TABLE OF CONTENTS

Sr.No	Topic	Page No.
1	Problem Description	3
2	Data Description	3
3	Data Pre-Processing	5
4	Data – Exploration (EDA)	8
5	Model Building	15
6	Model Evaluation	20
7	Limitations	22
8	Closing Reflections	23
9	References	24

CUSTOMER CHURN PREDICTION ANALYSIS

PROBLEM DESCRIPTION

Customer churn is the percentage of customers that stops using a company's product or service during a certain time frame. It is an important metric to track as lost customers equals lost revenue. If a company loses enough customers, it can have a serious impact on its revenue. Another reason that is critical to improve churn is that, it is generally more expensive to find new customers than it is to keep existing ones. So, companies that lose customers aren't just losing the revenue from those customers, but also stuck with high cost finding new customers. Usually, there is no single reason, but a combination of events that somehow culminated in customer displeasure. In this project, the goal is to identify customer churn, that is, customers most likely to cancel subscription to a fictitious Telecom company.

The main goal is to develop a machine learning model capable to predict customer churn based on the customer's data available. That is a supervised classification problem and Machine Learning algorithms will be used for the development of predictive models and evaluation of accuracy and performance. It seeks to find the most appropriate model for the business.

DATA DESCRIPTION

The dataset for this exercise relates features of account and usage for churn and non-churn clients. The Telecom dataset was obtained from data.world public repository.

This dataset contains a total of 12,892 customers and 22 attributes, coming from personal characteristics, services signatures, and contract details. Out of the entries, 11,069 are active customers and 1,823 are churned, which demonstrates that the dataset is highly unbalanced. The target variable for this assessment is going to be the feature Churn. Our telecom Customer Churn dataset contains 18 Numerical Variable, 4 Categorical Variable.

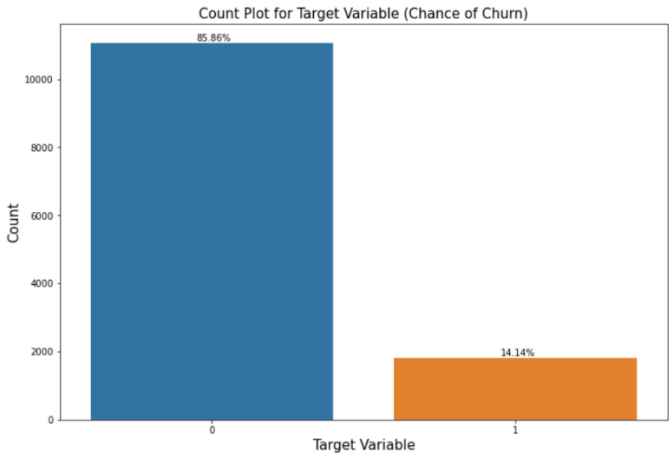
FEATURE NAME	VALUE AND DESCRIPTION	FEATURE TYPE
Record id	Primary key of the record.	Numeric
State	State information of the Customers.	Categoric
Account length	Age of account in months.	Numeric
Area code	Area code.	Numeric
International plan	Indicates whether the customer has an international calling plan or not.	Categoric
Voice mail plan	Indicates whether the customer has a voice mail plan or not.	Categoric
No. vmail message	Number of VM messages customer currently has on the server.	Numeric

Total day minutes	Customers total usage of day minutes in plan.	Numeric
Total day calls	Total number of calls customer has made during the day.	Numeric
Total day charge	How much the customer has been charged for day minutes.	Numeric
Total eve minutes	Customers total usage of evening minutes in plan.	Numeric
Total eve calls	Total number of calls customer has made during the evening.	Numeric
Total eve charge	How much the customer has been charged for evening minutes.	Numeric
Total night minutes	Customers total usage of night minutes in plan.	Numeric
Total night calls	Total number of calls customer has made during the night.	Numeric
Total night charge	How much the customer has been charged for night minutes.	Numeric
Total intl minutes	Total international minutes.	Numeric
Total intl calls	Total international calls.	Numeric
Total intl charge	Total international charges.	Numeric
Number customer service calls	How many times the customer has called the IVR system.	Numeric
Churn	Customer has churned.	Categoric
Customer id	Enterprise ID of the customer.	Numeric

TARGET VARIABLE

The target variable for our analysis is the feature 'Churn' which has the information about whether a customer has churned or not. The target column is of categoric type.

The distribution of the Target attribute ('Churn') is as follows:



From the plot, we can observe that the dataset is highly imbalanced. We will overcome this by SMOTE Technique.

DATA PRE-PROCESSING

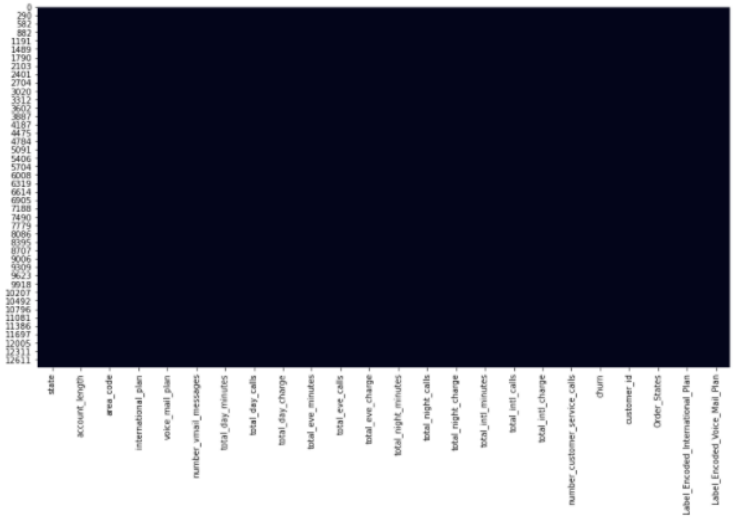
Data Cleaning

Data cleaning is commonly defined as the process of detecting and correcting corrupt or inaccurate records from a dataset, table, or database. Data quality is an important component in any data mining efforts. For this reason, many data scientists spend from 50% to 80% of their time preparing and cleaning their data before it can be mined for insights. In our project, we have dealt with data quality problems such as missing data, abnormal data (outliers), irrelevant features, scaling and transformation. We will also address the imbalance in the class variable using SMOTE.

Missing Value Treatment

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset. It can bias the results of the machine learning models and/or reduce the accuracy of the model. It can lead to wrong prediction or classification. So, we are checking the dataset for the presence of missing / null values.

```
# Visualization of no missing values using heatmap
plt.rcParams["figure.figsize"]=[15,8]
sns.heatmap(df.isnull(), cbar = False)
plt.show()
```

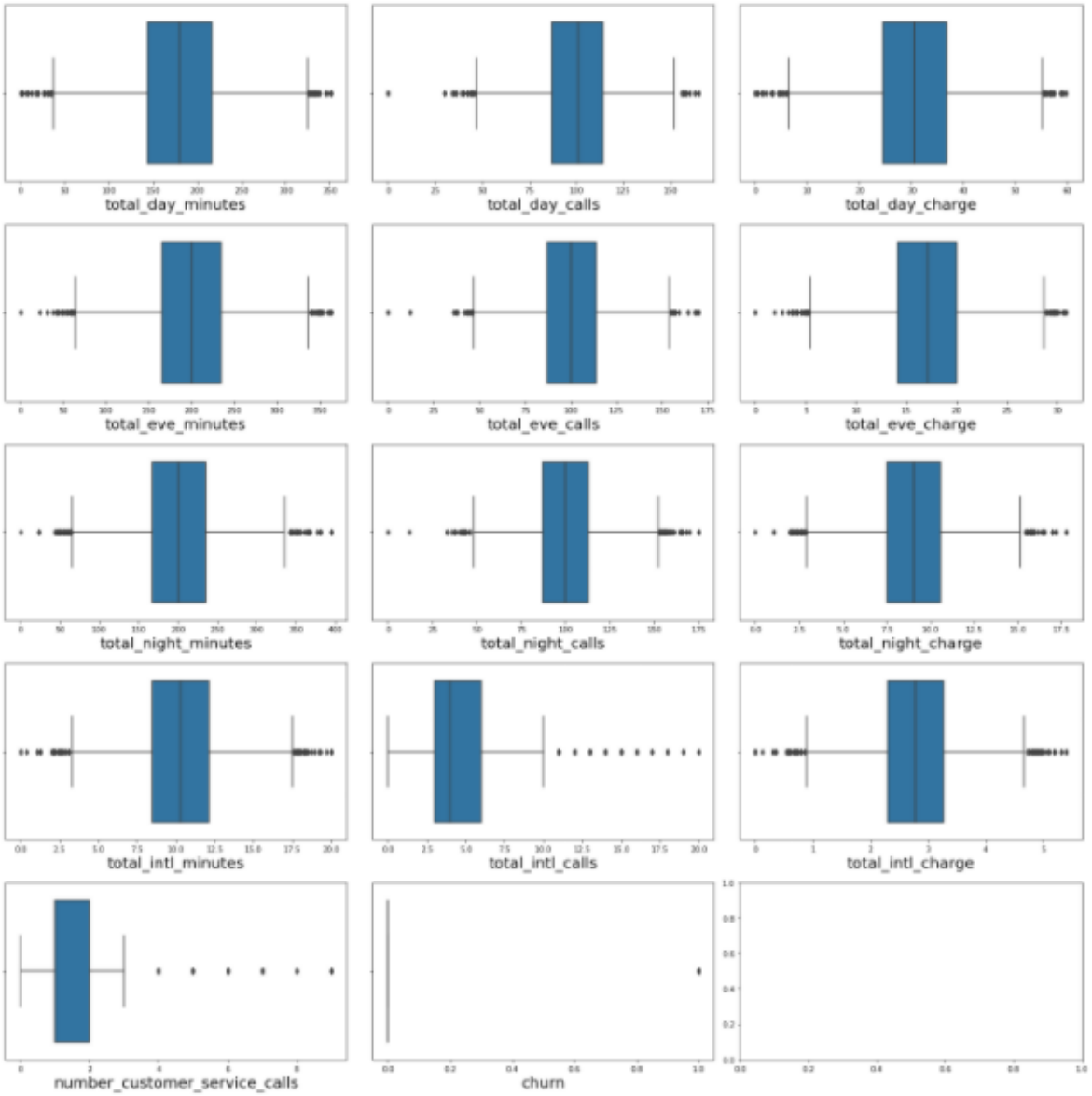


From the heatmap, we can observe that there are no missing / null values in the dataset.

Outlier Treatment

Outliers are extreme values that fall a long way outside of the other observations. The machine learning model are generally sensitive towards the outliers. Hence, we need to handle outliers very carefully. We can check the outliers using various techniques.

Here, we are plotting boxplot for all the numerical variables to find out the number of outliers present in the dataset. Below is the python approach, we used to determine the outliers.



From the plot, we can observe the presence of outliers in the dataset. But due to the nature of the dataset, we are creating two datasets, one by removing outliers and another with outliers. We are building model for both the dataset, after comparing the result, we can make a conclusion on which dataset gives better predictions.

Dropping Insignificant Features

In the dataset, the features, ‘record_id’ and ‘customer_id’ gives the same information about customer, so we will drop ‘record_id’ and retain ‘customer_id’ for initial analysis.

Statistical Analysis

We have performed chi squared test on the categoric features ‘area_code’, ‘International_plan’, ‘voice_mail_plan’, ‘state’ and ‘customer_id’ each with the target variable ‘Churn’. It is observed that ‘Customer_id’ has no relationship with the target feature ‘churn’ as the p-value is greater than 0.05. Hence, we drop the feature ‘customer_id’ from the dataset for further analysis.

Feature Engineering

i.

```
area = df['area_code']

def new_area(area):
    if area == 510:
        return 'Area 1'
    if area == 408:
        return 'Area 2'
    if area == 415:
        return 'Area 3'

df['area_code'] = df['area_code'].apply(new_area)
df
```

We are performing Feature Engineering on 'area code' column in order to turn the variable to understandable information. Hence, we are assigning the classes of the column to 'Area 1', 'Area 2' and 'Area 3'.

ii.

```
df_fea['Call_Minutes'] = df_fea['total_day_minutes']+df_fea['total_eve_minutes']+df_fea['total_night_minutes']
+df_fea['total_intl_minutes']

df_fea['Call_Count']= df_fea['total_day_calls']+df_fea['total_eve_calls']+df_fea['total_night_calls']+df_fea['total_intl_calls']

df_fea['Total_Charge']= df_fea['total_day_charge']+df_fea['total_eve_charge']+df_fea['total_night_charge']
+df_fea['total_intl_charge']
```

```
df_fea.head()
```

_code	international_plan	voice_mail_plan	number_vmail_messages	number_customer_service_calls	churn	customer_id	Call_Minutes	Call_Count	Total_Charge
Area 1	no	no	0	3	0	23383607.0	529.4	272	43.54
Area 1	no	no	0	0	0	22550362.0	572.1	313	65.63
Area 2	no	yes	29	1	0	59063354.0	846.0	333	87.48
Area 3	no	no	0	1	0	25464504.0	579.5	295	61.77
Area 3	no	no	0	2	0	691824.0	533.6	290	55.38

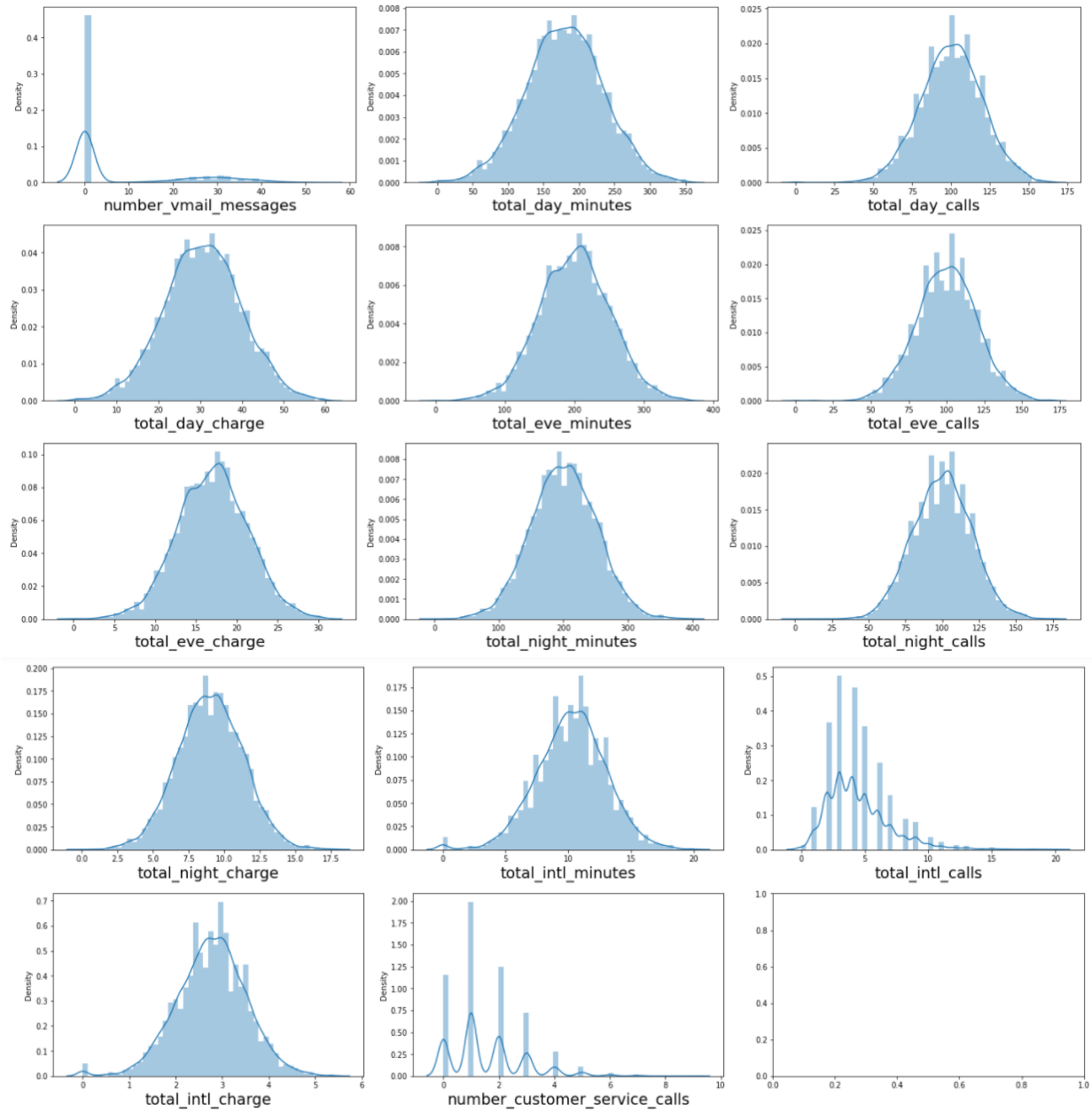
In the dataset, we can observe that few features have some interaction between them. So we will be creating new features from the data set which have interaction between them.

- Variables 'total_day_minutes', 'total_eve_minutes', 'total_night_minutes', 'total_intl_minutes' are representing Call duration in minutes of call made on different timings. So, we can sum all the variables and make it a single variable 'Call_Minutes'.
- Variables 'total_day_calls', 'total_eve_calls', 'total_night_calls', 'total_intl_calls' are representing total number of calls made on different timings. So, we can sum all the variables and make it a single variable 'Call_Count'.
- Variables 'total_day_charge', 'total_eve_charge', 'total_night_charge', 'total_intl_charge' are representing the amount Charged on customer for the call made on different timings. So, we can sum all the variables and make it as single variable 'Total_Charge'.
- By adding the features having interaction between them, we have created three new features. We will create a new dataset with the feature engineered columns and separately build a model and evaluate its performance.

DATA EXPLORATION (EDA)

Univariate Analysis

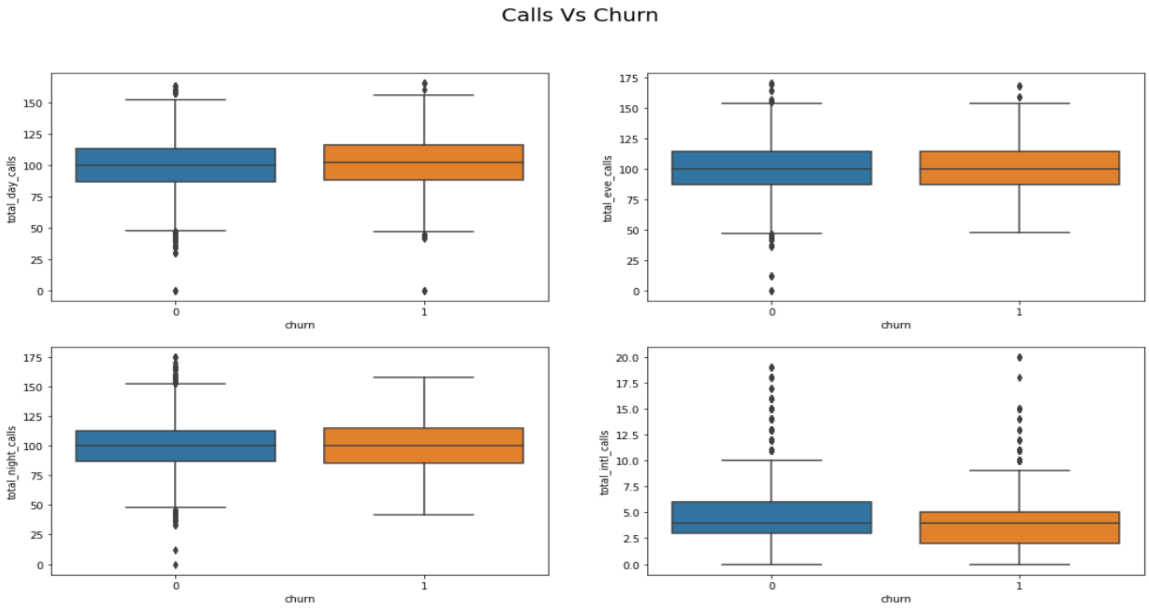
Distribution plot of the numeric features of the dataset.



From the plot, we can observe that all the columns are almost normally distributed except the features ‘number_vmail_messages’, ‘total_intl_calls’ and ‘number_customer_service_calls’ are slightly skewed.

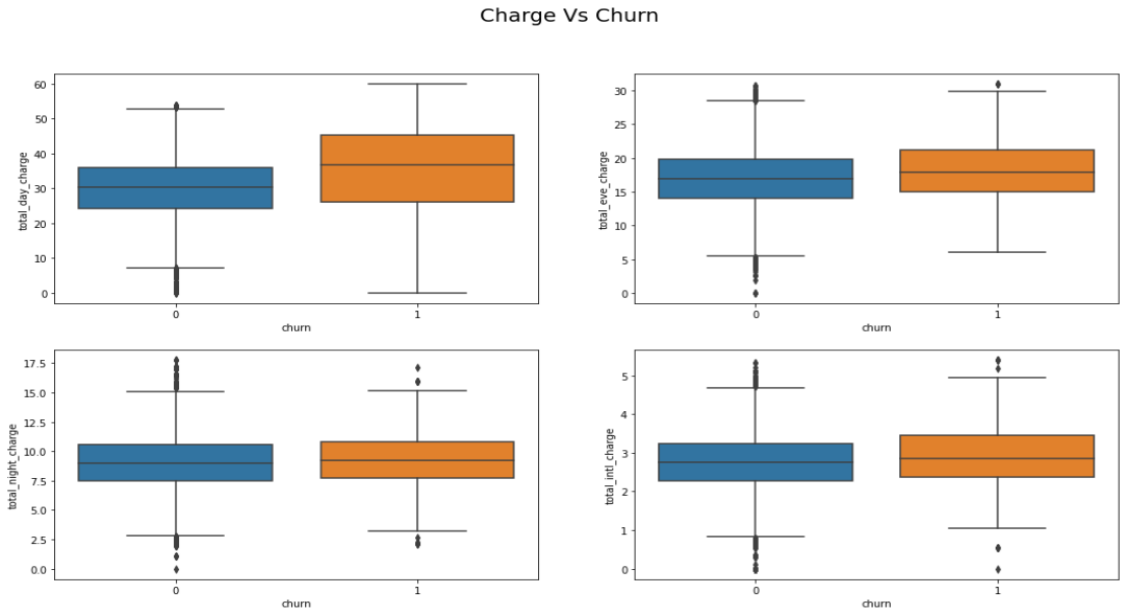
Bivariate Analysis

1 – Box plot between calls made during various time of the day & international calls and churn rate.



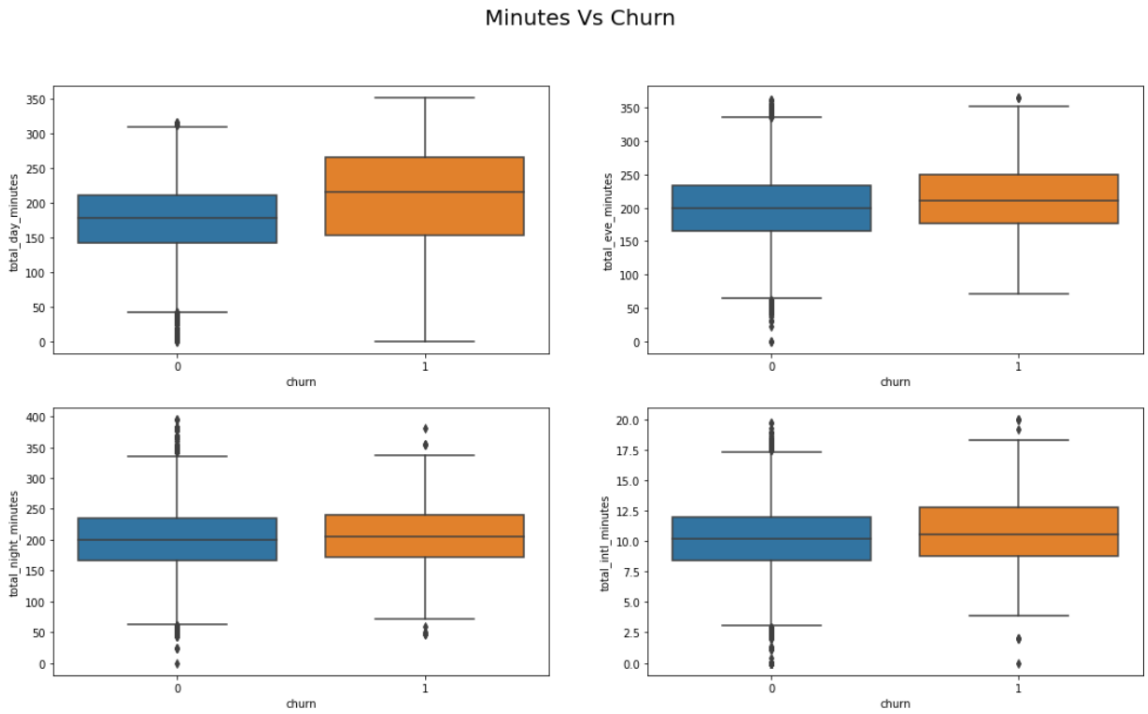
From the plot, we can infer that churn is not affected by calls made on different times of the day by the customers and average calls made by customers who churn and do not churn is more or less the same.

2 – Box plot between total charges for calls made at various time of the day & international calls made and churn rate.



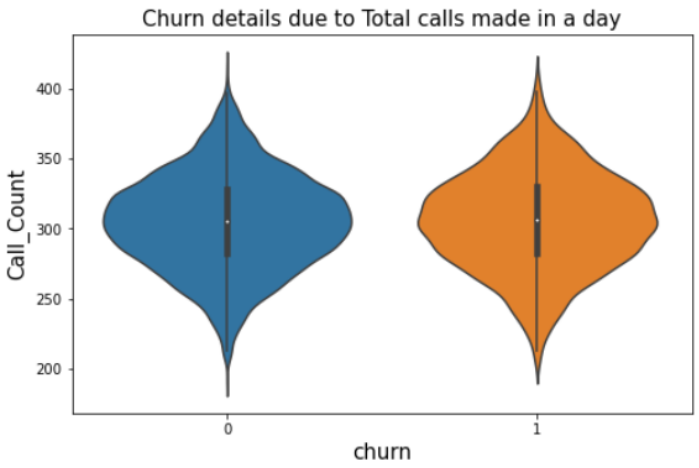
From the plot, we can infer that, churn is not affected by the charge spent on call during evening, night and international because average charge spent on calls by customers who churn and do not churn is more or less the same. But churn is affected by charge during the Daytime calls, since the average charge on customer who churn is slightly higher than customer who do not churn.

3 – Box plot between total call duration in minutes made at various time of the day & international calls made and churn rate.



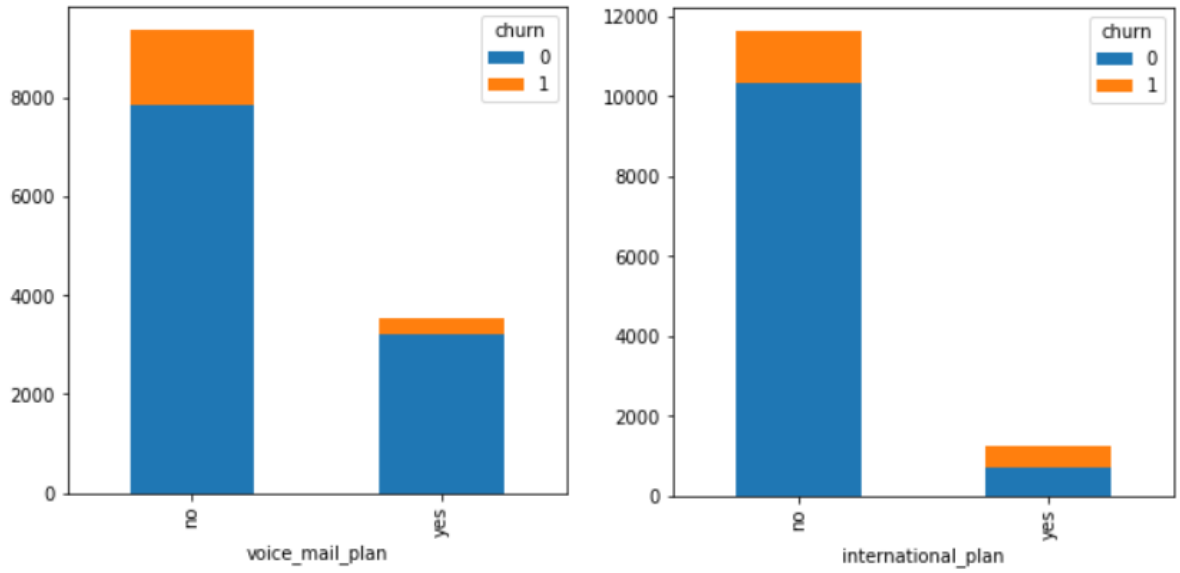
From the plot, we can infer that churn is not affected by amount of time spent on call during evening, night and international because Average minutes spent on calls by customers who churn and do not churn is more or less the same. But churn is affected by amount of time spent on call during Daytime, since the average time spent on customer who churn is slightly higher than customer who do not churn.

4 – Violin plot between call count and churn rate.



From the plot, we can infer that, churn is not affected by total calls made by the customers as average calls made by customers who churn and do not churn is more or less the same.

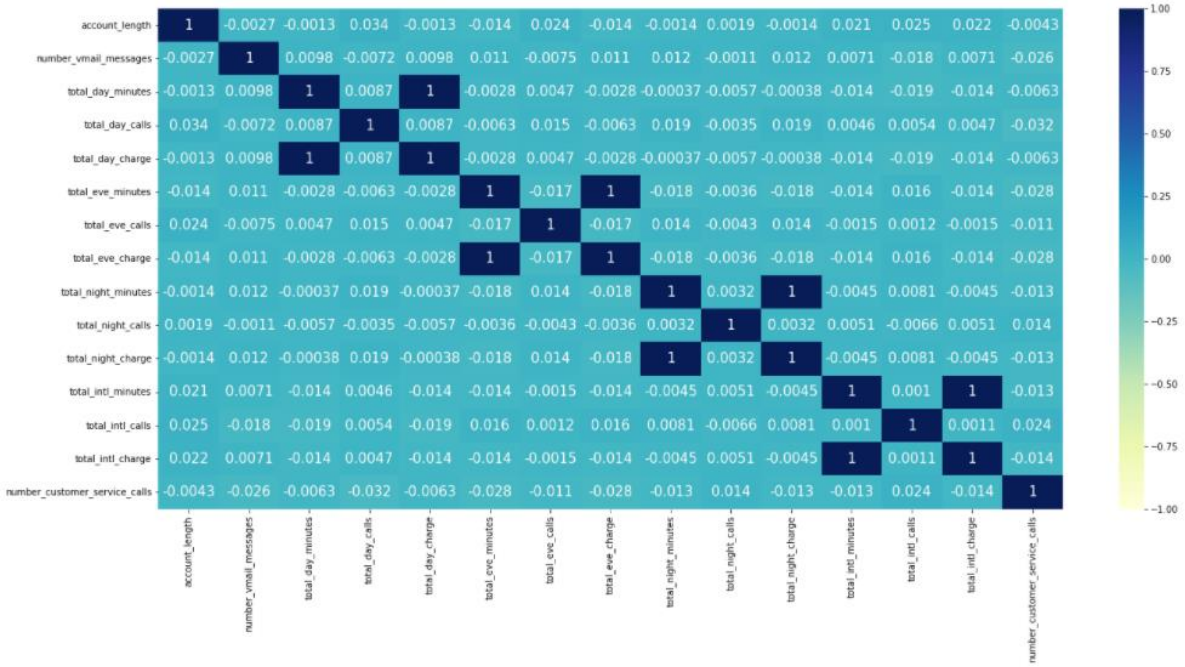
5 – Stacked bar plot for voice mail plan and international plan against churn rate.



From the plot, we can observe that Customers who do not have an international plan have churned the most and customers who do not own a voice mail plan have churned the most.

Multivariate Analysis

Checking for multicollinearity in the dataset using heatmap.



From the heatmap, total_day_minutes, total_eve_minutes, total_night_minutes and total_intl_minutes, and total_day_charge, total_eve_charge, total_night_charge and total_intl_charge respectively, have perfect positive correlation. There is high multicollinearity present in the data. Hence, we will handle multicollinearity before building the model.

Dataset Segregation

The dataset is further segregated on the following aspects in order to build the model and predict the churn rate with the best significant variable.

The dataset is segregated as:

1. Dataset without Feature Engineering, using VIF Technique.

2. Dataset without Feature Engineering, dropping highly positively correlated columns, using Heatmap.

3. Dataset with Feature Engineering, dropping highly positively correlated columns, using Heatmap.

4. Dataset without Feature Engineering, dropping highly positively correlated columns using Heatmap - After Outlier Treatment.

5. Dataset with Feature Engineering, dropping highly positively correlated columns using Heatmap - After Outlier Treatment.

Train Test Split

Before building the model, each data set is separated into train and test data and the train data is further separated into train and validation set for performing evaluation test on the model.

```
X2 = sm.add_constant(X2)

X2_full, X2_test, y2_full, y2_test = train_test_split(X2, df_target_mulcol_drop , random_state = 10, test_size = 0.2)

print('X2_full', X2_full.shape)
print('y2_full', y2_full.shape)

print('X2_test', X2_test.shape)
print('y2_test', y2_test.shape)
```

```
X2_full (10313, 17)
y2_full (10313,)
X2_test (2579, 17)
y2_test (2579,)
```

let us split the full train dataset into train and validation sets

```
X2_train, X2_val, y2_train, y2_val = train_test_split(X2_full, y2_full, random_state = 10, test_size = 0.2)

print('X2_train', X2_train.shape)
print('y2_train', y2_train.shape)

print('X2_val', X2_val.shape)
print('y2_val', y2_val.shape)
```

```
X2_train (8250, 17)
y2_train (8250,)
X2_val (2063, 17)
y2_val (2063,)
```

SMOTE – Synthetic Minority Oversampling Technique

In the initial analysis of the target variable, we observed that there was imbalance present in the dataset. The imbalance is corrected using SMOTE method.

```
smote = SMOTE(sampling_strategy = 0.3) # sampling startegy is the proportion of minority sample

X2_train_os , y2_train_os = smote.fit_resample(X2_train , y2_train)

print('X2_train_os', X2_train_os.shape)
print('y2_train_os', y2_train_os.shape)

print('X2_val', X2_val.shape)
print('Y2_val', y2_val.shape)

print('X2_test', X2_test.shape)
print('y2_test', y2_test.shape)

```

```
X2_train_os (9226, 17)
y2_train_os (9226,)
X2_val (2063, 17)
Y2_val (2063,)
X2_test (2579, 17)
y2_test (2579,)
```

Initially Target variable has imbalanced class distribution.

After the SMOTE Analysis Technique.

Positive class (Churn=1) increased to 23.08% which seems to be balanced along with the negative class (churn=0) 76.92%.

MODEL BUILDING

After training and testing these models on each of the segregated dataset, we identified that the 'dataset without feature engineering, dropping highly positively correlated columns', performed better on all the models compared to other datasets. Hence, the models built on the above said dataset is considered for further explanation.

Building Base Model.

Logistic Regression Using Logit Function.

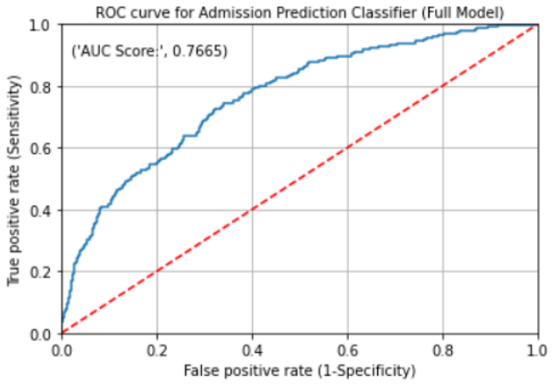
Logistic Regression is one of the simple and commonly used Machine Learning algorithms for two-class classification. It is easy to implement and can be used as the baseline for any binary classification problem. It uses a log of odds as the dependent variable. Logistic Regression predicts the probability of occurrence of a binary event utilizing a logit function.

The equation of a logistic regression equation is defined as,

$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Performance Metrics of Logistic Regression Using Logit Function.

```
Accuracy score:  0.8503
Precision score:  0.4832
Recall score:  0.3042
F1 score:  0.3734
AUC score:  0.7665
```

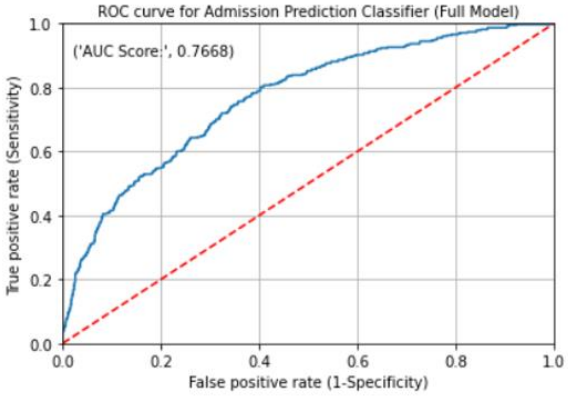


Logistic Regression Using Sklearn.

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Performance Metrics of Logistic Regression Using Logit Function.

```
Accuracy score:  0.8511
Precision score:  0.4868
Recall score:  0.2937
F1 score:  0.3663
AUC score:  0.7668
```



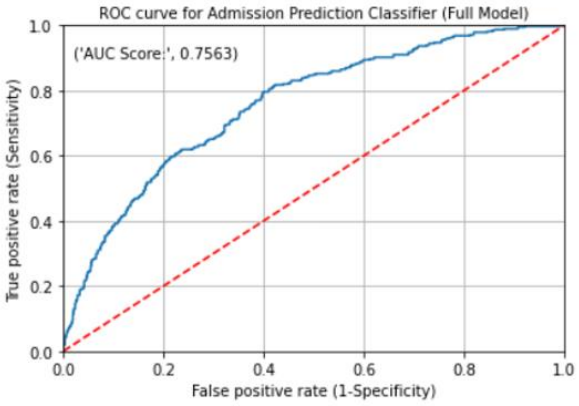
Logistic Regression Using Sklearn With Significant Features Using RFE.

Recursive feature elimination is a greedy optimization algorithm. The main aim of this method is to select a best-performing feature subset. It will not randomly select any feature. Rather than, it will find out which is the most useful feature. And in the next iteration, it will add the next useful feature concerning the target variable. Finally, it will rank all the features and eliminate the lower ones.

The logistic regression is built on the significant features from RFE.

Performance Metrics of Logistic Regression Using Sklearn with Significant Features Using RFE.

```
Accuracy score:  0.8468
Precision score:  0.4577
Recall score:  0.2434
F1 score:  0.3178
AUC score:  0.7563
```

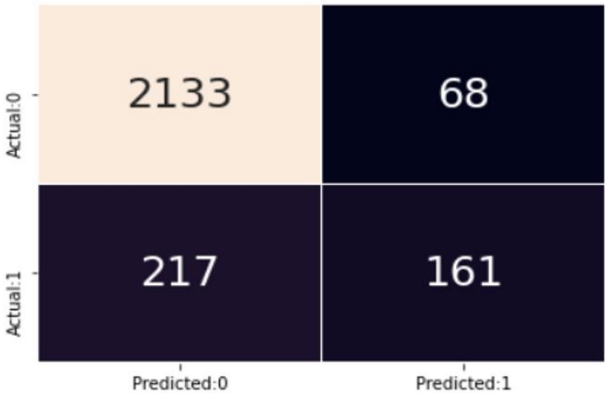


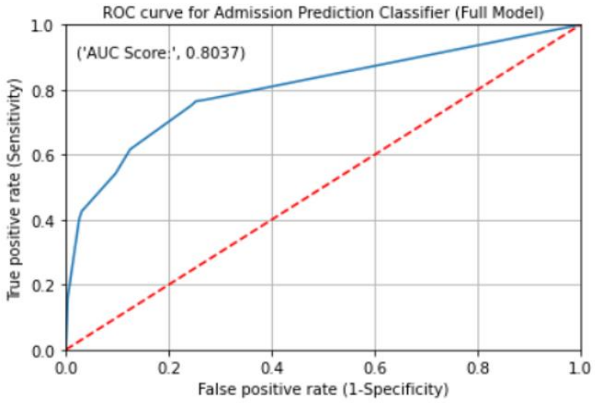
Decision Tree with Hyper Parameter Tuning.

Decision tree is a machine learning algorithm which can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves. They are powerful analytical models that have the ability to comprehend data with minimal pre-processing time. Initially we tried training the model with decision tree with default parameters and observed that the model was over fit. Hence in order to overcome this problem, we will be doing Hyperparameter tuning using GridsearchCV.

Performance Metrics of Decision Tree with Hyper Parameter Tuning:

```
Accuracy score:  0.8895
Precision score:  0.7031
Recall score:  0.4259
F1 score:  0.5305
AUC score:  0.8037
```



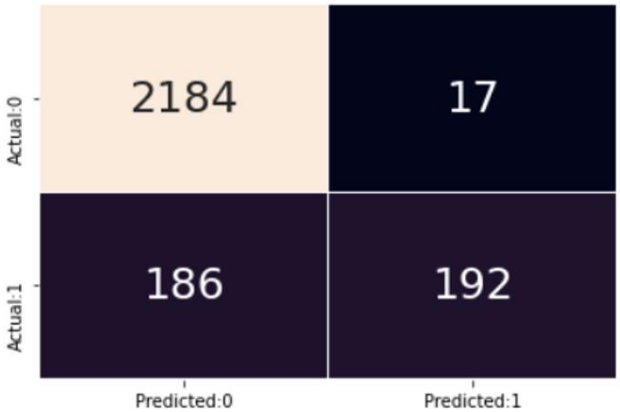


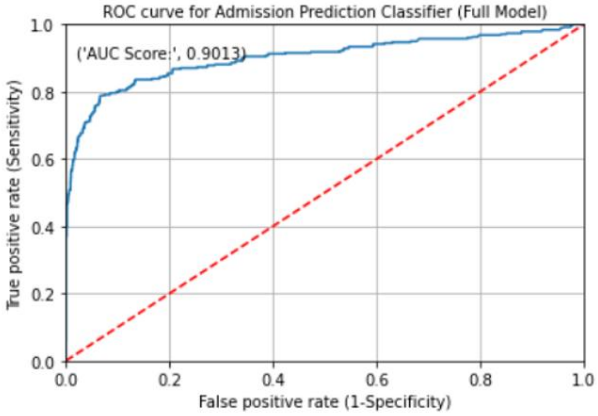
Random Forest with Hyper Parameter Tuning.

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

Performance Metrics of Random Forest with Hyper Parameter Tuning:

```
Accuracy score:  0.9213
Precision score:  0.9187
Recall score:  0.5079
F1 score:  0.6542
AUC score:  0.9013
```



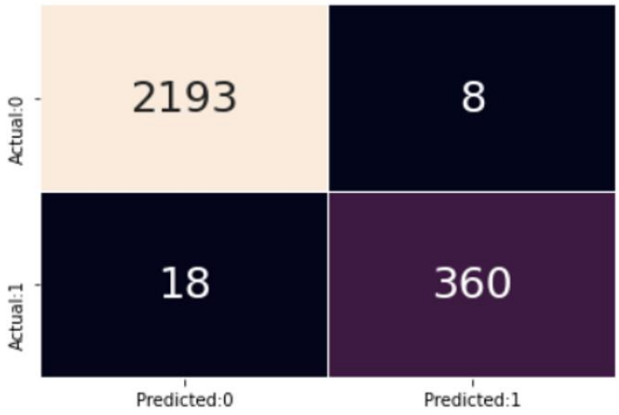


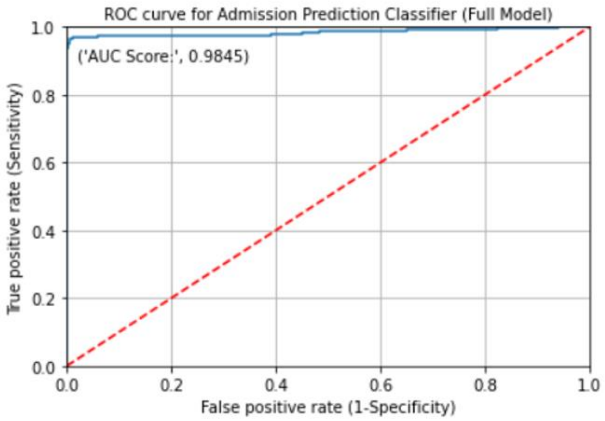
XGBoost with Hyper Parameter Tuning.

XGBoost or Extreme Gradient Boosting, is a decision-tree based ensemble learner using the boosting technique, which is designed to achieve great speed and optimal performance. XGBoost works by building decision trees in sequences, one after another, where the next built decision tree focuses on the errors and weaknesses from the previous one and improves the prediction performance with those weaknesses in consideration. In each iteration or sequence the tree structure learns from the previous tree's outcome and residuals. Residuals is the difference between the real value and the predictive value. By combining these sets of weak learners, starting with one base learner, XGBoost creates a strong classifier in order to make more accurate predictions. XGBoost's great cache optimization comes with both pros and cons. The advantage is that XGBoost can predict outputs with a high accuracy. Unlike most tree learning algorithms, the XGBoost classifier finds the best split amongst trees and is able to handle datasets with missing values accurately. This is why it is considered a good model when it comes to performance in terms of its high prediction accuracy.

Performance Metrics of XGBoost with Hyper Parameter Tuning:

```
Accuracy score:  0.9899
Precision score:  0.9783
Recall score:  0.9524
F1 score:  0.9651
AUC score:  0.9845
```





Bias and Variance Error

```
from sklearn.model_selection import cross_val_score

crossvalscore = cross_val_score(xgb_grid_model2, X2_train_os,
                                y2_train_os, cv=5, scoring='f1_weighted')

print(crossvalscore)
```

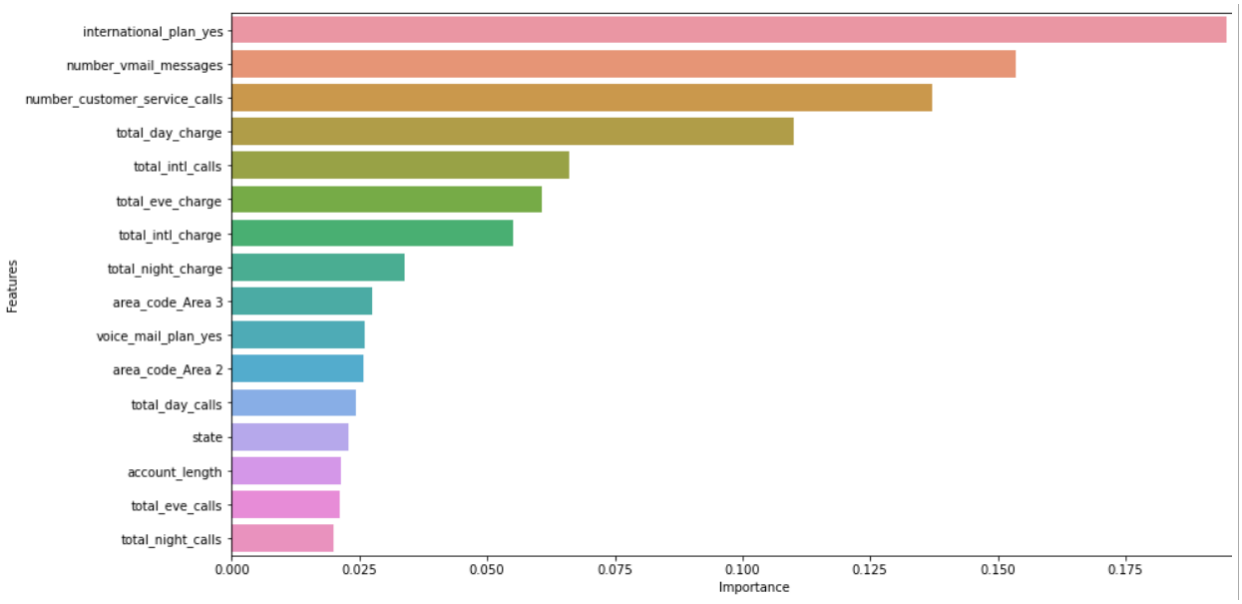
```
[0.97970483 0.98304668 0.98250805 0.97069698 0.97799755]
```

```
print('Bias - ', 1 - np.mean(crossvalscore))
print('Variance - ', np.std(crossvalscore)/np.mean(crossvalscore))
```

```
Bias -  0.021209181474988403
Variance -  0.0045439579404180485
```

We can say that the XGBoost model is good as both the bias and variance is low.

Feature Importance



In the dataset, the features ‘international_plan’, ‘number_customer_service_calls’, ‘number_vmail_messages’ and ‘total_day_charge’ has the highest importance in predicting the customer churn using XGBoost model.

MODEL EVALUATION

	Model Name	Accuracy	Precision	Recall	F1 Score
0	Logit-VIF	0.846452	0.457944	0.259259	0.331081
1	LogReg_Sk_VIF	0.844513	0.445498	0.248677	0.319185
2	LogReg_sk_rfe_VIF	0.846840	0.459330	0.253968	0.327087
3	DecisionTree_HYP_VIF	0.873594	0.577844	0.510582	0.542135
4	RandomForest_HYP_VIF	0.904614	0.726027	0.560847	0.632836
5	XGBoost_HYP_VIF	0.978674	0.940054	0.912698	0.926174
6	Logit_Col_drop	0.851105	0.487395	0.306878	0.376623
7	LogReg_SK_Col_drop	0.853044	0.497797	0.298942	0.373554
8	LogReg_SK_rfe_Col_drop	0.846452	0.453608	0.232804	0.307692
9	DecisionTree_HYP_Col_drop	0.878247	0.619403	0.439153	0.513932
10	RandomForest_HYP_Col_drop	0.921287	0.926829	0.502646	0.651801
11	XGBoost_HYP_Col_drop	0.991857	0.997214	0.947090	0.971506
12	Logit_FeaEng_Col_drop	0.848003	0.468468	0.275132	0.346667
13	LogReg_SK_FeaEng_Col_drop	0.847228	0.463964	0.272487	0.343333
14	LogReg_SK_rfe_FeaEng_Col_drop	0.850717	0.480000	0.222222	0.303797
15	DecisionTree_HYP_FeaEng_Col_drop	0.921675	0.843750	0.571429	0.681388
16	RandomForest_HYP_FeaEng_Col_drop	0.945715	1.000000	0.629630	0.772727
17	XGBoost_HYP_FeaEng_Col_drop	0.987592	0.983240	0.931217	0.956522
18	Logit_WO_FeaEng_Col_drop_out	0.910772	0.610619	0.567901	0.588486
19	LogReg_SK_WO_FeaEng_Col_drop_out	0.911697	0.615044	0.572016	0.592751
20	LogReg_SK_rfe_WO_FeaEng_Col_drop_out	0.904300	0.586538	0.502058	0.541020
21	DecisionTree_HYP_WO_FeaEng_Col_drop_out	0.918632	0.705521	0.473251	0.566502
22	RandomForest_HYP_WO_FeaEng_Col_drop_out	0.937587	0.950000	0.469136	0.628099
23	XGBoost_HYP_WO_FeaEng_Col_drop_out	0.990754	0.991189	0.925926	0.957447
24	Logit_FeaEng_Col_drop_out	0.915159	0.577778	0.560345	0.568928
25	LogReg_SK_FeaEng_Col_drop_out	0.913867	0.570796	0.556034	0.563319
26	LogReg_SK_rfe_FeaEng_Col_drop_out	0.917743	0.592760	0.564655	0.578366
27	DecisionTree_HYP_FeaEng_Col_drop_out	0.894918	0.481481	0.672414	0.561151
28	RandomForest_HYP_FeaEng_Col_drop_out	0.949182	0.952381	0.517241	0.670391
29	XGBoost_HYP_FeaEng_Col_drop_out	0.987941	0.951327	0.926724	0.938865

On comparing the evaluation metrics of various models built, we observe that the ‘dataset without feature engineering, dropping highly positively correlated columns using heatmap’ has overall best performance and, the XGBoost model in this dataset has the highest overall score. Hence, we will take this model as our final model for the prediction of churn.

- Model is doing a good job predicting the majority class and not at all able to predict the minority class. We have 14% records corresponding to the churn - yes category.
- After doing SMOTE process combined with boosting & bagging techniques individual class predictions and model performance has improved.
- In particular, the XGBoost classifier performed best based on overall accuracy, precision, recall, and F1-score.
- It should be emphasized that the dataset used for this study was imbalanced, and we applied different sampling method in order to investigate how balancing the training dataset would affect the result. We conclude that balancing, using sampling method combined with boosting, has a positive influence for the studied problem.

Implications

Our solution is predicting the customer churn specifically in the Telecom sector.

- Churn prediction is valuable for firms since we could target actual churners with preventative measures and stop wasting money on customers with no intentions of leaving.
- It is important to know which of the customers might churn or not to increase the possibilities to even retain a few of them.
- Each loss of a customer affects the revenue stream and if there is a possibility to decrease this effect, firms should act upon this.
- Even a small improvement in customer retention could increase the profits and revenues significantly.

Limitations

- Databases of telecom customer can present difficulties related to outliers, missing values, incomplete or inconsistent records, high dimensionality, and complexity of features.
- The number of observations is decent, but if we could have more columns of features like competitor's information, customer churned to which competitor's service and their services provided, we could draw more insights from the result.
- Model is designed based on the available features as given in the datasets. We need almost all the features as in this dataset to successfully predict the churn
- The nature of our dataset is a cross-sectional dataset. We have a feature account length which represents the time period which customer stayed in the service. There are no detailed time series factors inside it. Since our goal is to predict churn rate, it is best that we can find a time series dataset containing all the customer's churn information like exact time period of the churn happened to obtain better results for predicting and making decisions for the future market.

Closing Reflections

There is huge learning in terms of understanding the domain and how important is the understanding the dataset is imperative. As the lack of understanding of the dataset can lead to catastrophic results.

- Significant amount of domain knowledge is required to analyze each feature in the dataset.
- From our results we learnt that ensemble techniques such as boosting and bagging improved the performance of our churn prediction model better relative to the studied single learner Decision Tree
- Understood how to handling imbalanced dataset and different imbalance handling techniques.
- In Outlier treatment we have implemented basic IQR technique but there is huge scope to advanced techniques like Automatic Outlier Detection Algorithms & PyOD ext. has to be learned.

REFERENCES

- ✓ https://www.analyticsvidhya.com/blog/2021/05/logistic-regression-supervised-learning-algorithm-for-classification/
- ✓ https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python
- ✓ https://www.analyticsvidhya.com/blog/2021/05/dealing-with-missing-values-in-python-a-complete-guide/
- ✓ https://www.geeksforgeeks.org/grouping-categorical-variables-in-pandas-dataframe/
- ✓ https://www.analyticsvidhya.com/blog/2021/09/a-complete-guide-to-understand-classification-in-machine-learning/
- ✓ https://pythonbasics.org/split-train-test/
- ✓ https://www.analyticsvidhya.com/blog/2021/05/detecting-and-treating-outliers-treating-the-odd-one-out/
- ✓ https://towardsdatascience.com/churn-prediction-with-machine-learning-ca955d52bd8c
- ✓ https://towardsdatascience.com/3-ways-to-improve-your-machine-learning-results-without-more-data-f2f0fe78976e
- ✓ https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/
- ✓ https://www.analyticsvidhya.com/blog/2021/04/a-profound-comprehension-of-bias-and-variance/
- ✓ https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/