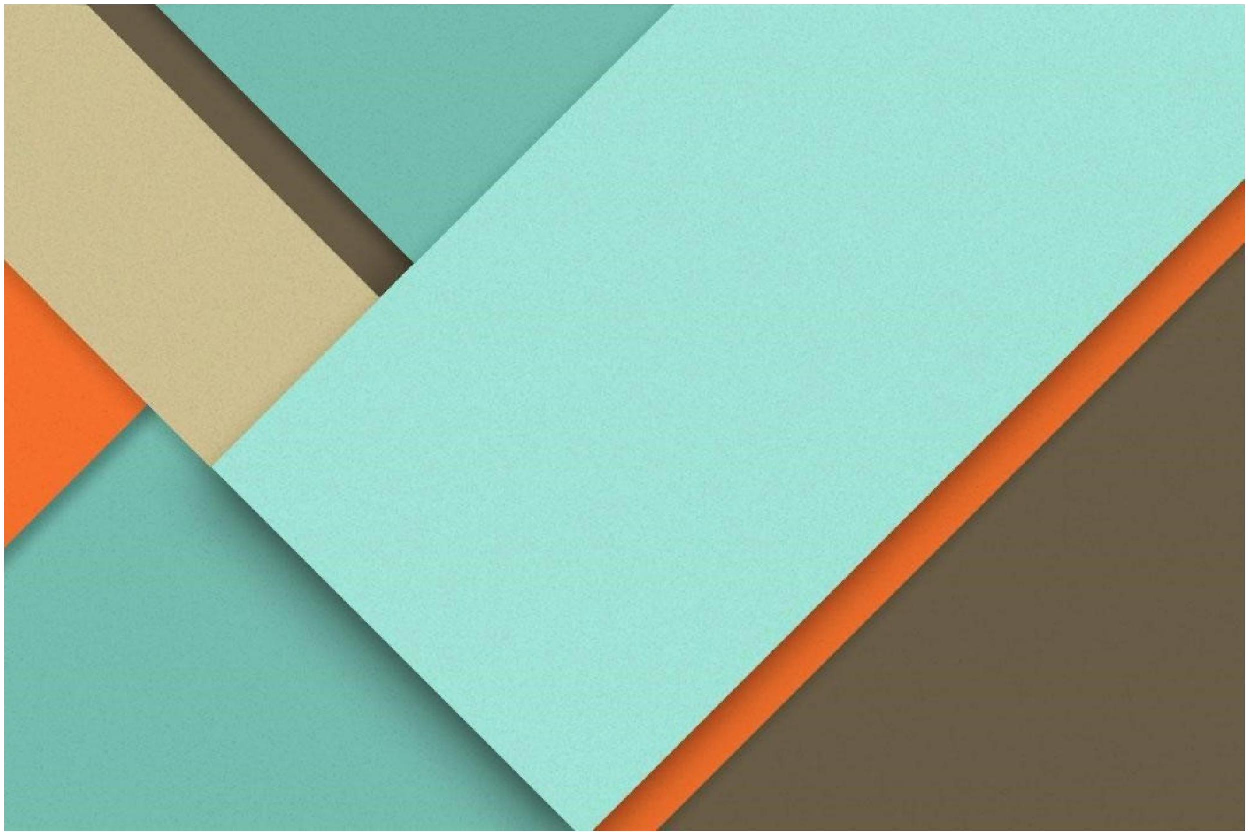


Telco - Customer Churn

10.02.2018

Annapoorani
Springboard
Capstone project proposal

Problem Statement

Predict behavior to retain customers. To can analyze all relevant customer data and develop focused customer retention programs.

Outcomes

Telco - a Telecommunication company will be beneficial by this project. This will help them in the following ways

- Improve user experience with their products
- Identifying important features that are key for customer retention
    - To come up with new Marketing Strategies.

s.

Data

Telco data collected from Kaggle.

https://www.kaggle.com/blastchar/telco-customer-churn

Approach

The Steps to solve the identified problems are as below

- Data Wrangling using Python Pandas, so that data is cleaned and readily available for analysis
- EDA to discover the underlying facts about the data and to study the different features

- Applying machine learning algorithms like Logistic regression, Decision Trees and Deep Learning.

Data Wrangling

Success of analysis depends upon how the data is cleaned up as usable for analysis with columns as separate features and each row as single observation. As it is highlighted always as Data Wrangling is time consuming, wrangling for this project also was very challenging.

As this data is downloaded from Kaggle it is already cleaned. Two main steps performed is

1. Converting the Total and Monthly charges to numeric
2. Checked for any NA values in the columns, identified very few rows with Total Charges as NA and those rows are dropped

Data Pre-Processing

Most features of the Telco – Customer Churn Dataset are categorical variables. One hot encoding is performed on those categories and redundant columns are removed.

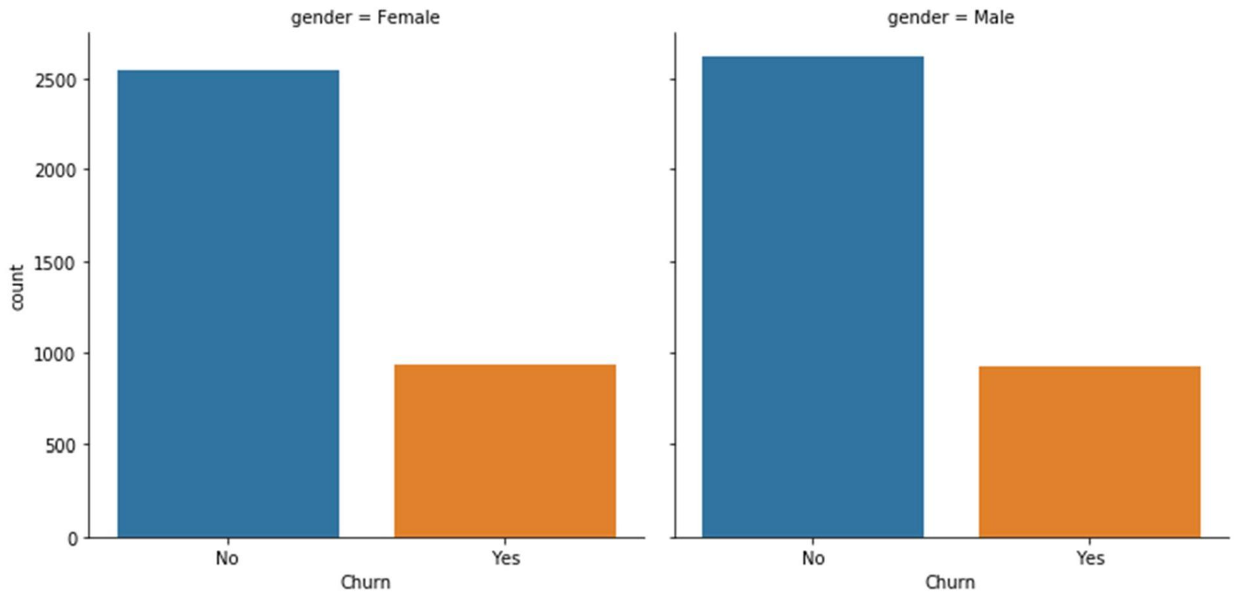
One major column of interest is the Internet Service, if the column is ‘No’ then obviously the concern customer won’t have Streaming TV or Streaming Movies.

Henceforth “Internet_Service_No” column is retained by dropping Streaming_TV_No and Streaming_Movies_No columns.

Exploratory Data Analysis

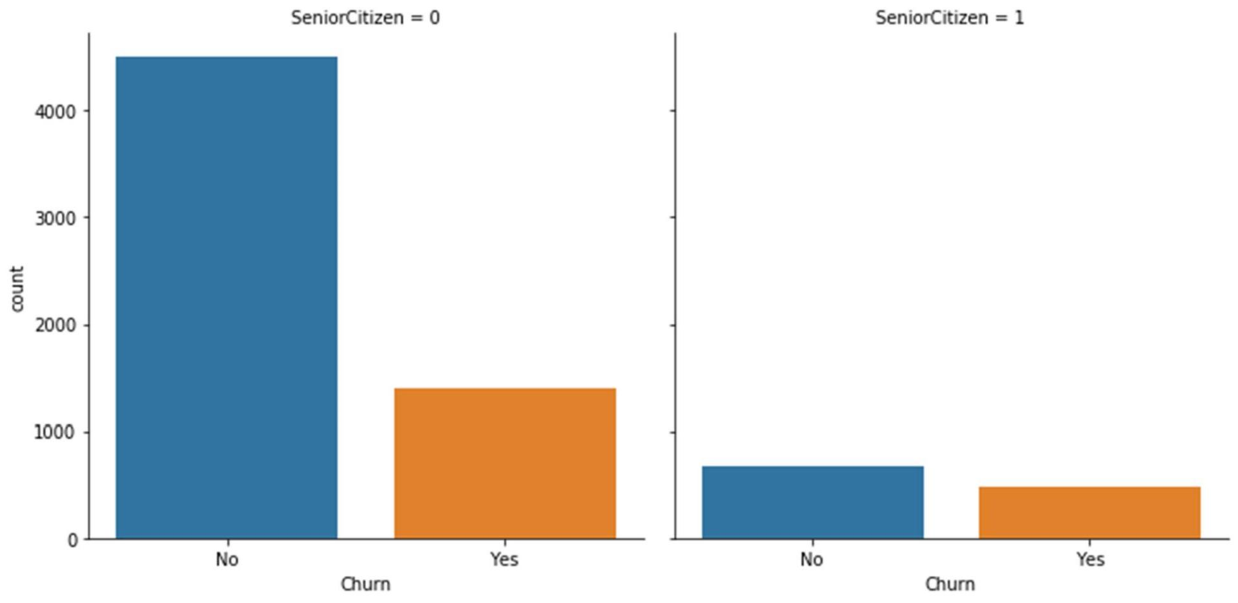
EDA helps in Visualizing underlying insights from the data. It helps in Feature Engineering also paves way for identifying new scope of data. As churn is the key field of study, first part of the notebook deals with the picturization of churn data based on different features.

Churn Based on Gender



As from above chart churning rate looks similar for both male and female so there isno significant difference in churning rate based on gender.

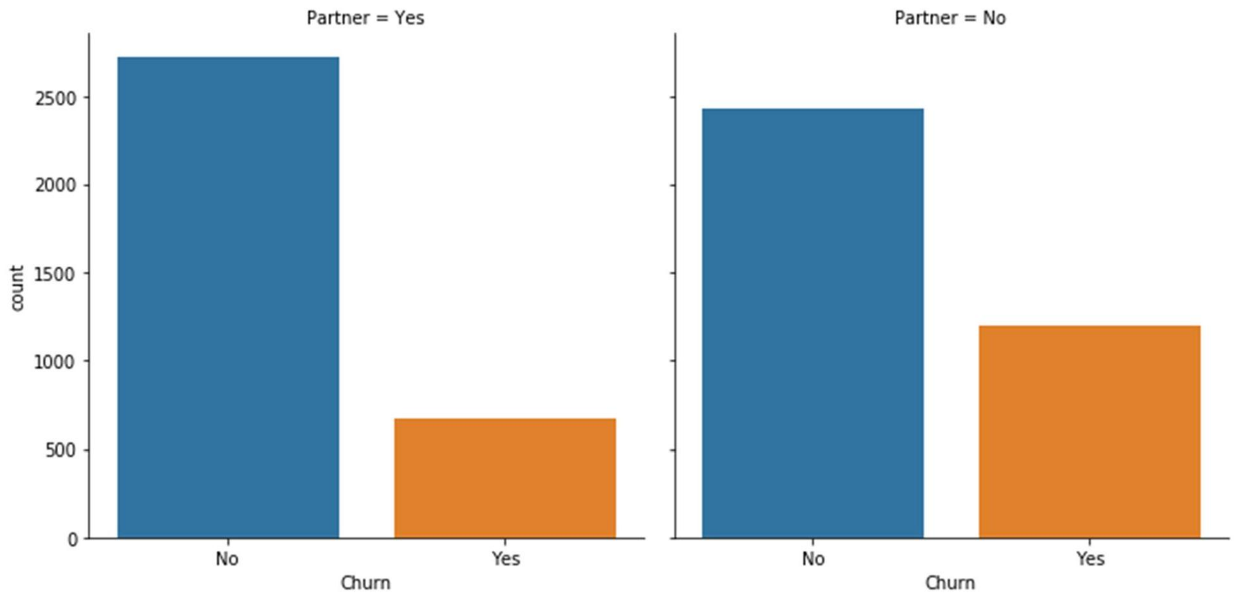
Churn Behavior with Senior Citizen



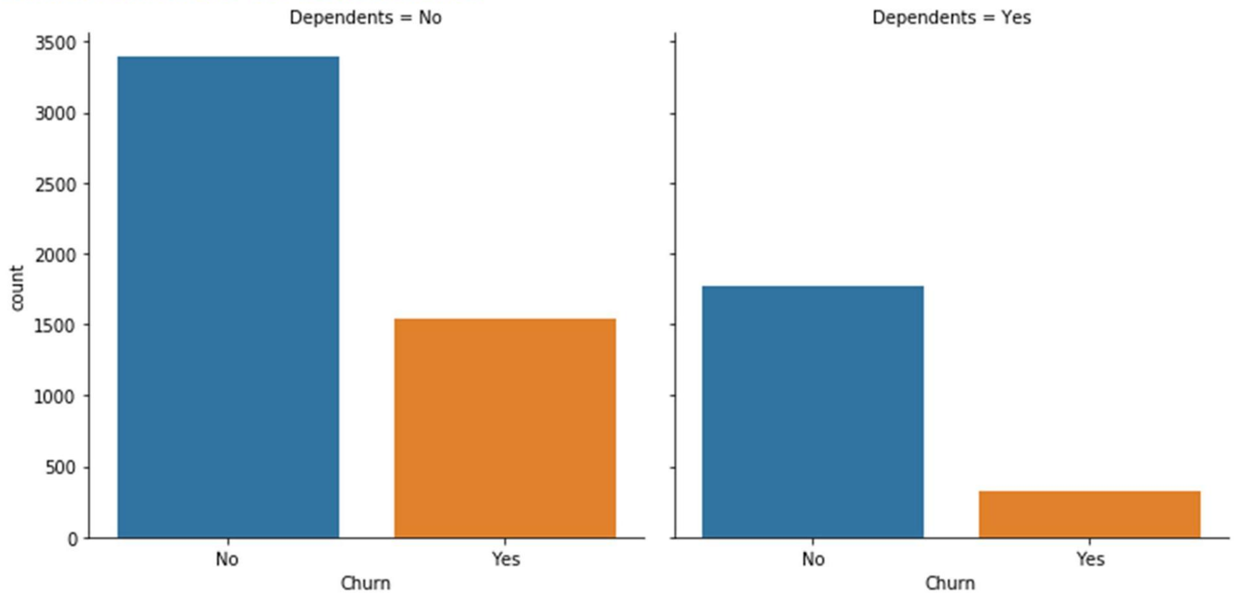
It seems Senior citizens churned and not churned numbers have very less difference, also the number of Total Senior Citizens is less compared to remaining population.

Partners Churn

If there is no partner then churn rate is little higher.

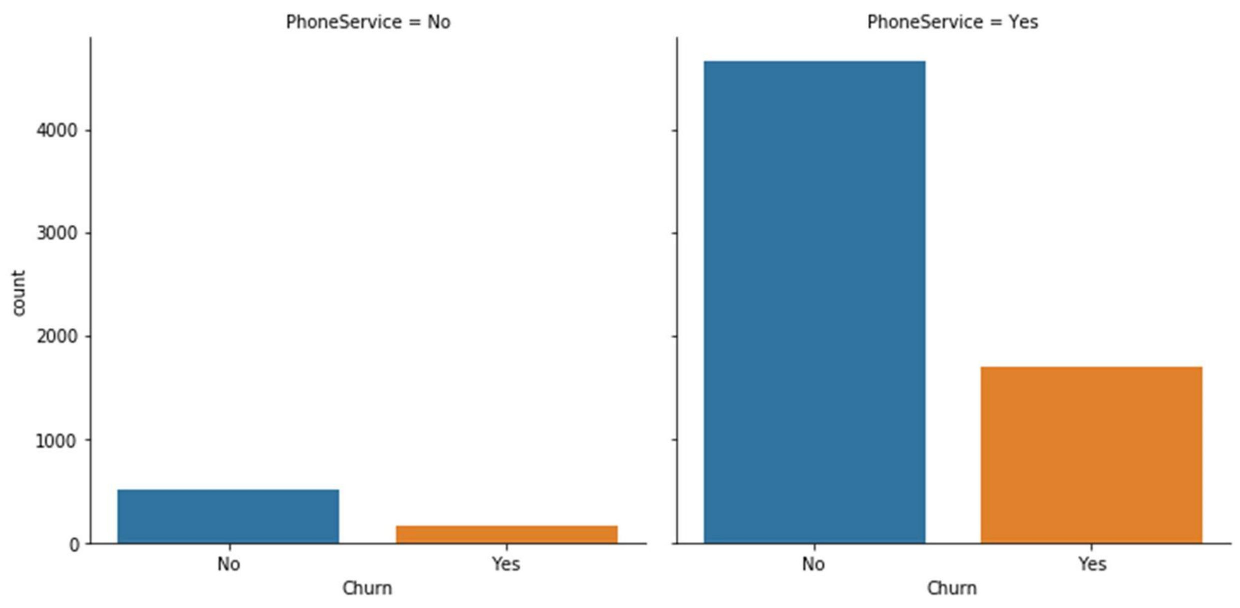


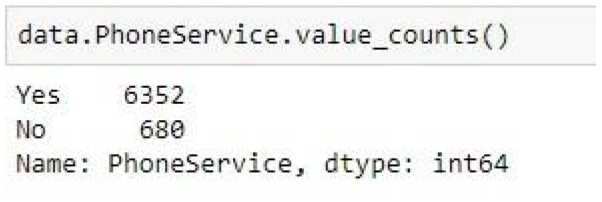
Churn Numbers for Dependents



If there are no Dependents then Churning numbers are higher

Customer Churn with Phone Service

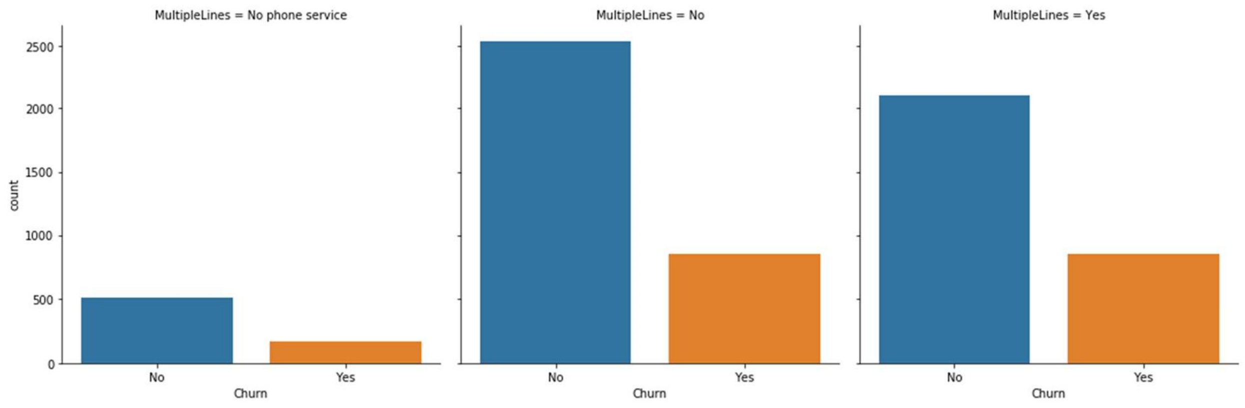




```
data.PhoneService.value_counts()

Yes    6352
No      680
Name: PhoneService, dtype: int64
```

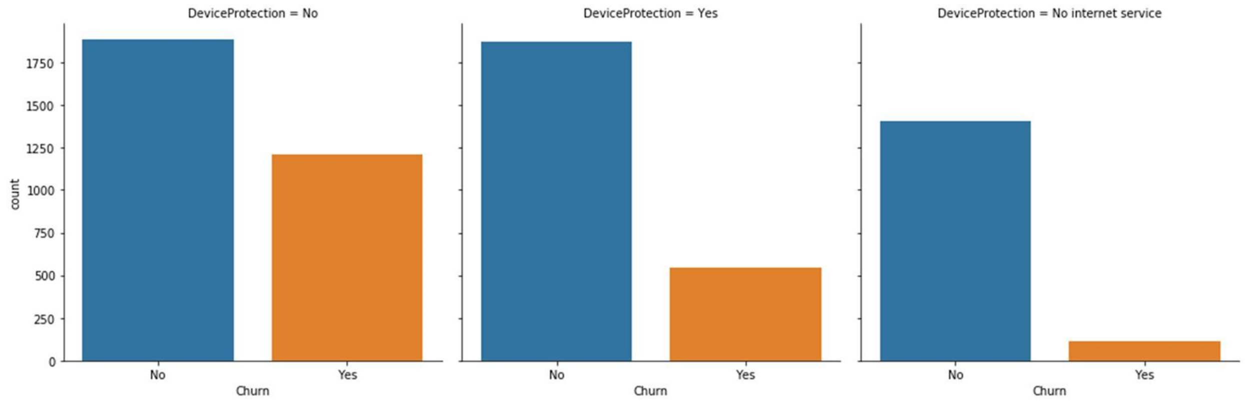
Most of the Telcom customers have Phone service too, nearly 10% of them Phone Service customers are churned

Churn based on Multiple Lines



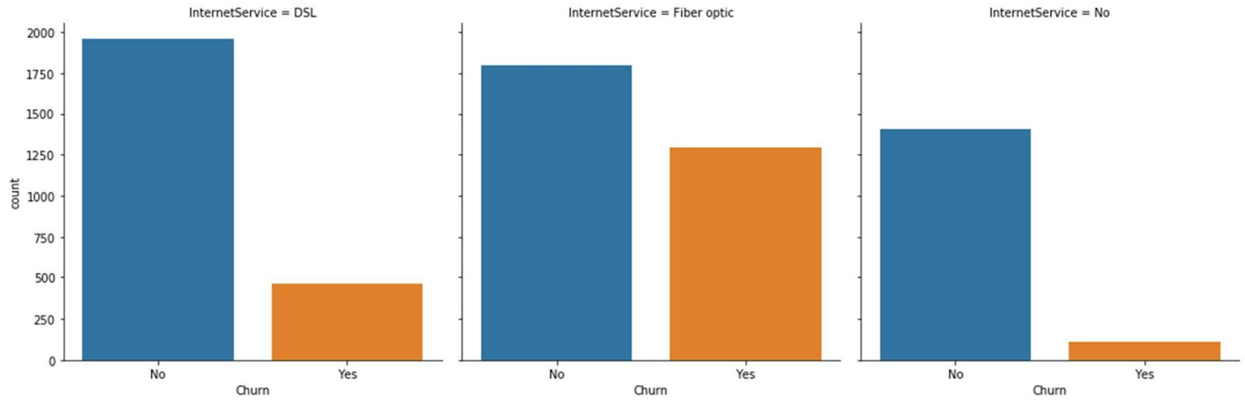
Irrespective of customer have multiple lines or not the Churning numbers are same.

Churn based on Device Protection



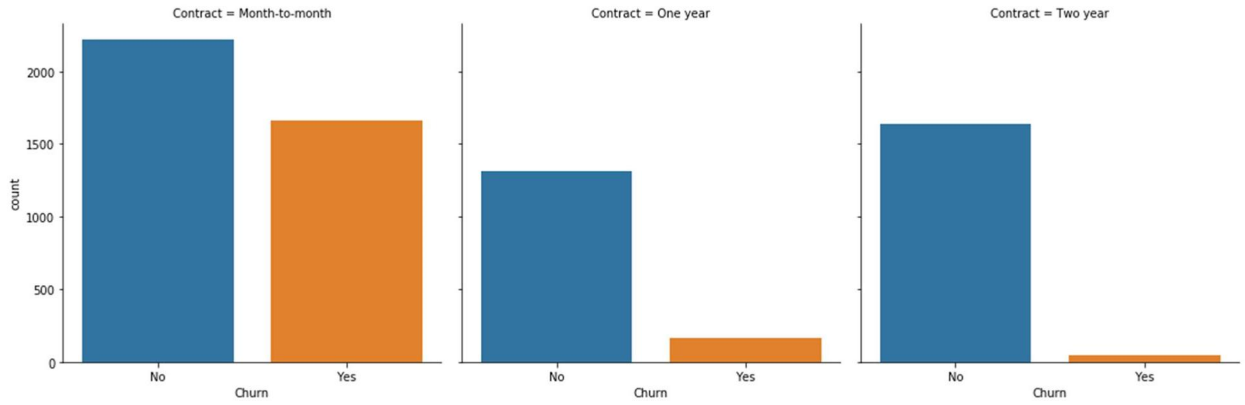
Customer without device protection churned higher

Churn based on Customer option on Internet service



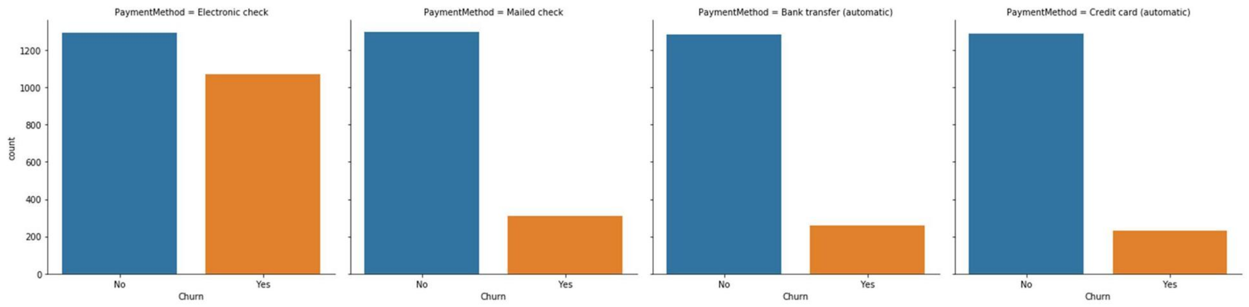
Customers getting Internet Service by Fiber optics Churned more, so definetly that may be a important feature.

Churn based on Contract Pattern



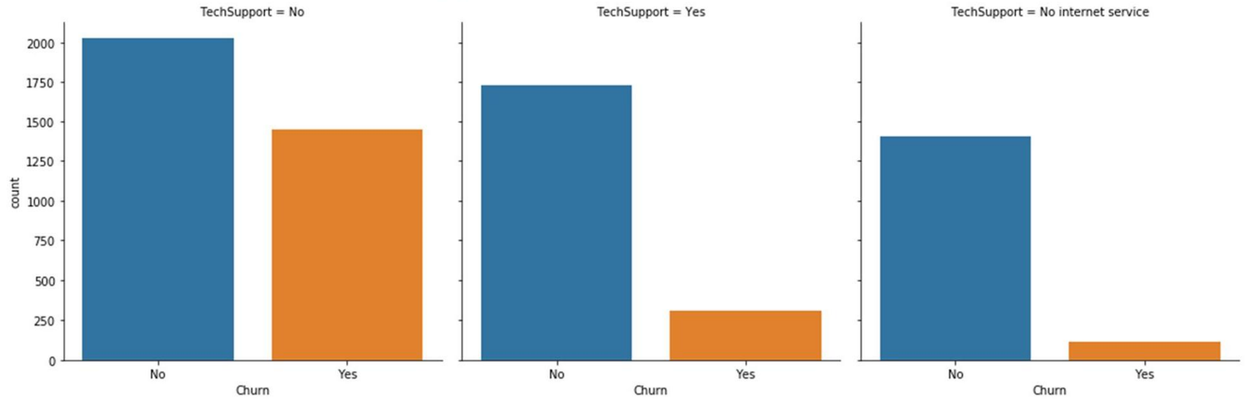
Customers chosen Month-to-month contract churned higher, this also may be important feature

Payment Method – Churn Behavior



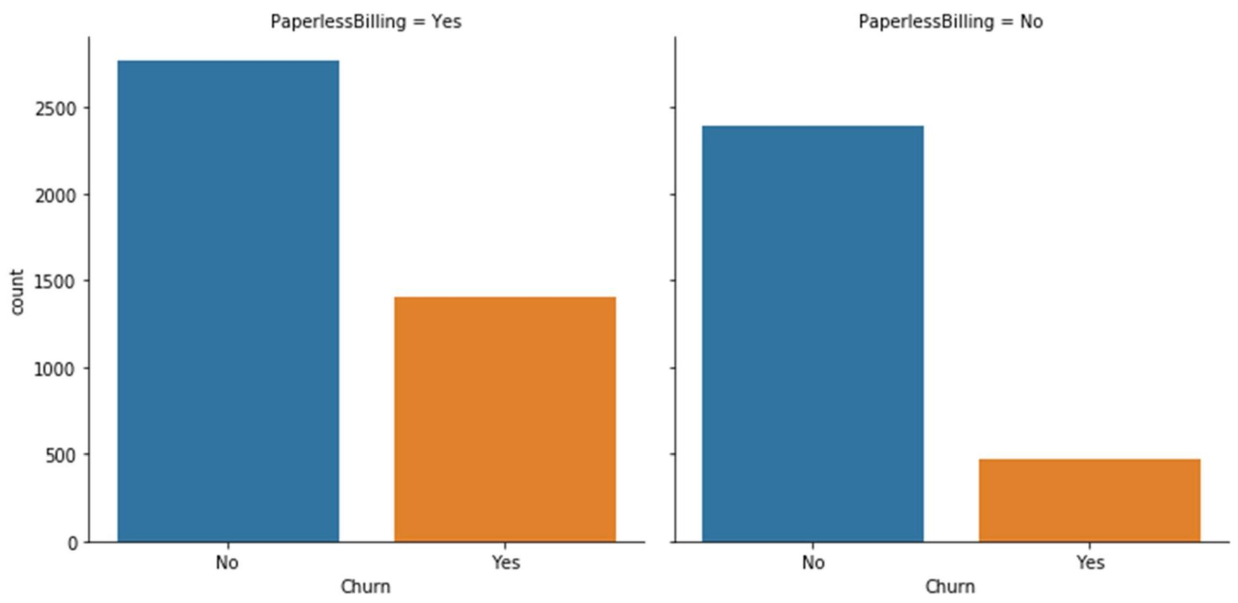
Customer paid by electronic check churned more

Customer Churn based on support

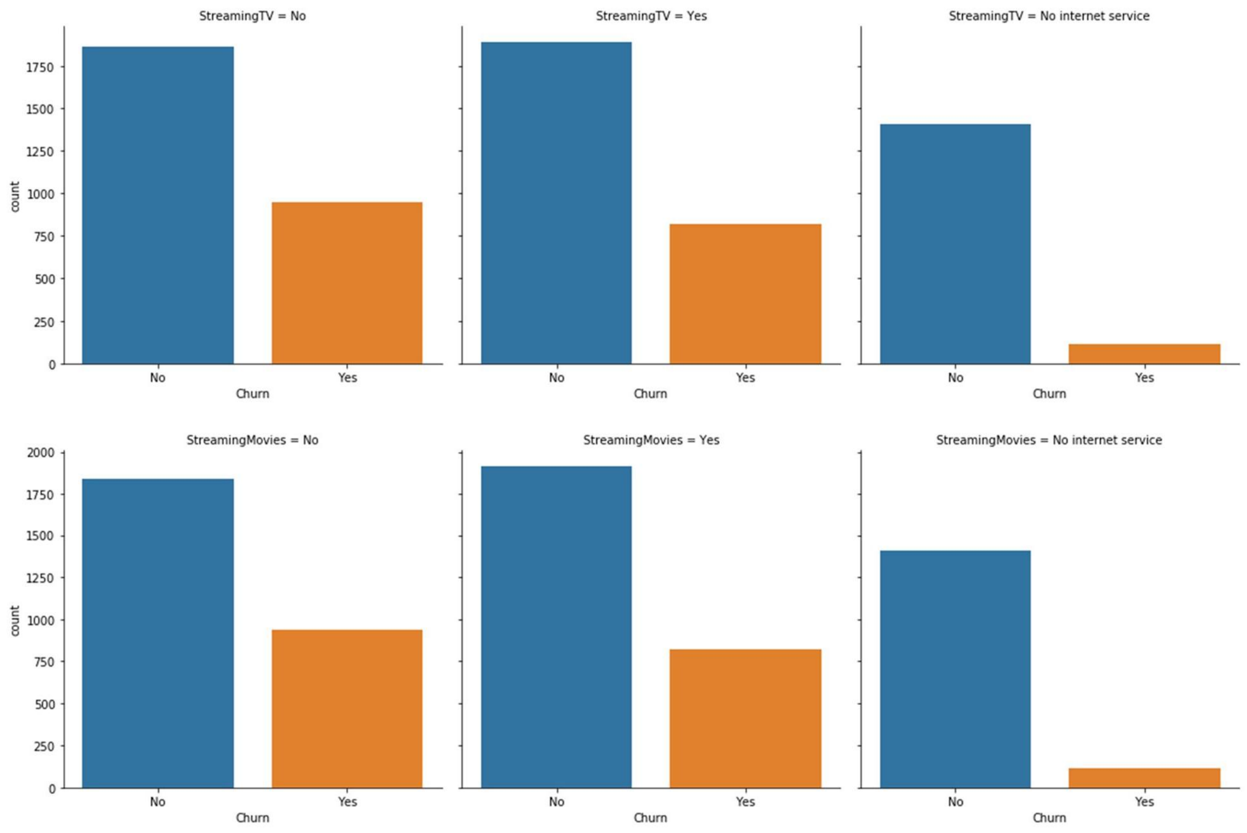


Customer without tech support churned higher than customer with tech support

Churn based on Billing Method



Churn by Streaming Services

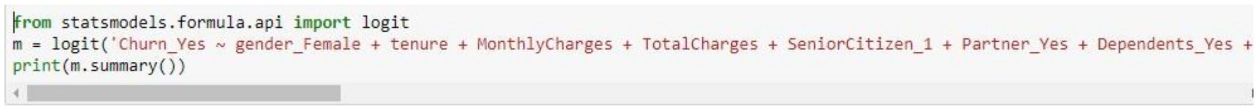


Both Streaming Tv and Streaming Movies have greater churn rate

Feature Engineering

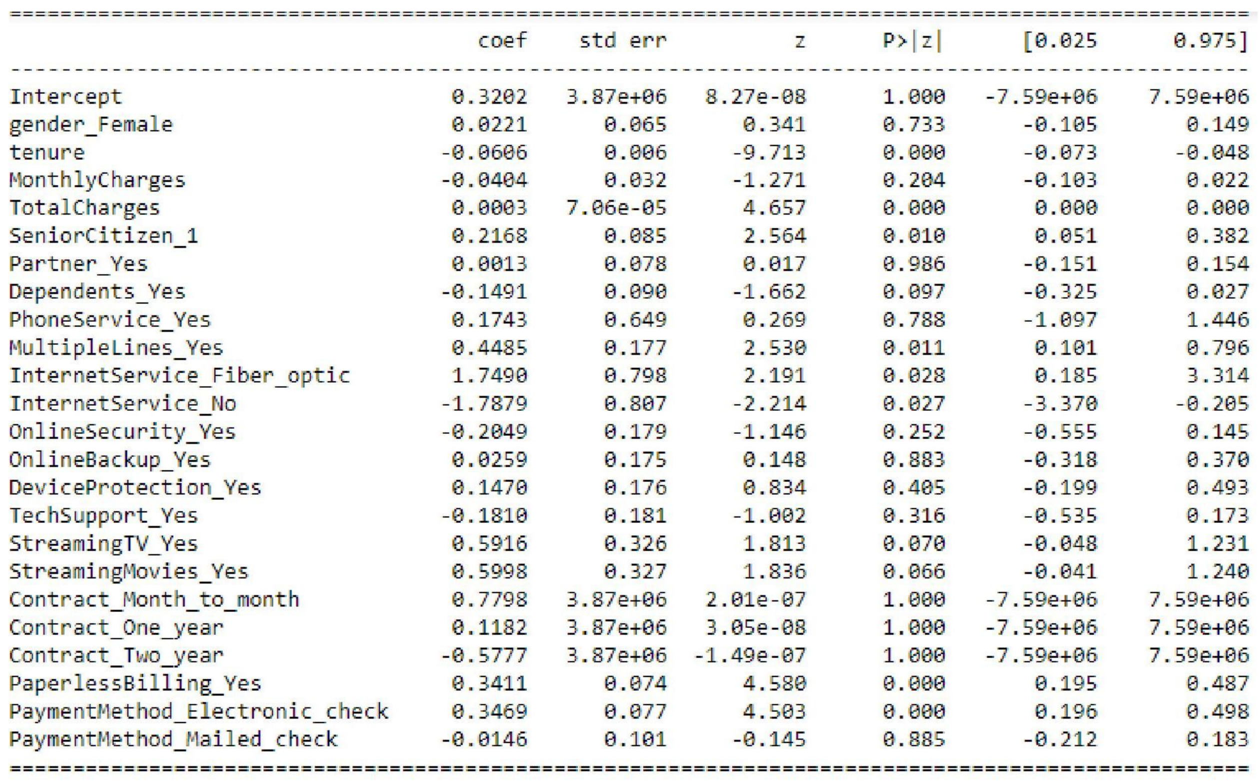
From EDA Internet Service, Contract, Payment Method, Streaming Tv and Streaming Movies, Tech Support are the Primary features and Paperless billing and Senior Citizen can be of secondary importance.

Second way is by applying stats model Logit function



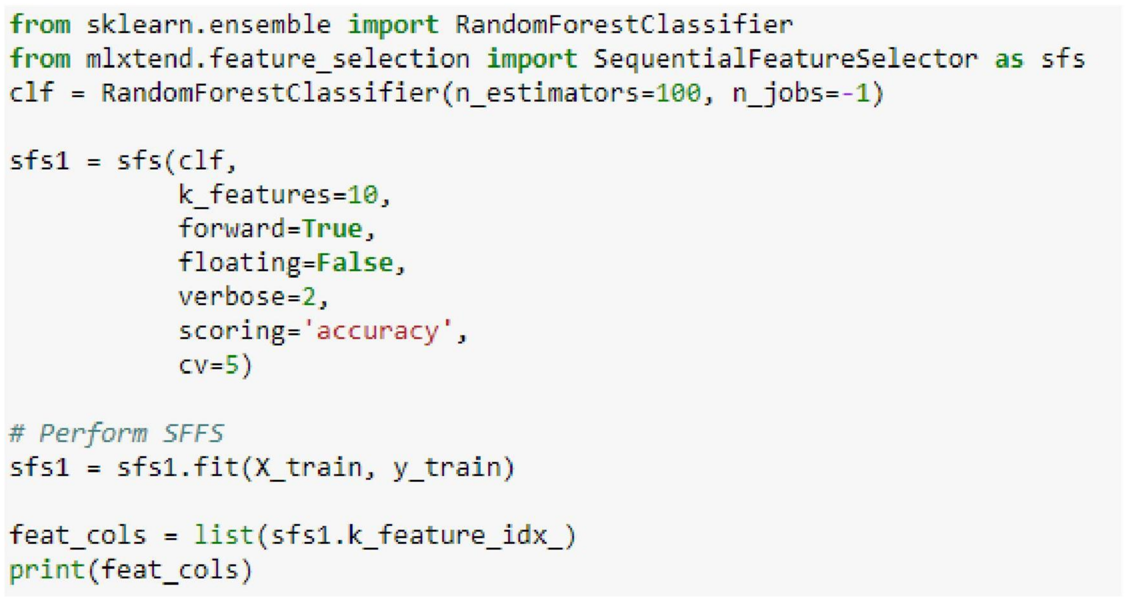
```
from statsmodels.formula.api import logit
m = logit('Churn_Yes ~ gender_Female + tenure + MonthlyCharges + TotalCharges + SeniorCitizen_1 + Partner_Yes + Dependents_Yes +
print(m.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.414311
         Iterations 9
                           Logit Regression Results
==============================================================================
Dep. Variable:              Churn_Yes   No. Observations:                 7032
Model:                          Logit   Df Residuals:                     7009
Method:                           MLE   Df Model:                           22
Date:                Wed, 03 Oct 2018   Pseudo R-squ.:                  0.2845
Time:                        12:51:43   Log-Likelihood:                -2913.4
converged:                       True   LL-Null:                       -4071.7
                                        LLR p-value:                     0.000
```

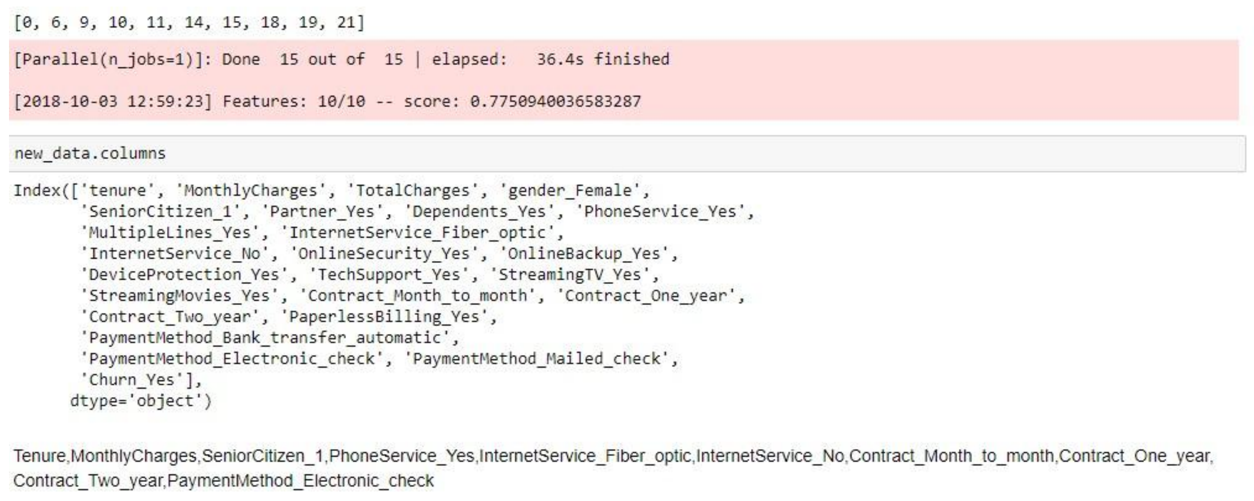


```
==================================================================================================
                                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
Intercept                        0.3202   3.87e+06   8.27e-08      1.000   -7.59e+06    7.59e+06
gender_Female                    0.0221      0.065      0.341      0.733      -0.105       0.149
tenure                          -0.0606      0.006     -9.713      0.000      -0.073      -0.048
MonthlyCharges                  -0.0404      0.032     -1.271      0.204      -0.103       0.022
TotalCharges                     0.0003   7.06e-05      4.657      0.000       0.000       0.000
SeniorCitizen_1                  0.2168      0.085      2.564      0.010       0.051       0.382
Partner_Yes                      0.0013      0.078      0.017      0.986      -0.151       0.154
Dependents_Yes                  -0.1491      0.090     -1.662      0.097      -0.325       0.027
PhoneService_Yes                 0.1743      0.649      0.269      0.788      -1.097       1.446
MultipleLines_Yes                0.4485      0.177      2.530      0.011       0.101       0.796
InternetService_Fiber_optic      1.7490      0.798      2.191      0.028       0.185       3.314
InternetService_No              -1.7879      0.807     -2.214      0.027      -3.370      -0.205
OnlineSecurity_Yes              -0.2049      0.179     -1.146      0.252      -0.555       0.145
OnlineBackup_Yes                 0.0259      0.175      0.148      0.883      -0.318       0.370
DeviceProtection_Yes             0.1470      0.176      0.834      0.405      -0.199       0.493
TechSupport_Yes                 -0.1810      0.181     -1.002      0.316      -0.535       0.173
StreamingTV_Yes                  0.5916      0.326      1.813      0.070      -0.048       1.231
StreamingMovies_Yes              0.5998      0.327      1.836      0.066      -0.041       1.240
Contract_Month_to_month          0.7798   3.87e+06   2.01e-07      1.000   -7.59e+06    7.59e+06
Contract_One_year                0.1182   3.87e+06   3.05e-08      1.000   -7.59e+06    7.59e+06
Contract_Two_year               -0.5777   3.87e+06  -1.49e-07      1.000   -7.59e+06    7.59e+06
PaperlessBilling_Yes             0.3411      0.074      4.580      0.000       0.195       0.487
PaymentMethod_Electronic_check   0.3469      0.077      4.503      0.000       0.196       0.498
PaymentMethod_Mailed_check      -0.0146      0.101     -0.145      0.885      -0.212       0.183
==================================================================================================
```

The third approach is to apply Random Forest Classifier to identify the important approach



```python
from sklearn.ensemble import RandomForestClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
clf = RandomForestClassifier(n_estimators=100, n_jobs=-1)

sfs1 = sfs(clf,
           k_features=10,
           forward=True,
           floating=False,
           verbose=2,
           scoring='accuracy',
           cv=5)

# Perform SFFS
sfs1 = sfs1.fit(X_train, y_train)

feat_cols = list(sfs1.k_feature_idx_)
print(feat_cols)
```



```
[0, 6, 9, 10, 11, 14, 15, 18, 19, 21]
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:   36.4s finished

[2018-10-03 12:59:23] Features: 10/10 -- score: 0.7750940036583287
```

```
new_data.columns
```

```
Index(['tenure', 'MonthlyCharges', 'TotalCharges', 'gender_Female',
       'SeniorCitizen_1', 'Partner_Yes', 'Dependents_Yes', 'PhoneService_Yes',
       'MultipleLines_Yes', 'InternetService_Fiber_optic',
       'InternetService_No', 'OnlineSecurity_Yes', 'OnlineBackup_Yes',
       'DeviceProtection_Yes', 'TechSupport_Yes', 'StreamingTV_Yes',
       'StreamingMovies_Yes', 'Contract_Month_to_month', 'Contract_One_year',
       'Contract_Two_year', 'PaperlessBilling_Yes',
       'PaymentMethod_Bank_transfer_automatic',
       'PaymentMethod_Electronic_check', 'PaymentMethod_Mailed_check',
       'Churn_Yes'],
      dtype='object')
```

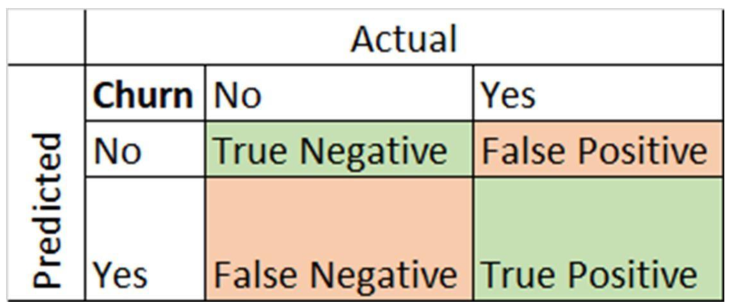
Tenure,MonthlyCharges,SeniorCitizen_1,PhoneService_Yes,InternetService_Fiber_optic,InternetService_No,Contract_Month_to_month,Contract_One_year,
Contract_Two_year,PaymentMethod_Electronic_check

From all the above three methods the concluded features are Tenure, Monthly Charges, SeniorCitizen_1, PhoneService_Yes, InternetService_Fiber_optic, InternetService_No, Contract_Month_to_month, Contract_One_year, Contract_Two_year, PaymentMethod_Electronic_check

CLASSIFICATION

Approaches for Predicting Qualitative Responses are called Classification. There are many classification techniques, widely used are Logistic Regression, Decision Trees, SVM, Random Forest and XG Boost. Deep Learning also can be applied for classification problems.

Confusion Matrix



	Actual		
Predicted	**Churn**	No	Yes
	No	True Negative	False Positive
	Yes	False Negative	True Positive

The above table is called the Confusion Matrix, it clearly helps in studying our model performance.

- True Positive - Actual Data point is positive and Predicted as positive
- True Negative - Actual Data point is negative and Predicted as negative False Positive - Actual Data point is negative and Predicted as positive False Negative - Actual Data point is positive and Predicted as negative.

From above four values two metrics are calculated

- Precision = True Positives/ (True Positives + False Positives)
- Recall = True Positives/ (True Positives + False Negatives)

Precision talks about out of Total predicted positive values how many are actually Positive. Recall talks about the Total positive values predicted right.

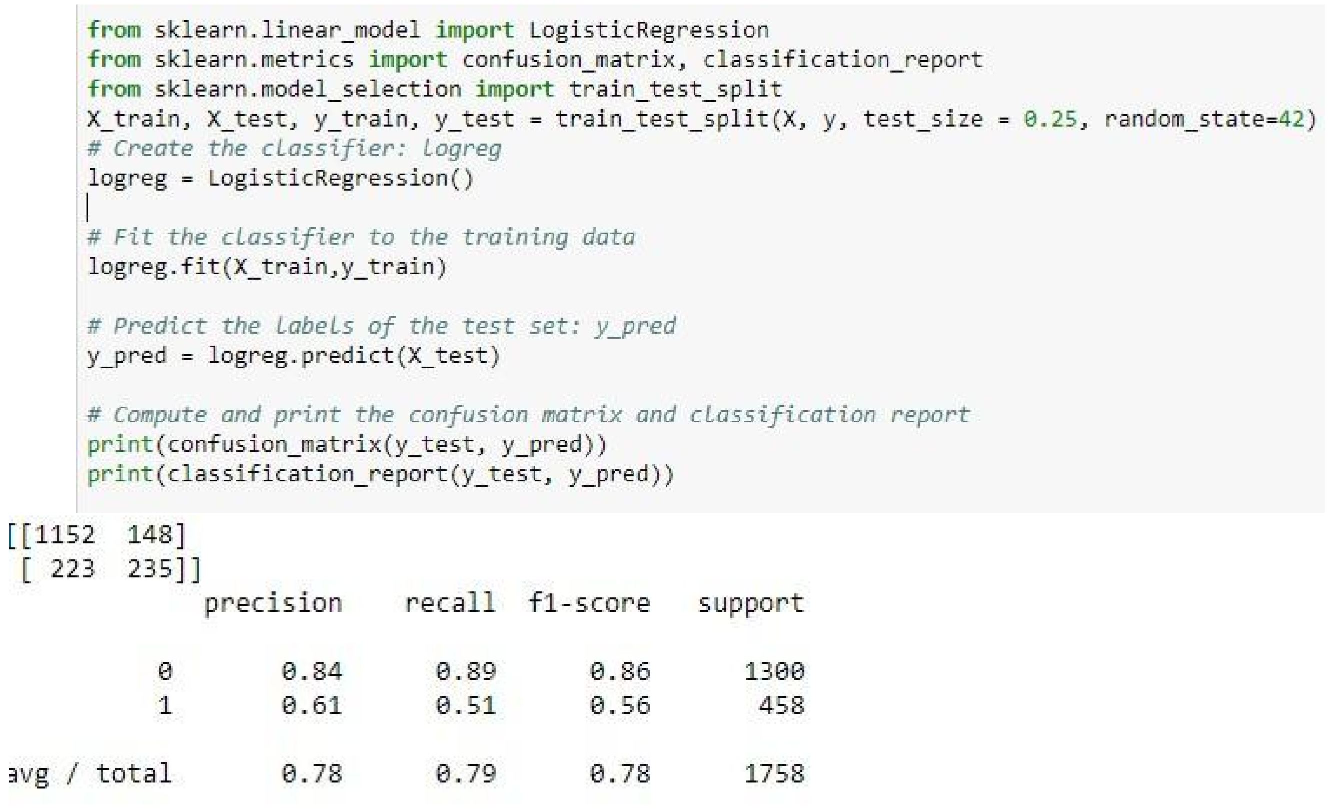
- Cost of False Positive is high, then Precision is a good measure Cost of False Negative is high, then recall is a good measure.

For Customer churn classification, False Negative cost is high that is if a customer churned, is identified as not churned customer. So, Recall is an important measure for this dataset.

Logistic Regression

Logistic Regression models the probability that the output variable belongs to a particular category. Logistic Regression uses Logistic function as follows

p(X) = e^(b0 + b1*X) / (1 + e^(b0 + b1*X))



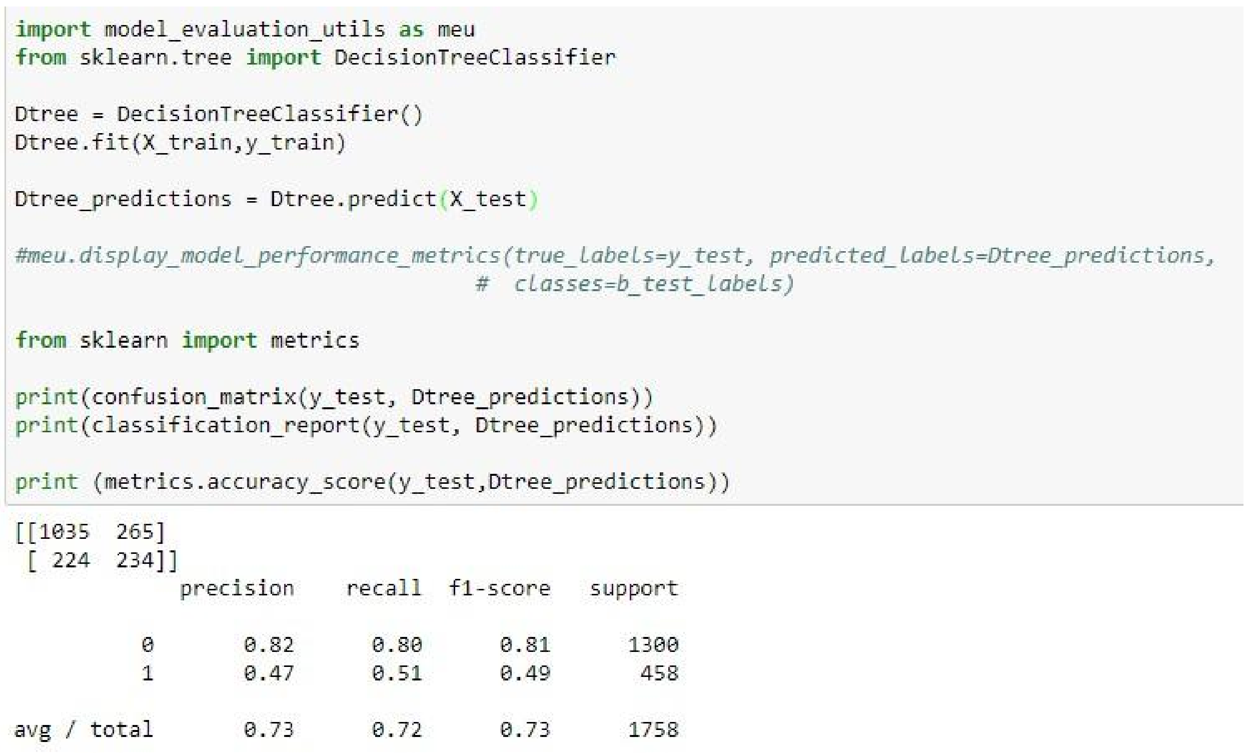
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state=42)
# Create the classifier: logreg
logreg = LogisticRegression()

# Fit the classifier to the training data
logreg.fit(X_train,y_train)

# Predict the labels of the test set: y_pred
y_pred = logreg.predict(X_test)

# Compute and print the confusion matrix and classification report
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[1152  148]
 [ 223  235]]
             precision    recall  f1-score   support

          0       0.84      0.89      0.86      1300
          1       0.61      0.51      0.56       458

avg / total       0.78      0.79      0.78      1758
```

Decision Trees

Decision Trees are non- non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

- Decision Trees can be able to handle both continuous and categorical data.
- The main disadvantage of Decision trees is it will create overcomplex trees that lead to Overfitting. It also creates biased tree if one class dominates.

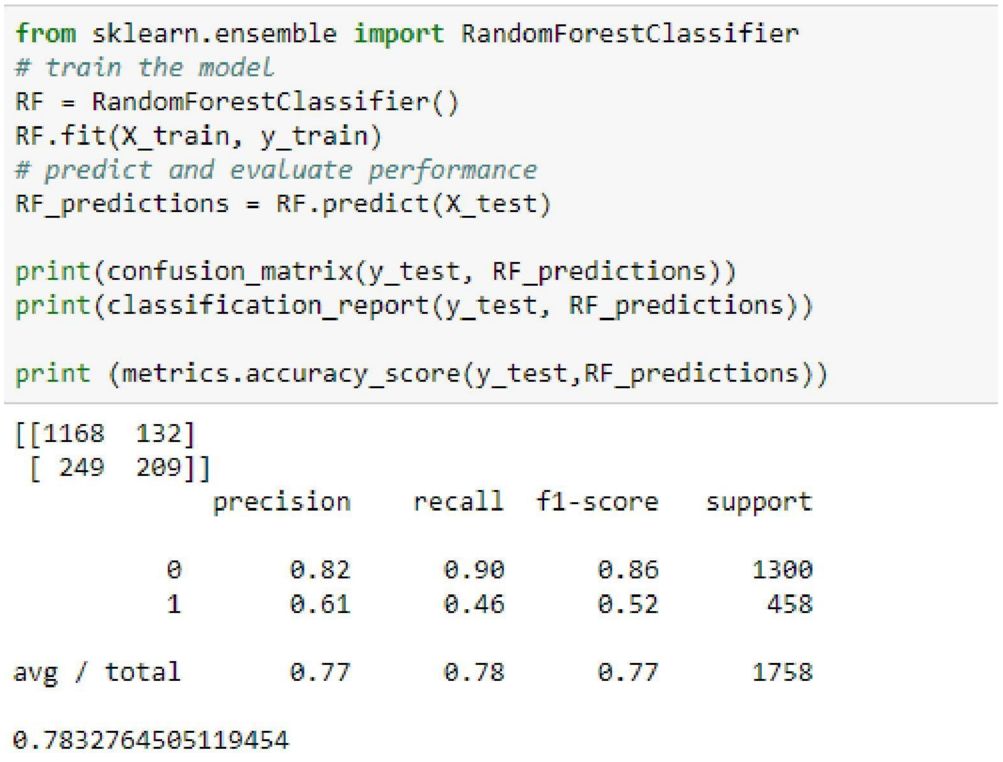


```
import model_evaluation_utils as meu
from sklearn.tree import DecisionTreeClassifier

Dtree = DecisionTreeClassifier()
Dtree.fit(X_train,y_train)

Dtree_predictions = Dtree.predict(X_test)

#meu.display_model_performance_metrics(true_labels=y_test, predicted_labels=Dtree_predictions,
                                    #  classes=b_test_labels)

from sklearn import metrics

print(confusion_matrix(y_test, Dtree_predictions))
print(classification_report(y_test, Dtree_predictions))

print (metrics.accuracy_score(y_test,Dtree_predictions))
```

```
[[1035  265]
 [ 224  234]]
             precision    recall  f1-score   support

          0       0.82      0.80      0.81      1300
          1       0.47      0.51      0.49       458

avg / total       0.73      0.72      0.73      1758
```

Random Forest

In Random Forest each tree is built from samples wit replacement from training set. Split is not chosen as the best split among features instead built from random subset of features.

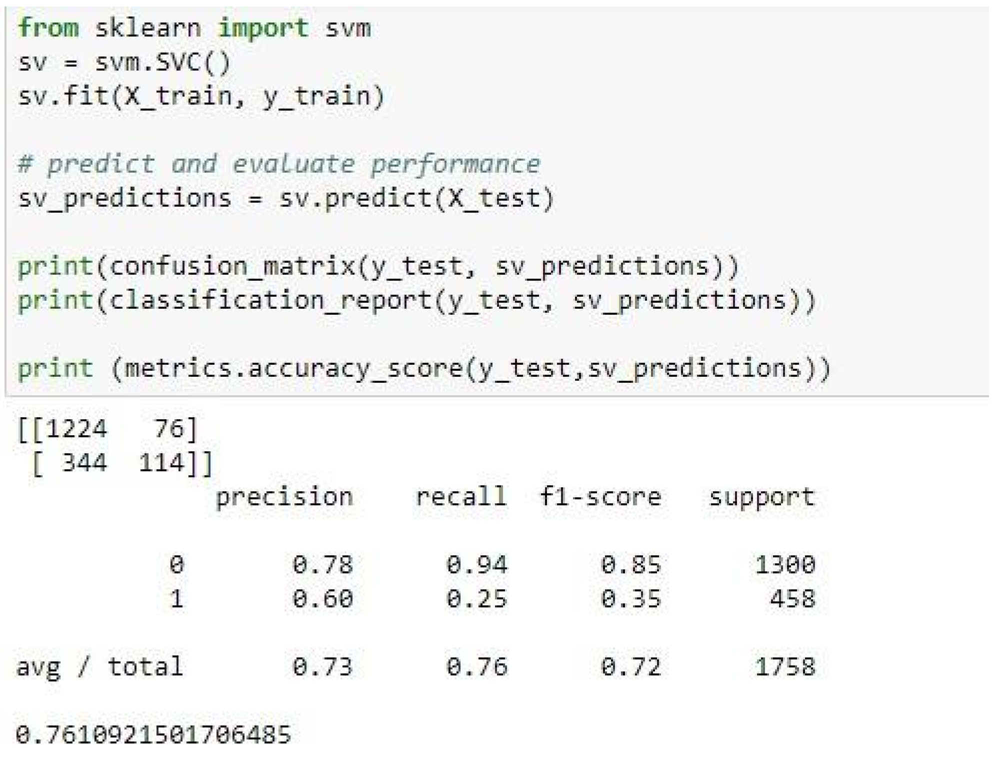


```
from sklearn.ensemble import RandomForestClassifier
# train the model
RF = RandomForestClassifier()
RF.fit(X_train, y_train)
# predict and evaluate performance
RF_predictions = RF.predict(X_test)

print(confusion_matrix(y_test, RF_predictions))
print(classification_report(y_test, RF_predictions))

print (metrics.accuracy_score(y_test,RF_predictions))
```

```
[[1168  132]
 [ 249  209]]
             precision    recall  f1-score   support

          0       0.82      0.90      0.86      1300
          1       0.61      0.46      0.52       458

avg / total       0.77      0.78      0.77      1758

0.7832764505119454
```

SVM

- SVM - Support Vector Machines are very effective in high dimensional spaces.
- Rather than simply drawing a zero-width line between the classes, margin of some width can be drawn around each line, up to the nearest point.

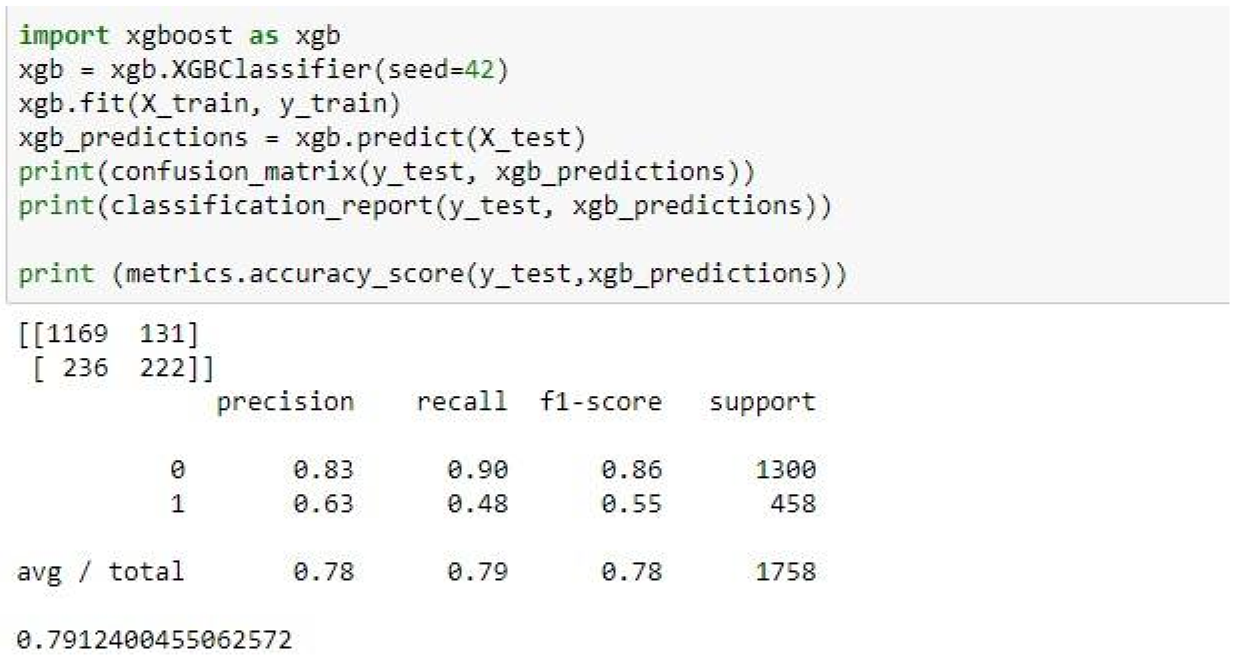


```
from sklearn import svm
sv = svm.SVC()
sv.fit(X_train, y_train)

# predict and evaluate performance
sv_predictions = sv.predict(X_test)

print(confusion_matrix(y_test, sv_predictions))
print(classification_report(y_test, sv_predictions))

print (metrics.accuracy_score(y_test,sv_predictions))
```

```
[[1224   76]
 [ 344  114]]
             precision    recall  f1-score   support

          0       0.78      0.94      0.85      1300
          1       0.60      0.25      0.35       458

avg / total       0.73      0.76      0.72      1758

0.7610921501706485
```

XG Boost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

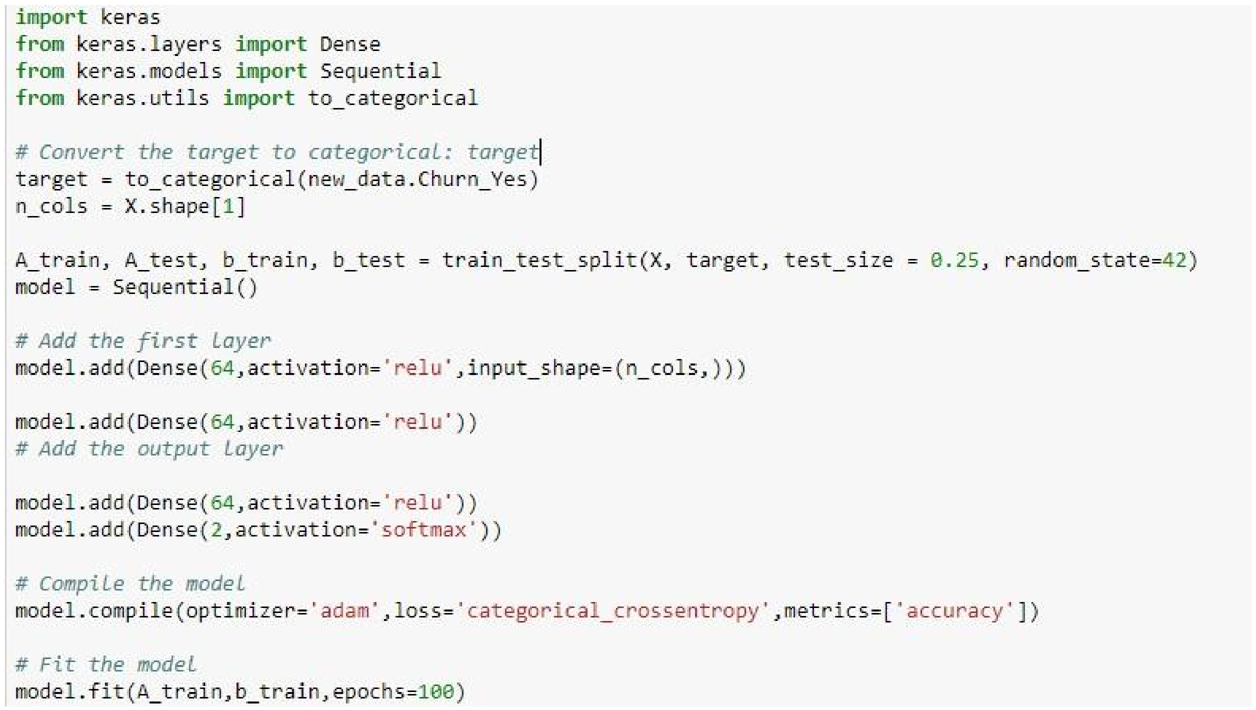


```
import xgboost as xgb
xgb = xgb.XGBClassifier(seed=42)
xgb.fit(X_train, y_train)
xgb_predictions = xgb.predict(X_test)
print(confusion_matrix(y_test, xgb_predictions))
print(classification_report(y_test, xgb_predictions))

print (metrics.accuracy_score(y_test,xgb_predictions))
```

```
[[1169  131]
 [ 236  222]]
             precision    recall  f1-score   support

          0       0.83      0.90      0.86      1300
          1       0.63      0.48      0.55       458

avg / total       0.78      0.79      0.78      1758

0.7912400455062572
```

Deep Learning

- the type of network that work well on this kind of problems is a simple stack of fully connected Dense layers with relu activations.
- The argument being passed to each Dense layer is the number of hidden units of layer. Here it is 64.



```
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.utils import to_categorical

# Convert the target to categorical: target
target = to_categorical(new_data.Churn_Yes)
n_cols = X.shape[1]

A_train, A_test, b_train, b_test = train_test_split(X, target, test_size = 0.25, random_state=42)
model = Sequential()

# Add the first layer
model.add(Dense(64,activation='relu',input_shape=(n_cols,)))

model.add(Dense(64,activation='relu'))
# Add the output layer

model.add(Dense(64,activation='relu'))
model.add(Dense(2,activation='softmax'))

# Compile the model
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

# Fit the model
model.fit(A_train,b_train,epochs=100)
```

As the point of interest is recall, from above all the models have very less recall value for Customer Churn (Value 1) - 0.48 to 0.51, but recall for not churned if from 0.79 to 0.94. Considering Precision, the values are higher for classifier label 0 than 1. This leads to the doubt about the distribution of rows for both classifiers.

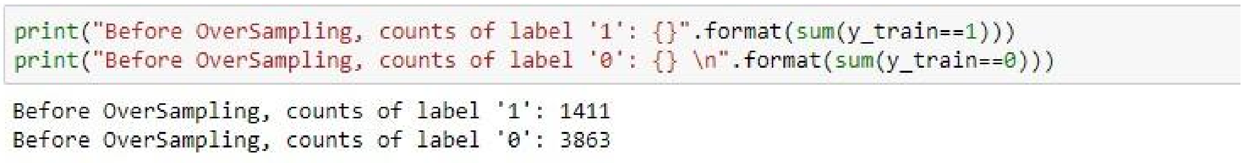
SMOTE

Imbalanced data sets affect the performance and predictions of a model. From the code below the number of customers churn (1) is in very less number compared to not churned. There can be three types of solutions for these problems, they are

- Over-sample the minority class.
- Under-sample the majority class. □

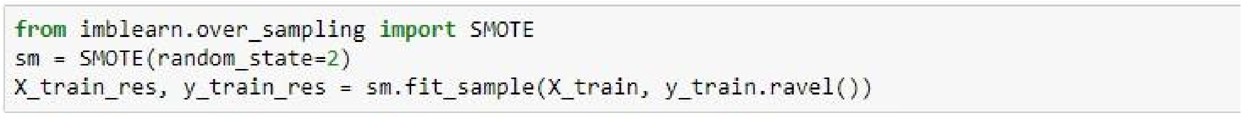
- Synthesize new minority classes.

SMOTE (Synthetic Minority Over-sampling Technique) is the process of creating a new minority classes from the datasets.

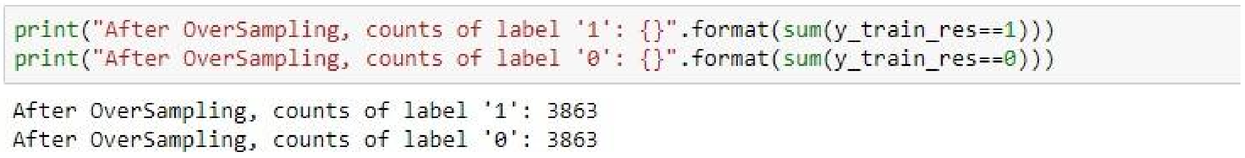


```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))
```

```
Before OverSampling, counts of label '1': 1411
Before OverSampling, counts of label '0': 3863
```



```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())
```

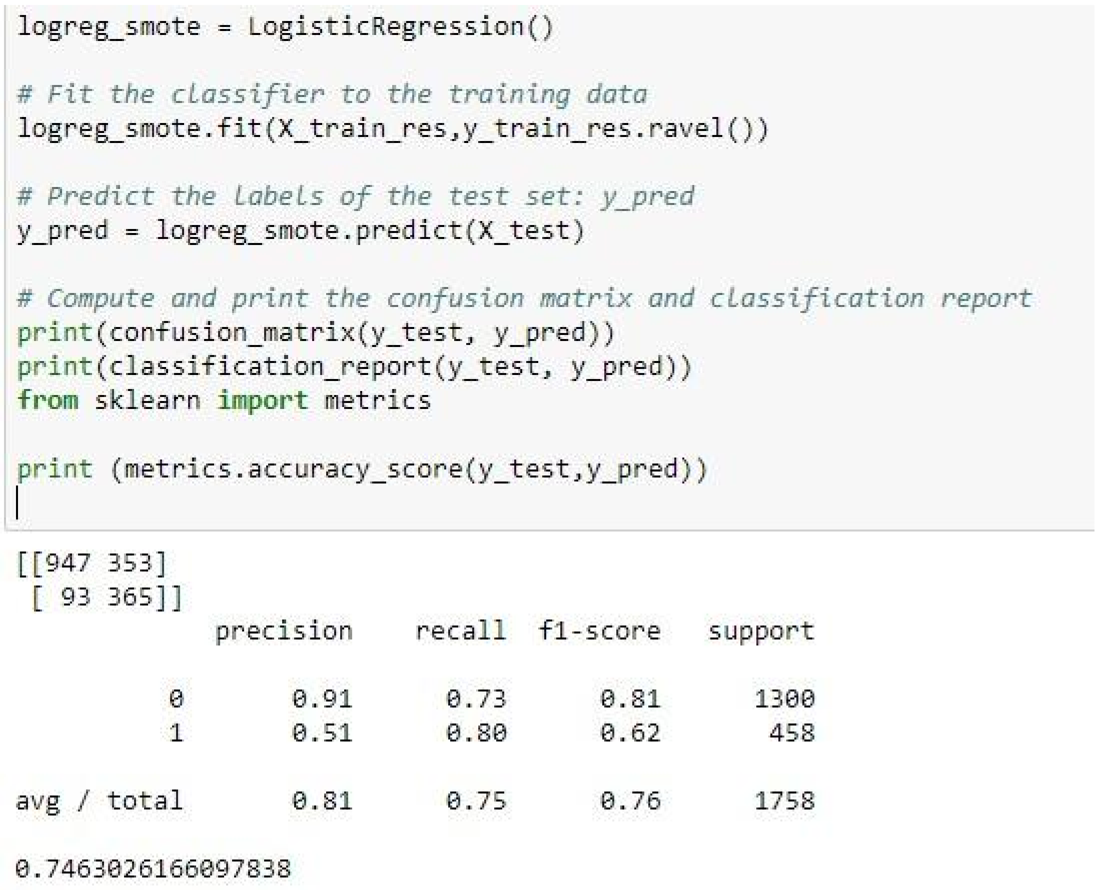


```
print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
```

```
After OverSampling, counts of label '1': 3863
After OverSampling, counts of label '0': 3863
```

After applying SMOTE the distribution of both are classifiers are synthesized equally. Now we can apply all the algorithms to see any considerable differences in model performance and predictions.

Logistic regression after SMOTE



```
logreg_smote = LogisticRegression()

# Fit the classifier to the training data
logreg_smote.fit(X_train_res,y_train_res.ravel())

# Predict the labels of the test set: y_pred
y_pred = logreg_smote.predict(X_test)

# Compute and print the confusion matrix and classification report
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
from sklearn import metrics

print (metrics.accuracy_score(y_test,y_pred))
```

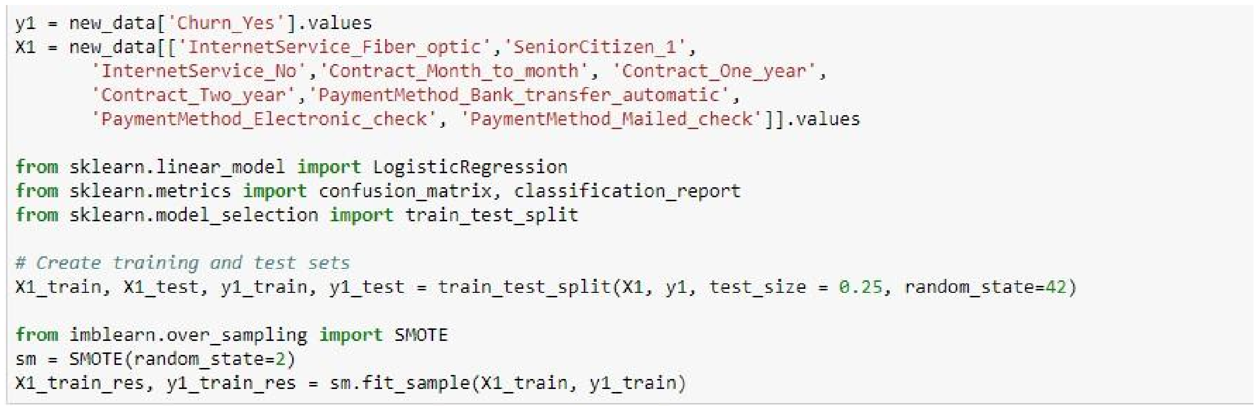
```
[[947 353]
 [ 93 365]]
             precision    recall  f1-score   support

          0       0.91      0.73      0.81      1300
          1       0.51      0.80      0.62       458

avg / total       0.81      0.75      0.76      1758

0.7463026166097838
```

		Before SMOTE		After SMOTE	
		Precision	Recall	Precision	Recall
Logistic	0	0.84	0.89	0.91	0.73
	1	0.61	0.51	0.51	0.8
Decision	0	0.83	0.79	0.83	0.78
	1	0.47	0.53	0.47	0.54
Random	0	0.82	0.89	0.82	0.88
	1	0.59	0.44	0.57	0.44
svm	0	0.78	0.94	0.81	0.84
	1	0.68	0.25	0.49	0.44
XG Boost	0	0.83	0.9	0.85	0.85
	1	0.63	0.48	0.58	0.58

- Decision Trees and Random Forest didn't show any considerable difference in recall and precision values before and after applying SMOTE
- SVM and XGBoost though recall value for classifier 1 increased but the precision for the same met with a decrease in value.
- Comparing other models Logistic Regression gives best value for recall and precision value with little drop in precision of Classifier 1

Classification with Important Features

Considering only the important features from Feature Engineering analysis, the features are Tenure, MonthlyCharges, SeniorCitizen_1, PhoneService_Yes, InternetService_Fiber_optic, InternetService_No, Contract_Month_to_month, Contract_One_year, Contract_Two_year, PaymentMethod_Electronic_check



```
y1 = new_data['Churn_Yes'].values
X1 = new_data[['InternetService_Fiber_optic','SeniorCitizen_1',
       'InternetService_No','Contract_Month_to_month', 'Contract_One_year',
       'Contract_Two_year','PaymentMethod_Bank_transfer_automatic',
       'PaymentMethod_Electronic_check', 'PaymentMethod_Mailed_check']].values

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

# Create training and test sets
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 0.25, random_state=42)

from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=2)
X1_train_res, y1_train_res = sm.fit_sample(X1_train, y1_train)
```

		Before SMOTE		After SMOTE		SMOTE less Features	
		Precision	Recall	Precision	Recall	Precision	Recall
Logistic	0	0.84	0.89	0.91	0.73	0.91	0.66
	1	0.61	0.51	0.51	0.8	0.46	0.81
Decision	0	0.83	0.79	0.83	0.78	NA	NA
	1	0.47	0.53	0.47	0.54	NA	NA
Random	0	0.82	0.89	0.82	0.88	0.9	0.68
	1	0.59	0.44	0.57	0.44	0.47	0.79
svm	0	0.78	0.94	0.81	0.84	0.91	0.64
	1	0.68	0.25	0.49	0.44	0.45	0.83

XG Boost	0	0.83	0.9	0.85	0.85	0.9	69
	1	0.63	0.48	0.58	0.58	0.47	0.78

As only a smaller number of features selected precision and recall values suffered a considerable amount. Recall for classifier 0 and Precision for 1 suffers the most

From the above analysis Logistic Regression after SMOTE application performs best for this problem implying that not always complex algorithms are essential for better performance.

Conclusion

Always it is not the accuracy is only important. Based on business question and problem we try to solve other measures are equally important. As for the problems like Customer Churn, Credit defaulters and Spam emails mainly based on recall value. Same way there is not any golden rule that complex algorithms gives best performance and accuracy. Each problem is different so gauging against all the algorithms will be a better choice.

For future analysis, still improvement on Feature Engineering and fine tuning the parameters will give better results.