transform-scipy-sparse-csr-to
X_train_imp_features = pd.DataFrame(None)
X_test_imp_features = pd.DataFrame(None)
X_train_imp_features = pd.DataFrame.sparse.from_spmatrix(X_tr, columns=top_
X_test_imp_features = pd.DataFrame.sparse.from_spmatrix(X_te, columns=top_f

print(X_train_imp_features.shape)
print(X_test_imp_features.shape)
```

```
(8261, 23)
(4069, 23)
```

In [182…

```
#https://stackoverflow.com/questions/4588628/find-indices-of-elements-equal
#https://stackoverflow.com/questions/11285613/selecting-multiple-columns-in
print("Totsl No.of Existing Features=",len(dTclf.feature_importances_))
important_features = dTclf.feature_importances_
non_zero_imp_features = np.nonzero(important_features)[0]
print("Total No.Of Important Features=",len(non_zero_imp_features))
X_train_data_with_important_features = X_train_imp_features.iloc[:, non_zer
X_test_data_with_important_features = X_test_imp_features.iloc[:, non_zero_

print(X_train_data_with_important_features.shape)
print(X_test_data_with_important_features.shape)
```

```
Totsl No.of Existing Features= 23
Total No.Of Important Features= 14
(8261, 14)
(4069, 14)
```

In [183…

```
X_train_data_with_important_features.head()
```

Out[183]:

	new_visitor	other	aug	dec	feb	jul	june	mar	nov	sep
0	0.013455	0.004301	0.0	0.008169	0.004094	0.000000	0.000355	0.0	0.0	1.0
1	0.000000	0.000000	0.0	0.003812	0.002931	0.008922	0.014197	0.0	0.0	1.0
2	0.000000	0.000000	0.0	0.008169	0.005479	0.005205	0.007986	0.0	0.0	1.0
3	0.000000	0.000000	0.0	0.000545	0.000486	0.000000	0.017036	0.0	1.0	0.0
4	0.000000	0.000000	0.0	0.001634	0.002704	0.027758	0.028393	0.0	0.0	1.0

Modeling using Logistic Regression

We will use important features extracted and use here for LR

Conver Data to Sparse matrix

In [184…

```
from scipy.sparse import csr_matrix
#https://stackoverflow.com/questions/20459536/convert-pandas-dataframe-to-s
X_train_data_imp_features = csr_matrix(X_train_data_with_important_features
X_test_data_imp_features= csr_matrix(X_test_data_with_important_features.va
print(X_train_data_imp_features.shape)
print(X_test_data_imp_features.shape)
```

```
(8261, 14)
(4069, 14)
```

Find best Hyperparams using Cross validation

In [185…

```
#Impliment find best Hyperparams using Cross validation
from sklearn.model_selection import RandomizedSearchCV
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()

parameters = {'C':  np.logspace(-10, 2, num=40, endpoint=True, base=10.0, d
randomSearchclf = RandomizedSearchCV(clf,parameters,cv=3,return_train_score
```

```
randomSearchclf.fit(X_train_data_imp_features,y_train)

svc_result = pd.DataFrame.from_dict(randomSearchclf.cv_results_)
best_hyper_param = randomSearchclf.best_params_['C'];
print('Best Hyperparameters=',best_hyper_param)
```

```
Best Hyperparameters= 100.0
```

In [186…

```
svc_result.sort_values('rank_test_score')[0:3]
```

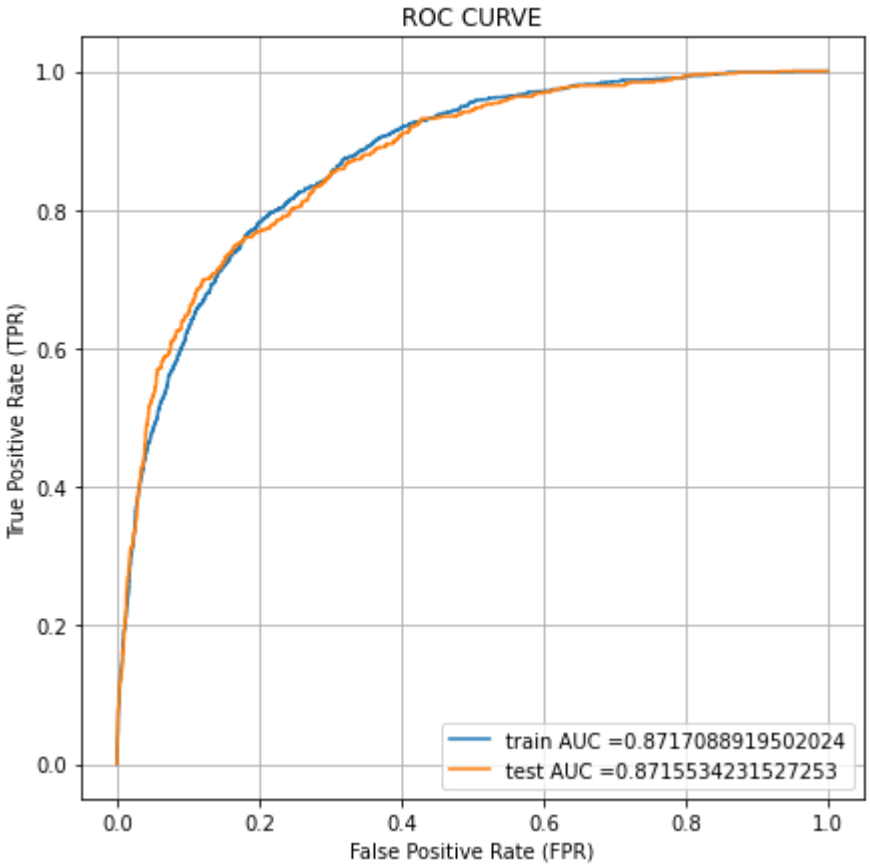
Out[186]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_C	
9	0.026603	0.000514	0.000752	0.000023	100.0	{
2	0.017300	0.001406	0.000770	0.000036	2.894266	2.89426612
8	0.005669	0.000242	0.000740	0.000008	0.020309	0.0203091762

Modeling Using Logisitc Regression

In [187…

```
best_model = LogisticRegression(C=best_hyper_param)
best_model.fit(X_train_data_imp_features,y_train)

#https://discuss.analyticsvidhya.com/t/what-is-the-difference-between-predi
y_train_pred = best_model.predict_proba(X_train_data_imp_features)[:,1]
y_test_pred = best_model.predict_proba(X_test_data_imp_features)[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)

#Plot roc curve, AUC cureve
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)
plt.legend()
plt.xlabel("False Positive Rate (FPR)")
plt.ylabel("True Positive Rate (TPR)")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



In [188…

```
best_auc_lr = str(auc(test_fpr, test_tpr))
```

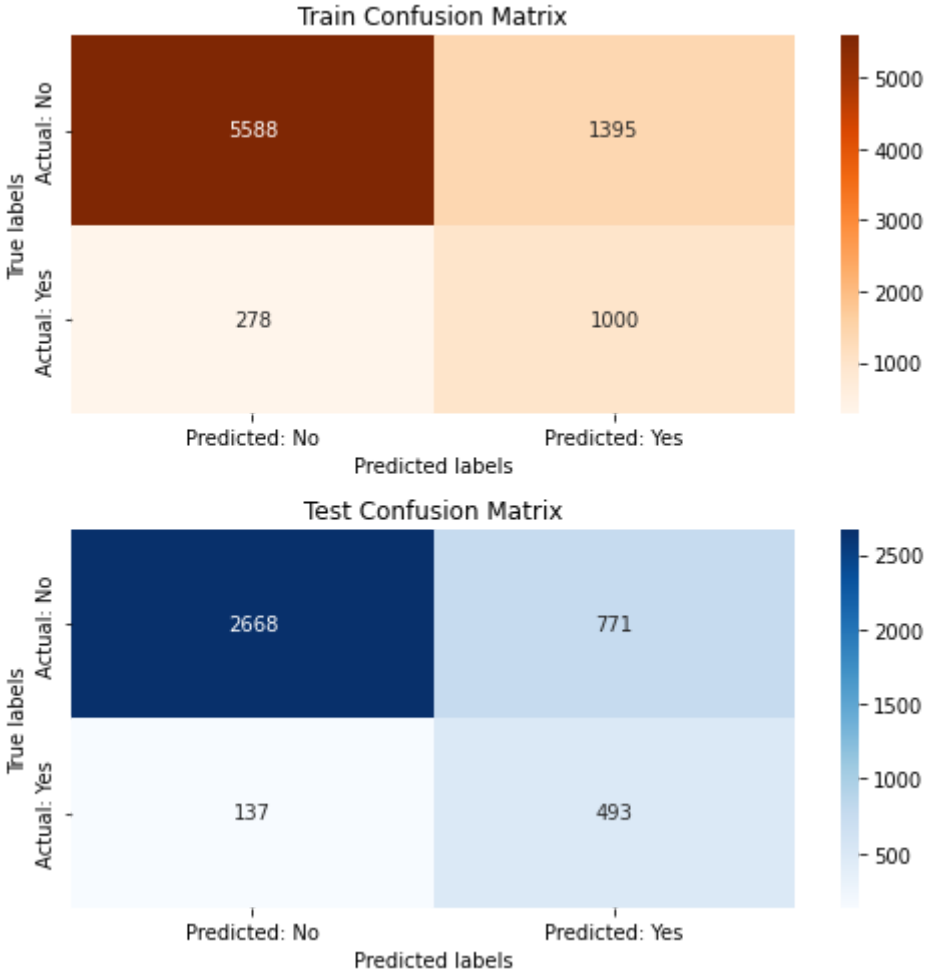
Confusion Matrix for Logistic Regression

In [189…

```
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
y_test_preicted = predict_with_best_t(y_test_pred, best_t)
train_conf = confusion_matrix(y_train, predict_with_best_t(y_train_pred, be
test_conf = confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_



#https://stackoverflow.com/questions/61748441/how-to-fix-the-values-display
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax= plt.subplots(nrows=2)
#https://stackoverflow.com/questions/41659188/how-to-adjust-subplot-size-in
plt.rcParams['figure.figsize']=(7,7)

sns.heatmap(train_conf, annot=True,fmt="d",cmap='Oranges',ax=ax[0]);
# labels, title and ticks
ax[0].set_xlabel('Predicted labels');
ax[0].set_ylabel('True labels');
ax[0].set_ylim(2.0, 0)
ax[0].set_title('Train Confusion Matrix');
ax[0].xaxis.set_ticklabels(['Predicted: No','Predicted: Yes']);
ax[0].yaxis.set_ticklabels(['Actual: No','Actual: Yes']);

sns.heatmap(test_conf, annot=True,fmt="d",cmap='Blues',ax=ax[1]);
# labels, title and ticks
ax[1].set_xlabel('Predicted labels');ax[1].set_ylabel('True labels');
ax[1].set_ylim(2.0, 0)
```

```
ax[1].set_title('Test Confusion Matrix');
ax[1].xaxis.set_ticklabels(['Predicted: No','Predicted: Yes']);
ax[1].yaxis.set_ticklabels(['Actual: No','Actual: Yes']);


fig.tight_layout()
```

```
the maximum value of tpr*(1-fpr) 0.6261573770594673 for threshold 0.149
```



Summary

In [190…

```
from prettytable import PrettyTable

summary = PrettyTable()
summary.field_names = ["Model", "Hyper Parameter", "AUC"]
summary.add_row(["Decision Tree", str(max_depth)+'-'+str(min_samples_split)
summary.add_row(["Logistic Regression", round(best_hyper_param,3), best_auc
#summary.add_row(["GradientBoostingClassifier", str(best_n_estimators)+'-'+
print(summary)
```

```
+---------------------+-----------------+--------------------+
|        Model        | Hyper Parameter |        AUC         |
+---------------------+-----------------+--------------------+
|    Decision Tree    |      10-500     | 0.8804885141029369 |
| Logistic Regression |      100.0      | 0.8715534231527253 |
+---------------------+-----------------+--------------------+
```