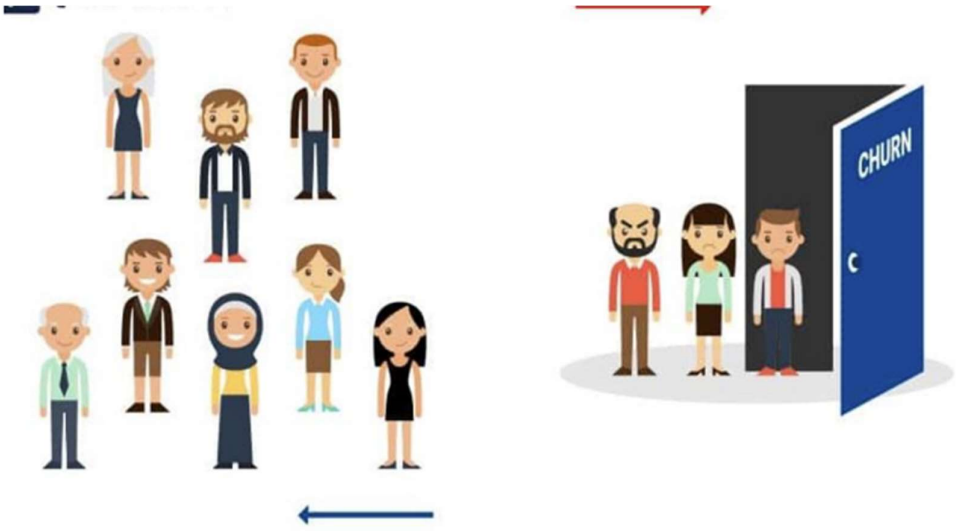
Customer Churn Analysis in the Telecom Industry

Mehul Bisht

1. Problem Statement

The telecom industry is flourishing these days. New telecom companies are trying to stabilise themselves in the market. This poses a threat to the existing/local companies of losing a customer. It is important to come up with a solution to identify why customers leave their services, thereby forcing the companies to prevent this from happening. Therefore, we should build such a system which calculates the number of people who churn out or leave their service providers, thereby helping companies to identify opportunities to upsell or cross-sell to existing customers.



2. Market/Customer/Business Need Assessment

Market Need: There is a significant market need for churn analysis in the telecom industry to help companies understand why customers are leaving and what they can do to retain them. Customer Need: The Churn analysis can help telecom companies understand their

customer's requirements and needs and identify areas where they need to improve their services to meet those needs.

Business Need: Churn analysis can help companies identify the reasons why customers are leaving and take proactive measures to prevent churn. By understanding their customers' needs and preferences, companies can also identify opportunities to upsell and cross-sell to existing customers, increasing their revenue.

3. Target Specifications and Characterization

The proposed system will provide the service providers with the number of people likely to churn based on certain factors like age, postpaid, prepaid charges, etc for a local service provider.

Talking about the characterization could be divided into:

- Data Sources
- Key Performance Indicators
- Predictive Modeling
- Segmentation
- Actionable Insights
- Continuous Improvement

4. External Search

The resources that I have used to predict the number of people to churn in a local service provider:

- https://www.analyticsvidhya.com/blog/2023/01/understanding-churn-analytics-in-telecommunications-company/
- https://zyabkina.com/churn-analysis-ultimate-guide-to-customer-attrition/
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9051585/
- https://www.netsuite.com/portal/resource/articles/human-resources/customer-churn-analysis.shtml

5. Benchmarking alternate products

Global service providers have been using various ML analyses to perform Customer churn analysis, which uses this information to find out the number of to who drop out of the service provider. But this technique would also be beneficial when applied to small/ local service providers.

6. Applicable Patents

- U.S. Patent No. 9,225,981: This patent, entitled "Method and apparatus for predicting customer churn in a telecommunications network," describes a system for predicting customer churn by analysing data from multiple sources, including call logs, billing information, and customer demographics.

- U.S. Patent No. 9,332,235: This patent, entitled "Predicting customer churn in a communication network," describes a system for predicting customer churn by analysing call detail records (CDRs) and network performance data.

- U.S. Patent No. 9,633,258: This patent, entitled "Churn prediction in communication networks using behavioural analysis," describes a system for predicting customer churn by analysing customer behaviour, such as the frequency and duration of calls and data usage patterns.

7. Applicable Regulations

While talking about the various regulations that we could encounter:

- Data Privacy: Many countries have laws and regulations governing customer data collection, storage, and use.

- Network security regulations
- Consumer protection regulations

8. Applicable Constraints

- Several constraints we will come across:
- Data Quality
- Customer Behaviour
- Resource Constraint
- Regulatory Constraints

9. Business Model

Since the proposed idea can be used for large/global service providers, we can extend it to small/medium service providers. Therefore, it can be a great business model compared to small businesses. Every local service provider would wants customer retention would opt for such an analysis. want. The emergence of every small service provider is thus a great business opportunity for the service provided by this analysis.

10. Concept Generation

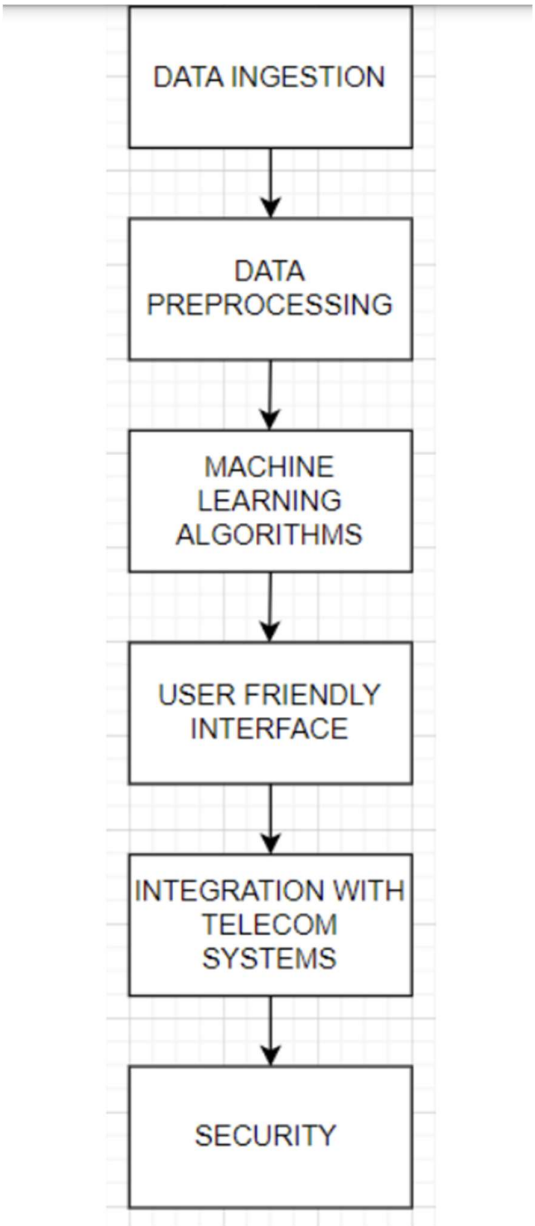
For this product, the use of machine learning models would greatly improve our product. We would clean the data from the dataset provided by the local service provider and apply pre-processing to understand the entire dataset and use visualization techniques which would help it understand better. Later we apply various machine learning algorithms to predict and choose the best machine learning algorithm (in terms of highest accuracy).

11. Concept Development

To develop our concept, we could use an API (here we use Flask) and for a framework, we could use Django. Later we could deploy it on any of the cloud services.

12. Final Product Prototype (abstract) with Schematic Diagram

Our product prototype would include:



- **Data ingestion:** Responsible for ingesting data from various sources/factors.
- **Data preprocessing:** Clean and preprocess the data to ensure it is ready for analysis.
- **Machine learning models:** Various machine learning algorithms that would analyse the data and predict which customers are most likely to churn.
- **User-Friendly Interface:** It would provide a user-friendly interface for telecom companies to monitor churn rates and track the effectiveness of retention strategies along with visualization graphs.
- Integration with telecom systems
- **Security and compliance:** The platform would need to comply with data privacy regulations and provide secure data storage and processing.

We can use Flask to view the results on the local host about the number of customers who would churn.

13. Product details:

13.1 How does it work?

We use machine learning to predict the people who would churn out of the local service provider and further build a user-friendly interface which would take in the features (age, postpaid, prepaid, etc) and can view this on the interface(front), thereby developing an app for the local service providers helping them.

13.2 Data Sources

We could ask the service providers for a dataset with the required set of features (nothing confidential) which would help in predicting the number of people to churn out.

13.3 Algorithms, frameworks, software etc. needed

We could use various machine learning algorithms like Decision Trees, Naïve Bayes, and Random Forests to predict the number of people to churn. We could use Flask as an API and a user interface to view the details (whether the person based on the feature would churn or not). We could later deploy this on any cloud service.

13.4 Team required to develop:

- Data Collection and preprocessing team
- Frontend Team
- Backend Team
- Deployment Team

13.5 Costing

Yet to be decided

14. Code Implementation/Validation on Small Scale

14.1 Some Basic Visualizations on Real World or Augmented Data

For this part, I have taken a dataset from Kaggle (since it was difficult to get the dataset from a local service provider).

14.2 Simple EDA

Below are the various visualizations that I have observed.

```
In [4]: dt.head()
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

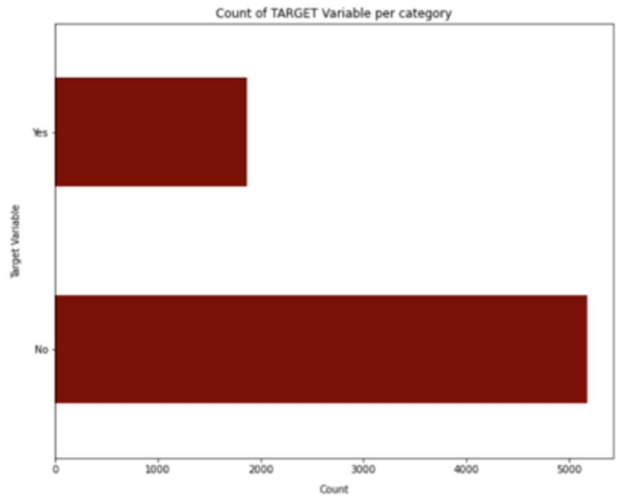
5 rows × 21 columns

```
In [5]: dt.shape
Out[5]: (7043, 21)
```

```
In [6]: dt.columns.values
Out[6]: array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

```
In [7]: dt.dtypes
Out[7]: customerID          object
        gender              object
        SeniorCitizen        int64
        Partner             object
        Dependents          object
        tenure               int64
        PhoneService        object
        MultipleLines       object
        InternetService     object
        OnlineSecurity      object
        OnlineBackup        object
        DeviceProtection    object
        TechSupport         object
        StreamingTV         object
        StreamingMovies     object
        Contract            object
        PaperlessBilling    object
        PaymentMethod       object
        MonthlyCharges     float64
        TotalCharges        object
        Churn               object
        dtype: object
```

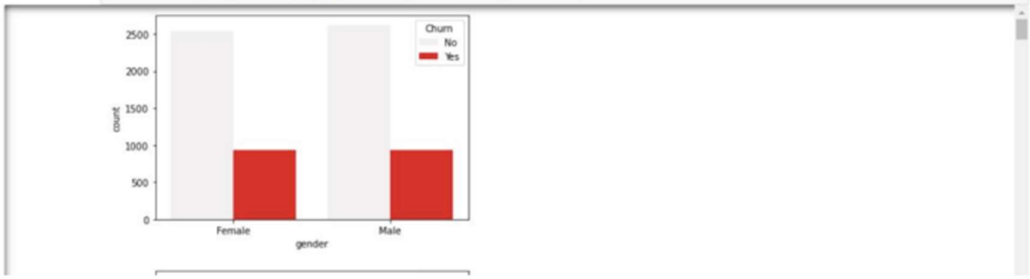
```
plt.ylabel("Target Variable", labelpad=10)
plt.title("Count of TARGET Variable per category", y=1.00);
```

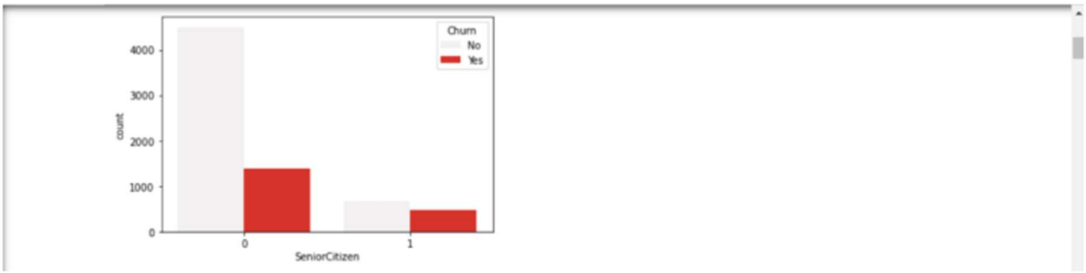


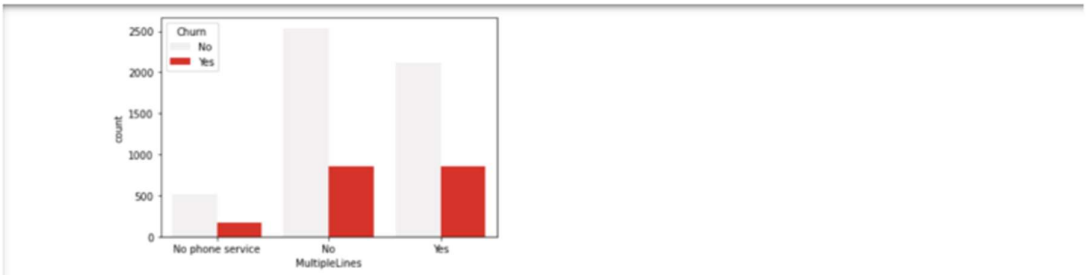
In [13]:
```
#to find the number of missiong values
missing = pd.DataFrame((dt.isnull().sum())*100/dt.shape[0]).reset_index()
plt.figure(figsize=(16,5))
ax = sns.pointplot('index',0,data=missing, color="maroon")
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. Fro
m version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
```

In [22]:
```
#Data Exploration
#Univariate Analysis
for i, predictor in enumerate(dt_new.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=dt_new, x=predictor, hue='Churn', color="r")
```

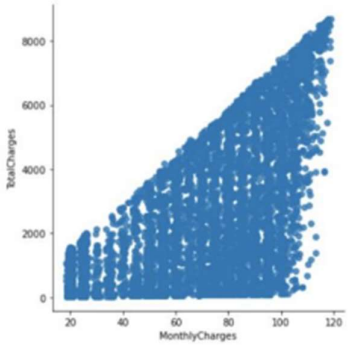






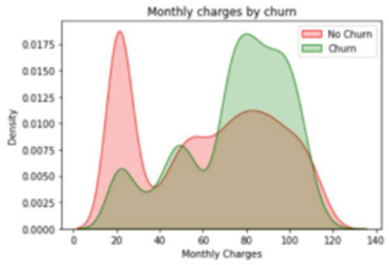
In [28]:
```
sns.lmplot(data=dt_new_dummies, x='MonthlyCharges', y='TotalCharges', fit_reg=False)
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x26897c75f70>



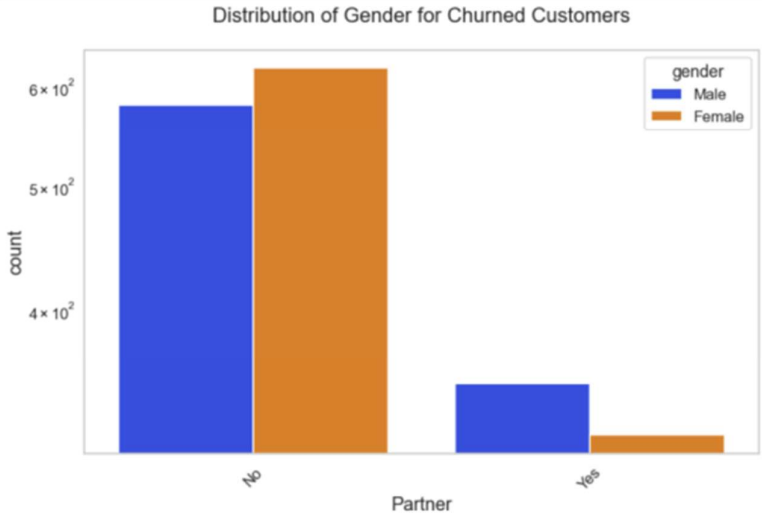
In [30]:
```
#Churn by Monthly Charges and Total Charges
Mth = sns.kdeplot(dt_new_dummies.MonthlyCharges[(dt_new_dummies["Churn"] == 0) ],
                  color="Red", shade = True)
Mth = sns.kdeplot(dt_new_dummies.MonthlyCharges[(dt_new_dummies["Churn"] == 1) ],
                  ax =Mth, color="Green", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')
Mth.set_xlabel('Monthly Charges')
Mth.set_title('Monthly charges by churn')
##CHURN IS HIGH WHEN MONTHLY CHARGES ARE HIGH
```

Out[30]: Text(0.5, 1.0, 'Monthly charges by churn')



Distribution of Gender for Churned Customers

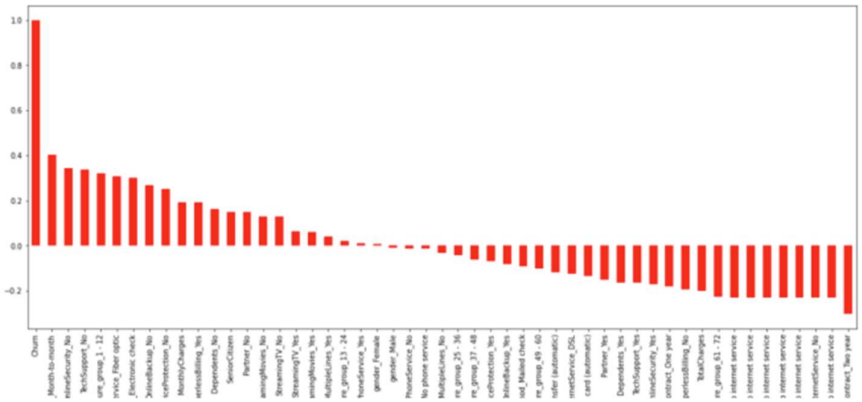
In [42]:
```
uniplot(new_df1_target1,col='Partner',title='Distribution of Gender for Churned Customers',hue='gender')
```



Correlation of all features with churn

In [33]:
```
#Build a corelation of all predictors with 'Churn'
plt.figure(figsize=(20,8))
dt_new_dummies.corr()['Churn'].sort_values(ascending = False).plot(kind='bar', color="red")
```

Out[33]: <AxesSubplot:>



14.3 ML Modelling

In [1]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in d:\anaconda\lib\site-packages (0.10.1)
Requirement already satisfied: scipy>=1.3.2 in d:\anaconda\lib\site-packages (from imbalanced-learn) (1.4.1)
Requirement already satisfied: scikit-learn>=1.0.2 in d:\anaconda\lib\site-packages (from imbalanced-learn) (1.2.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in d:\anaconda\lib\site-packages (from imbalanced-learn) (2.1.0)
Requirement already satisfied: joblib>=1.1.1 in d:\anaconda\lib\site-packages (from imbalanced-learn) (1.2.0)
Requirement already satisfied: numpy>=1.17.3 in d:\anaconda\lib\site-packages (from imbalanced-learn) (1.18.5)
```

In [2]:
```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN
```

In [3]:
```
df = pd.read_csv('telecom_churn.csv')
```

Train Test Split

In [10]:
```
y=df['Churn']
y
```

Out[10]:
```
0       0
1       0
2       1
3       0
4       1
       ..
7027    0
7028    0
7029    0
7030    1
7031    0
Name: Churn, Length: 7032, dtype: int64
```

In [11]:
```
#test-train
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Decision Tree Classifier

In [12]:
```
#decision tree classifier
model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [13]:
```
model_dt.fit(x_train,y_train)
```

Out[13]:
```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]:
```
y_pred=model_dt.predict(x_test)
y_pred
```

Out[14]:
```
array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [15]:
```
model_dt.score(x_test,y_test)
```

Out[15]:
```
0.783226723525231
```

In [16]:
```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1043
           1       0.60      0.49      0.54       364

    accuracy                           0.78      1407
   macro avg       0.72      0.69      0.70      1407
weighted avg       0.77      0.78      0.78      1407
```

Use of SMOTEENN (since accuracy is less)

In [17]:
```
#since the accuracy is less, we call SMOTEENN
```

In [19]:
```
from imblearn.over_sampling import SMOTE
```

In [22]:
```
sm = SMOTEENN()
X_resampled, y_resampled = sm.fit_resample(x,y)
```

In [23]:
```
xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)
```

In [24]:
```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [25]:
```
model_dt_smote.fit(xr_train,yr_train)
yr_predict = model_dt_smote.predict(xr_test)
model_score_r = model_dt_smote.score(xr_test, yr_test)
print(model_score_r)
print(metrics.classification_report(yr_test, yr_predict))
```

```
0.9414261460101867
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       533
           1       0.94      0.96      0.95       645

    accuracy                           0.94      1178
   macro avg       0.94      0.94      0.94      1178
weighted avg       0.94      0.94      0.94      1178
```

Random Forest Classifier

In [27]:
```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
```

In [28]:
```
model_rf=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [29]:
```
model_rf.fit(x_train,y_train)
```

Out[29]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
y_pred=model_rf.predict(x_test)
```

In [31]:
```
model_rf.score(x_test,y_test)
```

Out[31]:
```
0.7967306325515281
```

In [32]:
```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      1043
           1       0.65      0.46      0.54       364

    accuracy                           0.80      1407
   macro avg       0.74      0.69      0.70      1407
weighted avg       0.78      0.80      0.78      1407
```

In [34]:
```
sm = SMOTEENN()
X_resampled1, y_resampled1 = sm.fit_resample(x,y)
```

In [35]:
```
xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)
```

In [36]:
```
model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [37]:
```
model_rf_smote.fit(xr_train1,yr_train1)
```

Out[37]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
yr_predict1 = model_rf_smote.predict(xr_test1)
```

In [39]:
```
model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)
```

In [40]:
```
print(model_score_r1)
print(metrics.classification_report(yr_test1, yr_predict1))
```

```
0.9465977605512489
```

Pickling the model

```
In [61]: #Pickling the model
         import pickle

In [62]: filename = 'model.sav'

In [63]: pickle.dump(model_rf_smote, open(filename, 'wb'))

In [64]: load_model = pickle.load(open(filename, 'rb'))

In [65]: model_score_r1 = load_model.score(xr_test1, yr_test1)

In [66]: model_score_r1
Out[66]: 0.9465977605512489
```

14.3 GitHub link to the code Implementation

Not implemented it completely (flask and deployment part is remaining).

15. Conclusion

There are many times when we might not be able to predict the number of customers who churn out, but with this idea (a web app in this case) we can predict the number of customers who would churn thereby calculating the loss caused to the company which would motivate the small-scale service providers to bring out ways of customer retention maximizing the profit.