Churn Reduction Project Report

Sharang Shrivastava

25 December 2018

Contents

Customer Churn

Customer churn, customer turnover, or customer defection, is the loss of clients or customers.

Telephone service companies, Internet service providers, pay TV companies, insurance firms, and alarm monitoring services, often use customer attrition analysis and customer attrition rates as one of their key business metrics because the cost of retaining an existing customer is far less than acquiring a new one. Companies from these sectors often have customer service branches which attempt to win back defecting clients, because recovered long-term customers can be worth much more to a company than newly recruited clients.

Companies usually make a distinction between voluntary churn and involuntary churn. Voluntary churn occurs due to a decision by the customer to switch to another company or service provider, involuntary churn occurs due to circumstances such as a customer's relocation to a long-term care facility, death, or the relocation to a distant location. In most applications, involuntary reasons for churn are excluded from the analytical models. Analysts tend to concentrate on voluntary churn, because it typically occurs due to factors of the company-customer relationship which companies control, such as how billing interactions are handled or how after-sales help is provided.

predictive analytics use churn prediction models that predict customer churn by assessing their propensity of risk to churn. Since these models generate a small prioritized list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who are most vulnerable to churn.

1.Data

Let's have a look at our training data that is stored in "Train_data.csv" file

df_train.info()

```
Data columns (total 21 columns):
state                           3333 non-null object
account length                  3333 non-null int64
area code                       3333 non-null int64
phone number                    3333 non-null object
international plan              3333 non-null object
voice mail plan                 3333 non-null object
number vmail messages           3333 non-null int64
total day minutes               3333 non-null float64
total day calls                 3333 non-null int64
total day charge                3333 non-null float64
total eve minutes               3333 non-null float64
total eve calls                 3333 non-null int64
total eve charge                3333 non-null float64
total night minutes             3333 non-null float64
total night calls               3333 non-null int64
total night charge              3333 non-null float64
total intl minutes              3333 non-null float64
total intl calls                3333 non-null int64
total intl charge               3333 non-null float64
number customer service calls   3333 non-null int64
Churn                           3333 non-null object

dtypes: float64(8), int64(8), object(5)
```

1.1 Data overview

df_train.shape[0] Rows : 3333

df_train.shape[1] Columns : 21

Features : df_train.columns

['state', 'account length', 'area code', 'phone number', 'international plan', 'voice mail plan', 'number vmail messages', 'total day minutes', 'total day calls', 'total day charge', 'total eve minutes', 'total eve calls', 'total eve charge', 'total night minutes', 'total night calls', 'total night charge', 'total intl minutes', 'total intl calls', 'total intl charge', 'number customer service calls','Churn']

Missing values :

df_train.isnull().sum().values().sum()

And the output in this query also comes to be 0.

Unique Value Table

feature	unique
state	51
account length	212
area code	3
phone number	3333
international plan	2
voice mail plan	2
number vmail messages	46
total day minutes	1667
total day calls	119
total day charge	1667
total eve minutes	1611
total eve calls	123
total eve charge	1440
total night minutes	1591
total night calls	120
total night charge	933
total intl minutes	162
total intl calls	21
total intl charge	162
number customer service calls	10
Churn	2

2. Data Manipulation

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.
We can see that **"area code","international plan"** and **"voice mail plan"** are categorical variables or they could be transformed into one.

Let’s change the data types of above mentioned variables.

```
cat_cols=["area code","international plan","voice mail plan","Churn"]
for i in cat_cols:
    df_train[i]=df_train[i].astype("category")
    df_test[i]=df_test[i].astype("category")
    print(i," ",df_train[i].dtypes," ",df_test[i].dtypes)
```

now we will give categorical codes to these category column so that to convert it to numerical form to be able to be processed by our model algorithm

```
for i in cat_cols:
    df_train[i]=df_train[i].cat.codes
    df_test[i]=df_test[i].cat.codes
```

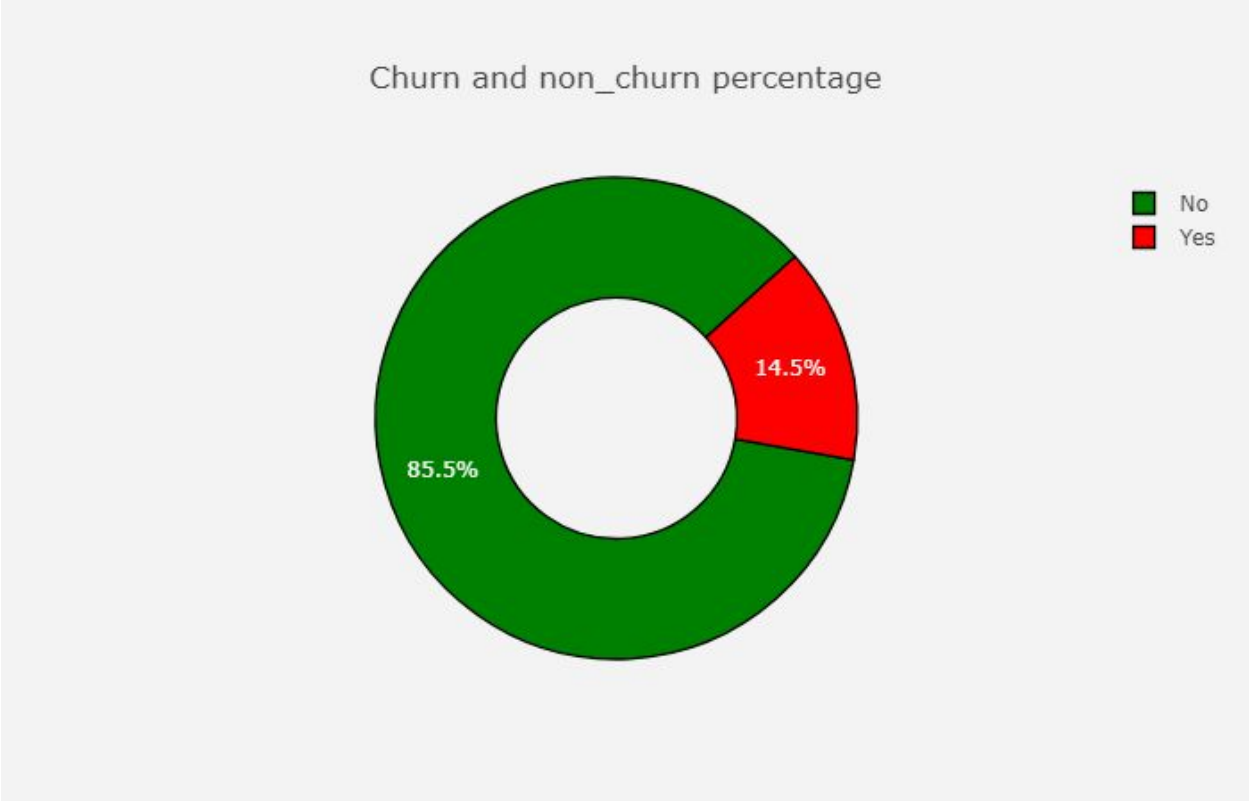
let’s divide dataframe into two parts ,churn and non churn , to visualize data better

```
churn     = df[df["Churn"].str.strip()== "True."]

churn["Churn"]=churn["Churn"].str.strip().replace({"True.":"Yes"})

not_churn = df[df["Churn"].str.strip()== "False."]

not_churn["Churn"]=not_churn["Churn"].str.strip().replace({"False.":"No"})
```

3. Exploratory Data Analysis

3.1. Customer attrition in data

We can now see that our data is pretty much imbalanced.

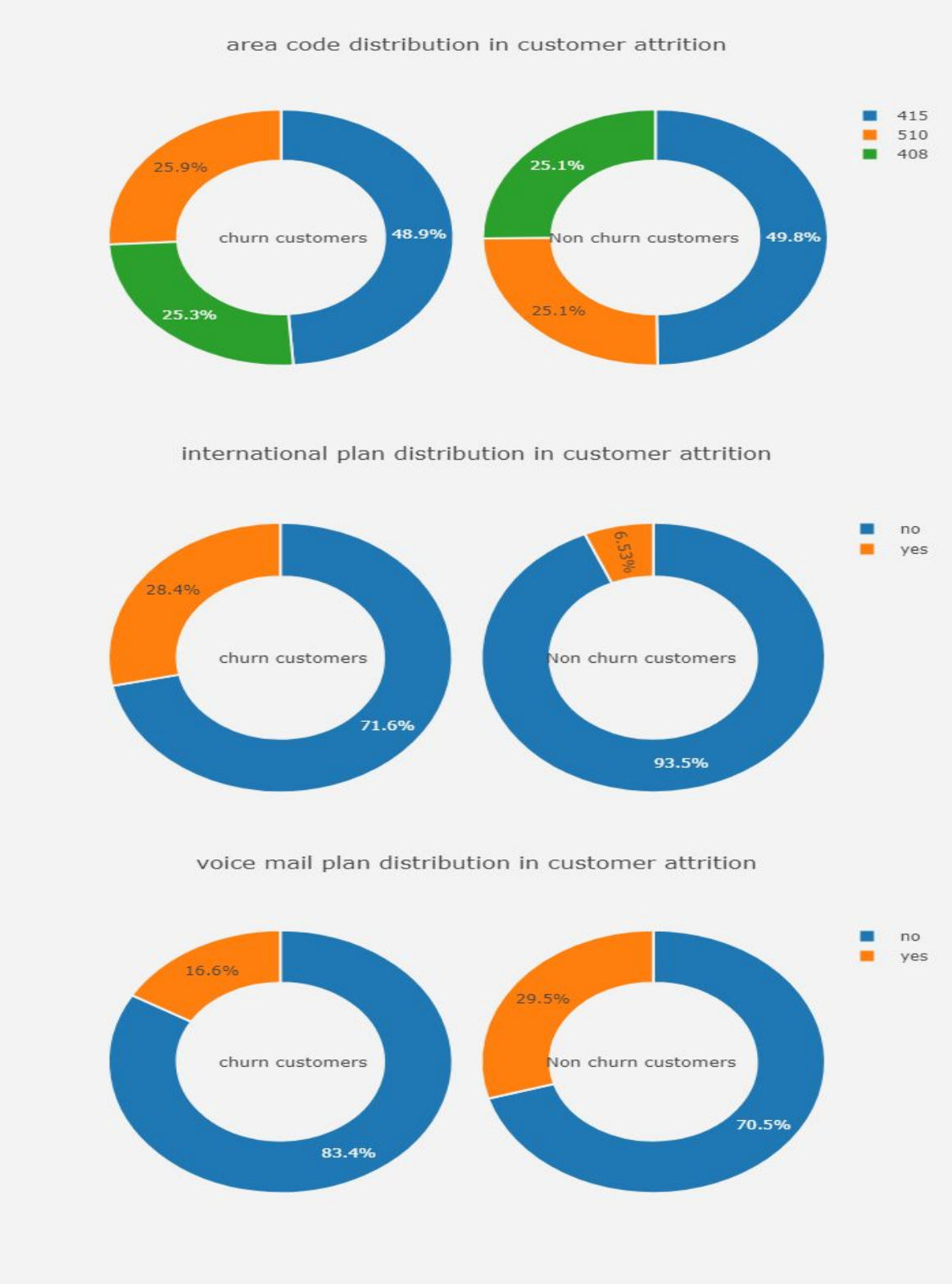


churn % = 14.491449144914492 not churn % = 85.5085508550855

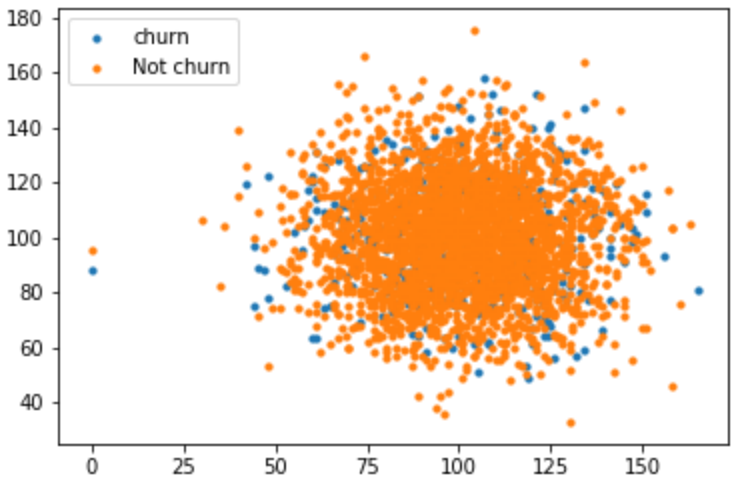
Most real-world classification problems display some level of class imbalance, which is when each class does not make up an equal portion of your data-set. It is important to properly adjust your metrics and methods to adjust for your goals. If this is not done, you may end up optimizing for a meaningless metric in the context of your use case.For example, suppose you have two classes — A and B. Class A is 90% of your data-set and class B is the other 10%, but you are most interested in identifying instances of class B. You can reach an accuracy of 90% by simply predicting class A every time, but this provides a useless classifier for your intended use case. Instead, a properly calibrated method may achieve a lower accuracy, but would have a substantially higher true positive rate (or recall), which is really the metric you should have been optimizing for.

3.2. Variables distribution in customer attrition

Probability distribution of categorical variables with respect to Churn and Non Churn customers



Scatter plot of Total day calls and Total night calls distribution
(Churn vs Non Churn)



3.3 Variable Summary

Let's look at the summary of our dataframe

feature	count	mean	std	min	25%	50%	75%	max
account length	3333	101.0648	39.8221	1	74	101	127	243
area code	3333	437.1824	42.3713	408	408	415	510	510
number vmail messages	3333	8.099	13.6884	0	0	0	20	51
total day minutes	3333	179.7751	54.4674	0	143.7	179.4	216.4	350.8
total day calls	3333	100.4356	20.0691	0	87	101	114	165
total day charge	3333	30.5623	9.2594	0	24.43	30.5	36.79	59.64
total eve minutes	3333	200.9803	50.7138	0	166.6	201.4	235.3	363.7
total eve calls	3333	100.1143	19.9226	0	87	100	114	170
total eve charge	3333	17.0835	4.3107	0	14.16	17.12	20	30.91
total night minutes	3333	200.872	50.5738	23.2	167	201.2	235.3	395
total night calls	3333	100.1077	19.5686	33	87	100	113	175
total night charge	3333	9.0393	2.2759	1.04	7.52	9.05	10.59	17.77
total intl minutes	3333	10.2373	2.7918	0	8.5	10.3	12.1	20
total intl calls	3333	4.4794	2.4612	0	3	4	6	20
total intl charge	3333	2.7646	0.7538	0	2.3	2.78	3.27	5.4
number customer service calls	3333	1.5629	1.3155	0	1	1	2	9

3.4 Feature importance

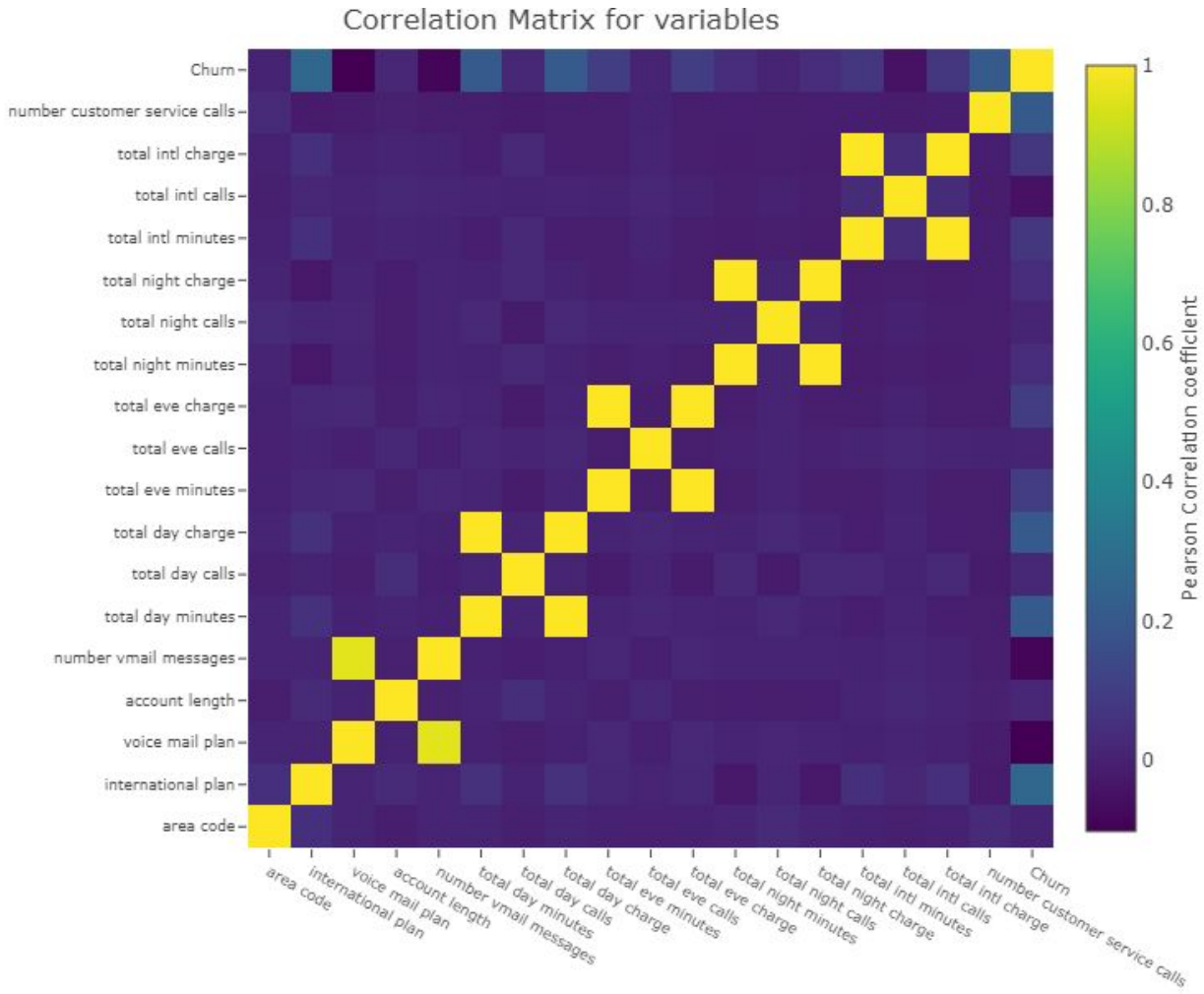
Let’s have a look at feature importance by using ExtraTreesClassifier

	features	importance
0	account length	0.035586
1	area code	0.020760
2	international plan	0.073920
3	voice mail plan	0.024064
4	number vmail messages	0.022815
5	total day minutes	0.107724
6	total day calls	0.032861
7	total day charge	0.148310
8	total eve minutes	0.056118
9	total eve calls	0.035783
10	total eve charge	0.059992
11	total night minutes	0.037815
12	total night calls	0.031634
13	total night charge	0.038282
14	total intl minutes	0.050548
15	total intl calls	0.056307
16	total intl charge	0.044085
17	number customer service calls	0.123395

we can see that "total day minutes","total day charges" and "number customer service calls" are the most important features

3.5 Correlation Matrix

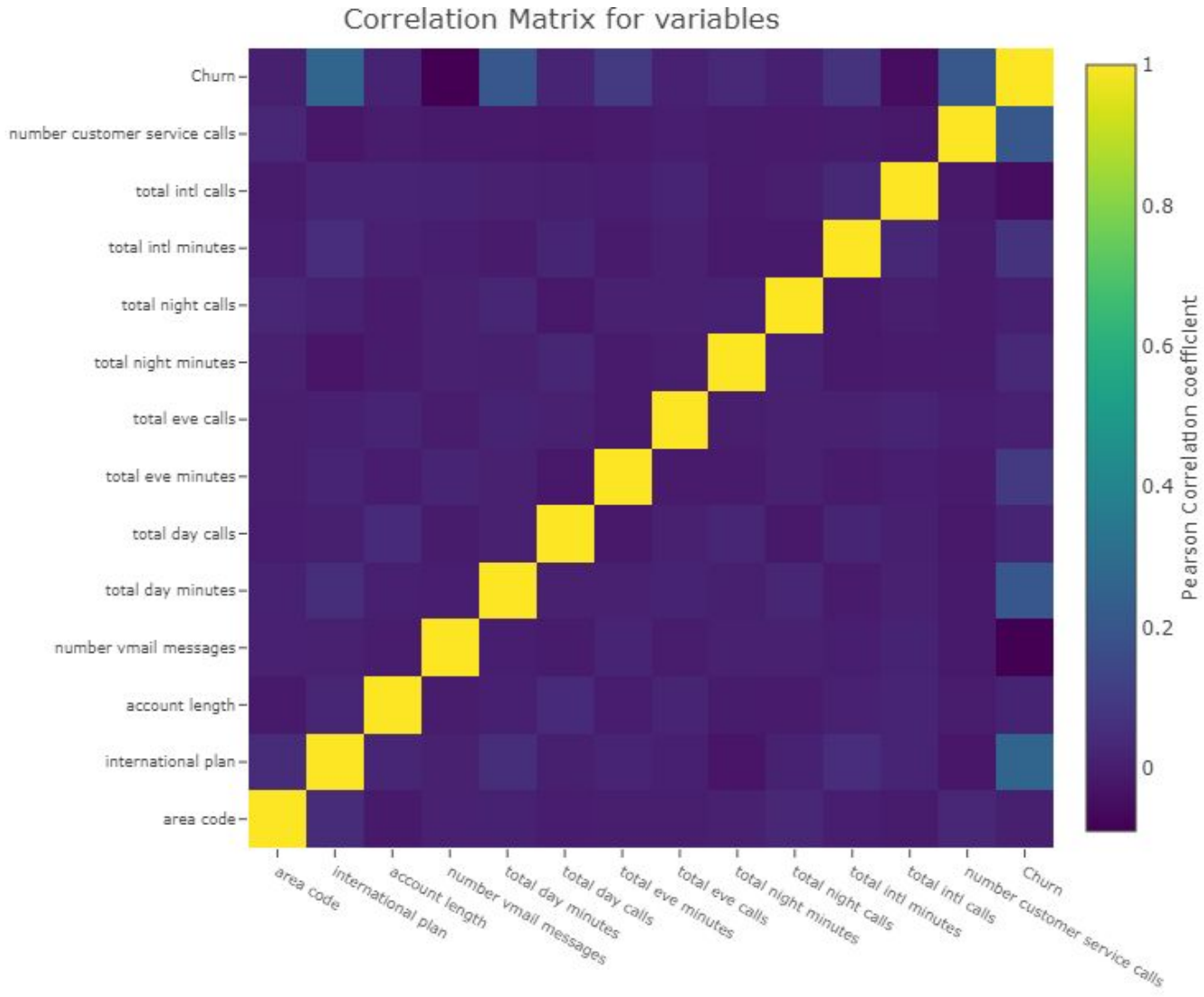
Let's check out the correlation between predictor variables and visualize it.



By looking at above plot we can determine high correlation between some variables
voice mail plan - number vmail messages =0.9569
total day charges - total day minutes = 1.0
total eve charges - total eve minutes = 1.0
total night charges - total night minutes = 1.0
total intl charges - total intl minutes = 1.0

Reason of such high collinearity could be that some of these variables are derived directly from other variables,therefore we could see that collinearity between some pair of variables is exactly 1.So we have to drop some of these to remove high correlation in our data

We have removed **"total day charge","total eve charge","total night charge","total intl charge"** variables from our data set.Now lets see again our correlation plot**.**



Now we can see that all correlation value is within permissible limit.Hence we can conclude that no high correlation is present in our data . Now we are left with only 13 predictor variables and one target variable.We will now move to the next step , that is , normalization of our data .

4. Data preprocessing

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
#Target columns
target_col = ["Churn"]
#categorical columns
cat_cols   = telcom.nunique()[telcom.nunique() < 6].keys().tolist()
cat_cols   = [x for x in cat_cols if x not in target_col]
#Binary columns with 2 values
bin_cols   = telcom.nunique()[telcom.nunique() == 2].keys().tolist()
#numerical columns
num_cols   = [x for x in telcom.columns if x not in cat_cols + target_col]
#Label encoding Binary columns
le = LabelEncoder()
for i in bin_cols :
    df_train[i] = le.fit_transform(df_train[i])
```

Standardized Distance

One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables. For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher influence on the distance calculated. One

solution is to standardize the training set

```
std = StandardScaler()
scaledtrain = std.fit_transform(df_train[num_cols])
```

Dropping original values merging scaled values for numerical columns

```
df_train = df_train.drop(columns = num_cols,axis = 1)
df_train = df_train.merge(scaledtrain,left_index=True,right_index=True,how
= "left")
```

5. Model Building

The dependent variable can fall in either of the four categories:
1. Nominal
2. Ordinal
3. Interval
4. Ratio

The dependent variable, in our case Churn, is Nominal.Now lets play with models
dataframe - processed data frame
Algorithm - Algorithm used
training_x - predictor variables data frame(training)
testing_x - predictor variables data frame(testing)
training_y - target variable(training)
testing_y - target variable(testing)
cf - "coefficients" for logistic regression & "features" for tree based models

5.1 Logistic Regression Baseline Model

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and

categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

1. A linear regression will predict values outside the acceptable range (e.g. predicting probabilities
outside the range 0 to 1)

2. Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

```
logit  = LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
        penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
        verbose=0, warm_start=False)
```

By using our model function **telecom_churn_prediction** we will generate results of our model algorithm

```
telecom_churn_prediction(logit,train_X,test_X,train_Y,test_Y,
                  cols,"coefficients",threshold_plot = True)
```

Classification report :

	precision	recall	f1-score	support
0	0.89	0.98	0.93	1443
1	0.58	0.18	0.28	224
micro avg	0.87	0.87	0.87	1667
macro avg	0.73	0.58	0.60	1667
weighted avg	0.84	0.87	0.84	1667

Accuracy Score : 0.8722255548890222

True Positives = 41 **True Negatives = 1413**

False Positives = 30 **False Negatives = 183**

Area under curve : 0.5811228467478466

We can see from the results that our accuracy is pretty good , more than 87% has been predicted correctly. But the cause of concern is the high false negative rates as the

customer might be interested in reducing false negative rate. We can also see that recall percentage for "churn" is very low. Our algorithm is failing to correctly predict those customer who could churn in future. We may have to improve our recall rate for "churn" customers and also reduce false negative rates in our further models.

We can see the recall rate for churn is very low, and also false negative rate is high,although accuracy is good but we are interested in recalling churn rate.
This is due to the fact that our data is imbalanced.
Let's use **Synthetic Minority Oversampling Technique (SMOTE)** approach to balance and model our data by oversampling the minority class.

5.2. Synthetic Minority Oversampling Technique (SMOTE)

Randomly pick a point from the minority class.
Compute the k-nearest neighbors (for some pre-specified k) for this point.
Add k new points somewhere between the chosen point and each of its neighbors

```
os = SMOTE(random_state = 0)
os_smote_X,os_smote_Y = os.fit_sample(smote_train_X,smote_train_Y)
os_smote_X = pd.DataFrame(data = os_smote_X,columns=cols)
os_smote_Y = pd.DataFrame(data = os_smote_Y,columns=target_col)
```

By this, we have oversampled the minority class,now our data is balanced.Let's build our model on top of it.

```
logit_smote = LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
    intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
       penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
       verbose=0, warm_start=False)
telecom_churn_prediction(logit_smote,os_smote_X,test_X,os_smote_Y,test_Y
,cols,"coefficients",threshold_plot = True)
```

Classification report :

	precision	recall	f1-score	support
0	0.96	0.78	0.86	1443
1	0.36	0.79	0.49	224
micro avg	0.78	0.78	0.78	1667
macro avg	0.66	0.79	0.68	1667
weighted avg	0.88	0.78	0.81	1667

Accuracy Score : 0.7792441511697661
True Positives = 178 **True Negatives = 1121**
False Positives = 322 **False Negatives = 46**
Area under curve : 0.785748316998317

After balancing our data using SMOTE approach we can see that we have pretty much improved our recall rate for “churn” customer ,and our false negative rate also got reduced, which is a good sign, but our accuracy got reduced a little bit.

Model performance

5.3 KNN classifier

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).

Algorithm

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

Applying knn algorithm to smote oversampled data.

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski',
        metric_params=None, n_jobs=1, n_neighbors=5, p=2,
        weights='uniform')
telecom_churn_prediction_alg(knn,os_smote_X,test_X,
                    os_smote_Y,test_Y,threshold_plot = True)
```

Classification report :

	precision	recall	f1-score	support
0	0.93	0.78	0.85	1443
1	0.31	0.63	0.42	224
micro avg	0.76	0.76	0.76	1667
macro avg	0.62	0.71	0.63	1667
weighted avg	0.85	0.76	0.79	1667

Accuracy Score : 0.7606478704259149

True Positives = 1126
False Positives = 317
True Negatives = 142
False Negatives = 82
Area under curve : 0.7071236758736759

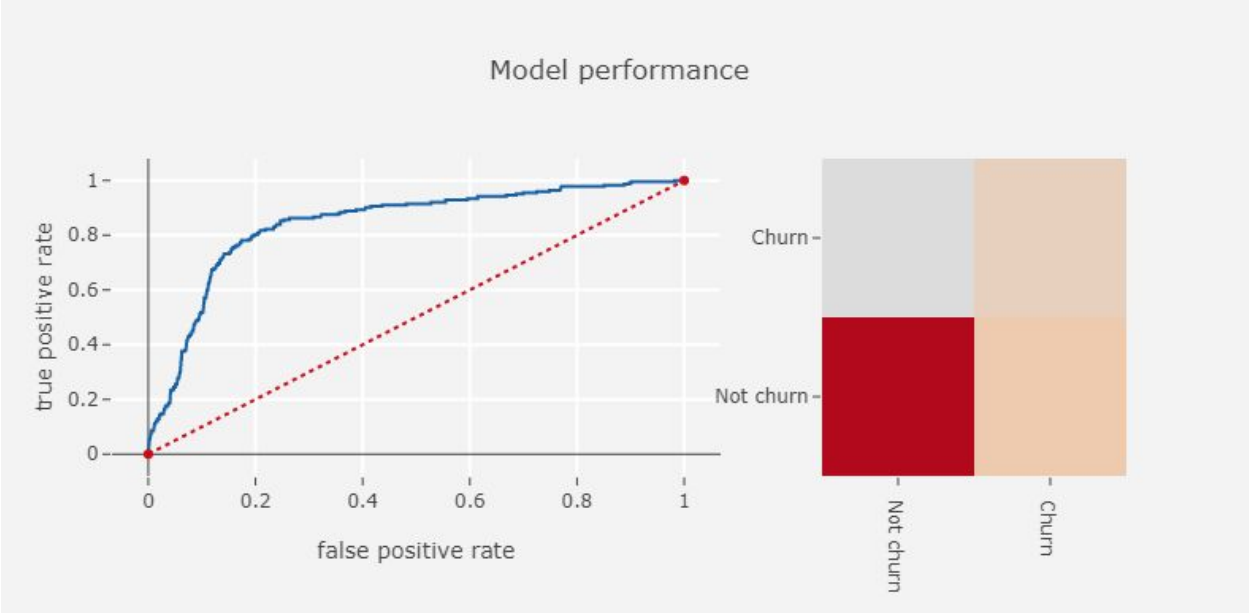
5.4 Gaussian Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB(priors=None)

telecom_churn_prediction_alg(gnb,os_smote_X,test_X,os_smote_Y,test_Y)
```

Classification report :

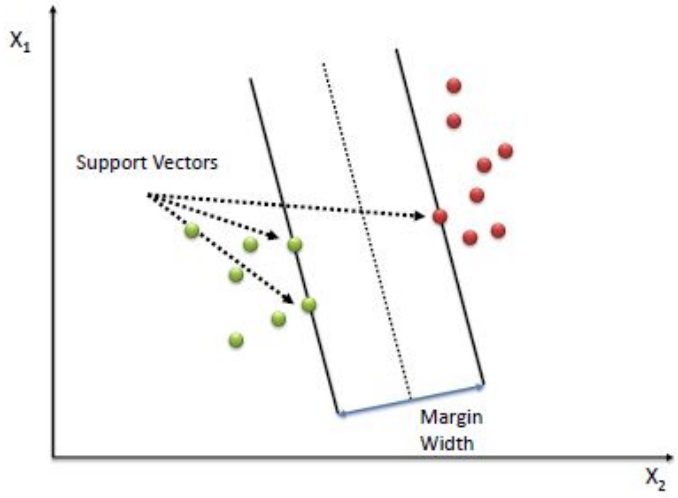
	precision	recall	f1-score	support
0	0.96	0.84	0.90	1443
1	0.43	0.75	0.54	224
micro avg	0.83	0.83	0.83	1667
macro avg	0.69	0.80	0.72	1667
weighted avg	0.88	0.83	0.85	1667

Accuracy Score : 0.8308338332333534
True Positives = 1217
False Positives = 226
True Negatives = 168
False Negatives = 56
Area under curve : 0.7966909216909217



5.5 Support Vector Machine

A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors.



Algorithm

1. Define an optimal hyperplane: maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications.
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

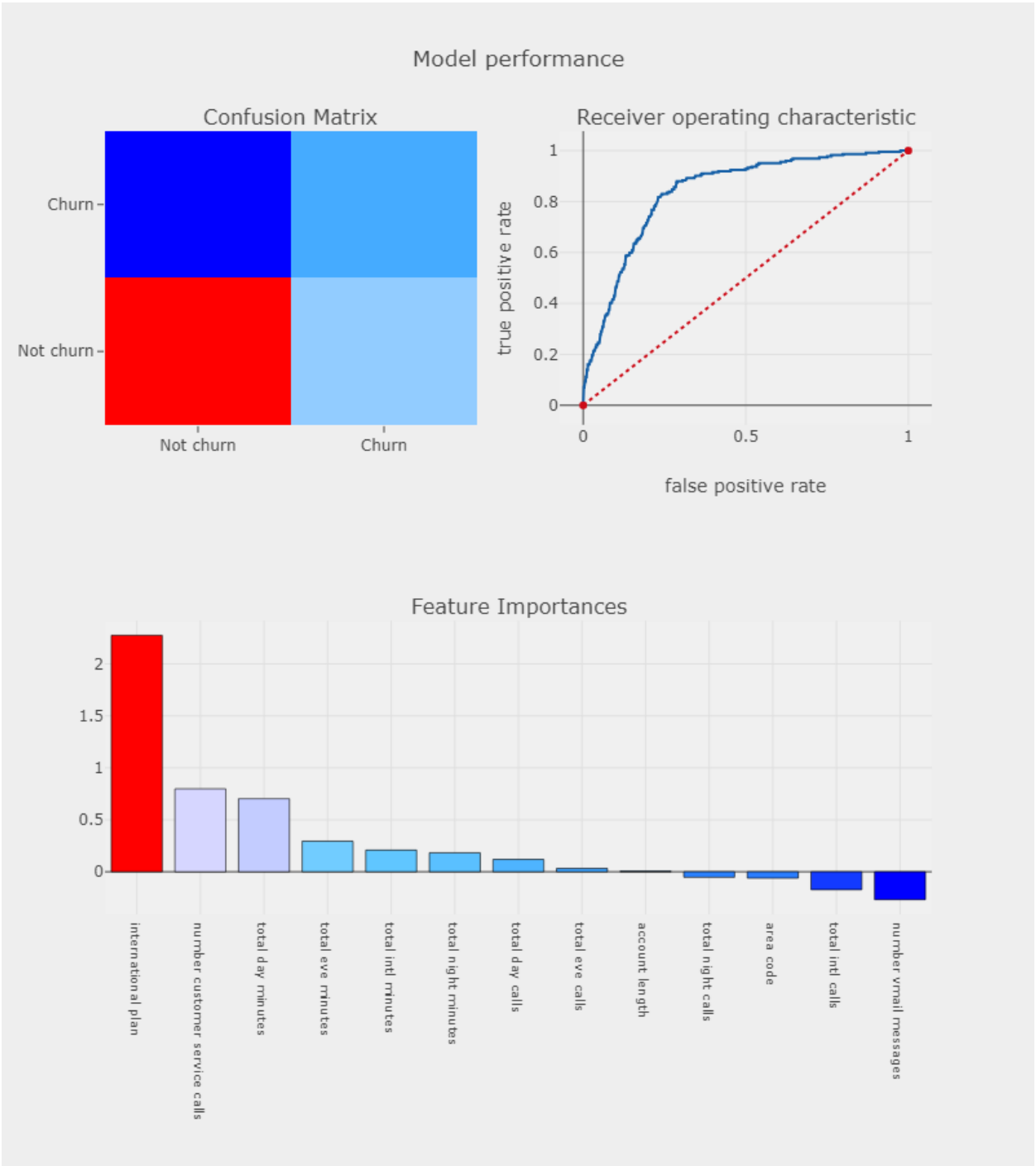
The beauty of SVM is that if the data is linearly separable, there is a unique global minimum value. An ideal SVM analysis should produce a hyperplane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly.

```
from sklearn.svm import SVC

svc_lin  = SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
                    decision_function_shape='ovr', degree=3, gamma=1.0,
               kernel='linear',max_iter=-1, probability=True,
               random_state=None, shrinking=True,
                    tol=0.001, verbose=False)

telecom_churn_prediction(svc_lin,os_smote_X,test_X,os_smote_Y,test_Y,
                  cols,"coefficients",threshold_plot = False)
```

Classification report :

	precision	recall	f1-score	support
0	0.96	0.76	0.85	1443
1	0.35	0.82	0.49	224
micro avg	0.77	0.77	0.77	1667
macro avg	0.66	0.79	0.67	1667
weighted avg	0.88	0.77	0.80	1667

Accuracy Score : 0.7696460707858428
True Positives = 1099
False Positives = 344
True Negatives = 184
False Negatives = 40
Area under curve : 0.7915181665181665


Model performance
Confusion Matrix
Churn
Not churn
Not churn
Churn
Receiver operating characteristic
true positive rate
1
0.8
0.6
0.4
0.2
0
0
0.5
1
false positive rate
Feature Importances
2
1.5
1
0.5
0
international plan
number customer service calls
total day minutes
total eve minutes
total intl minutes
total night minutes
total day calls
total eve calls
account length
total night calls
area code
total intl calls
number vmail messages

6. Model Performances

6.1 Model performance metrics

6.1.1 Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. For the problem in hand, we have N=2, and hence we get a 2 X 2 matrix. Here are a few definitions, you need to remember for a confusion matrix :

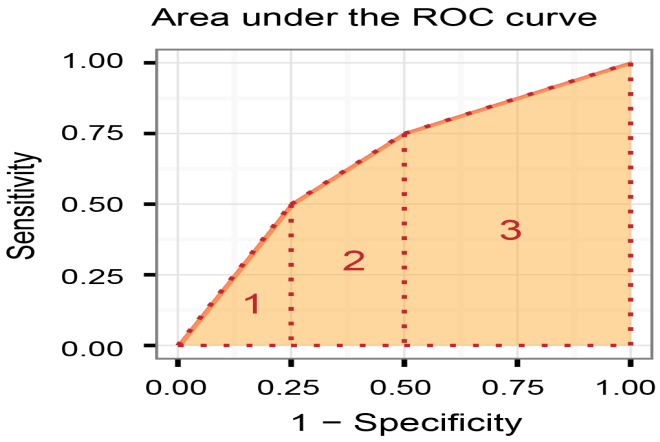
1. **Accuracy** : the proportion of the total number of predictions that were correct.
2. **Positive Predictive Value or Precision** : the proportion of positive cases that were correctly identified.
3. **Negative Predictive Value** : the proportion of negative cases that were correctly identified.
4. **Sensitivity or Recall** : the proportion of actual positive cases which are correctly identified.
5. **Specificity** : the proportion of actual negative cases which are correctly identified.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

6.1.2 Area Under the ROC curve (AUC – ROC)

The ROC curve is the plot between sensitivity and (1- specificity). (1- specificity) is also known as false positive rate and sensitivity is also known as True Positive rate. Following is the ROC curve for the case in hand.

This is again one of the popular metrics used in the industry. The biggest advantage of using ROC curve is that it is independent of the change in proportion of responders.



6.1.3 Root Mean Squared Error (RMSE)

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that error are unbiased and follow a normal distribution. Here are the key points to consider on RMSE:

1. The power of ‘square root’ empowers this metric to show large number deviations.
2. The ‘squared’ nature of this metric helps to deliver more robust results which prevents cancelling the positive and negative error values. In other words, this metric aptly displays the plausible magnitude of error term.
3. It avoids the use of absolute error values which is highly undesirable in mathematical calculations.
4. When we have more samples, reconstructing the error distribution using RMSE is considered to be more reliable.
5. RMSE is highly affected by outlier values. Hence, make sure you’ve removed outliers from your data set prior to using this metric.
6. As compared to mean absolute error, RMSE gives higher weightage and punishes large errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

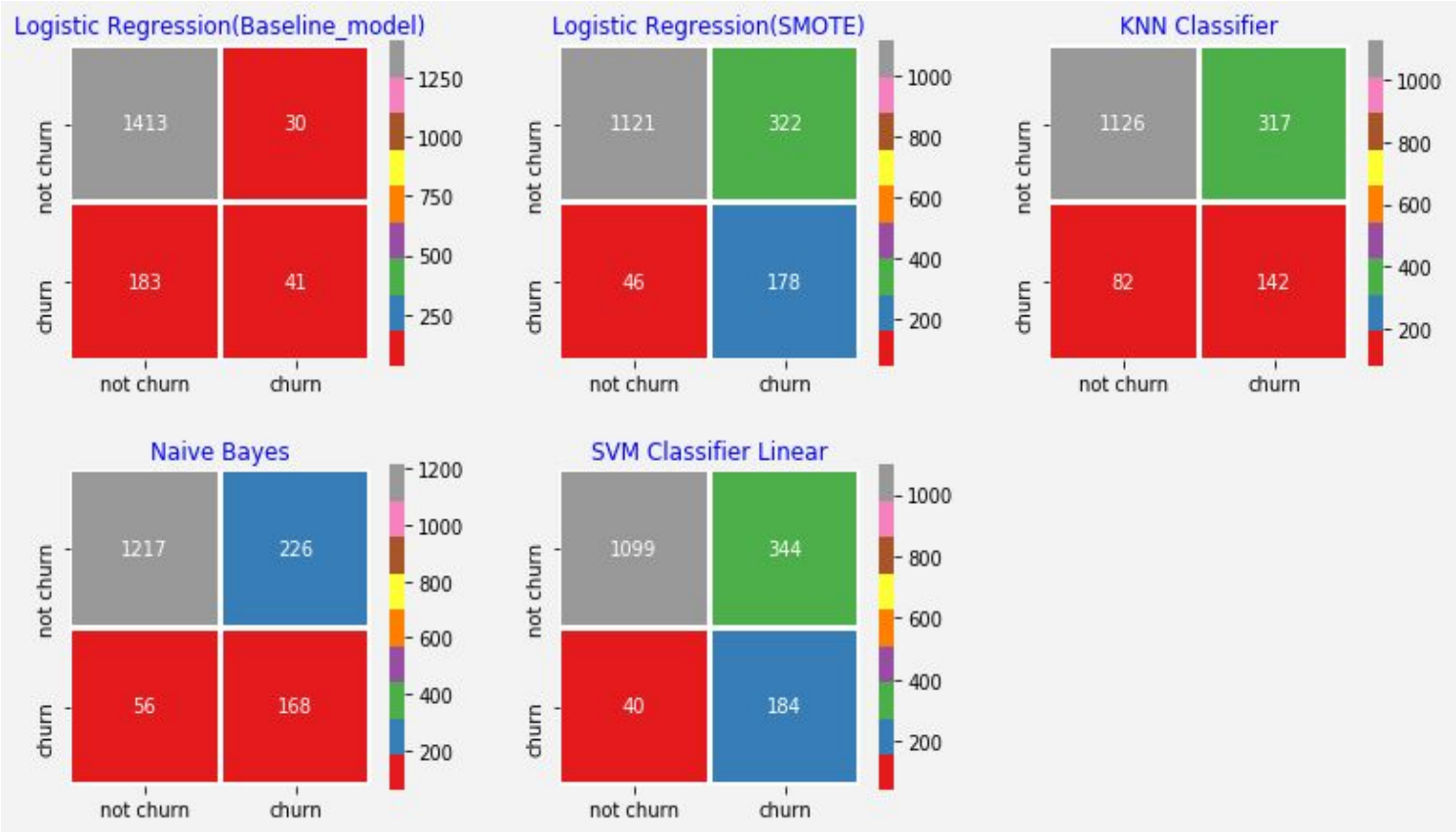
6.2 Compare model metrics

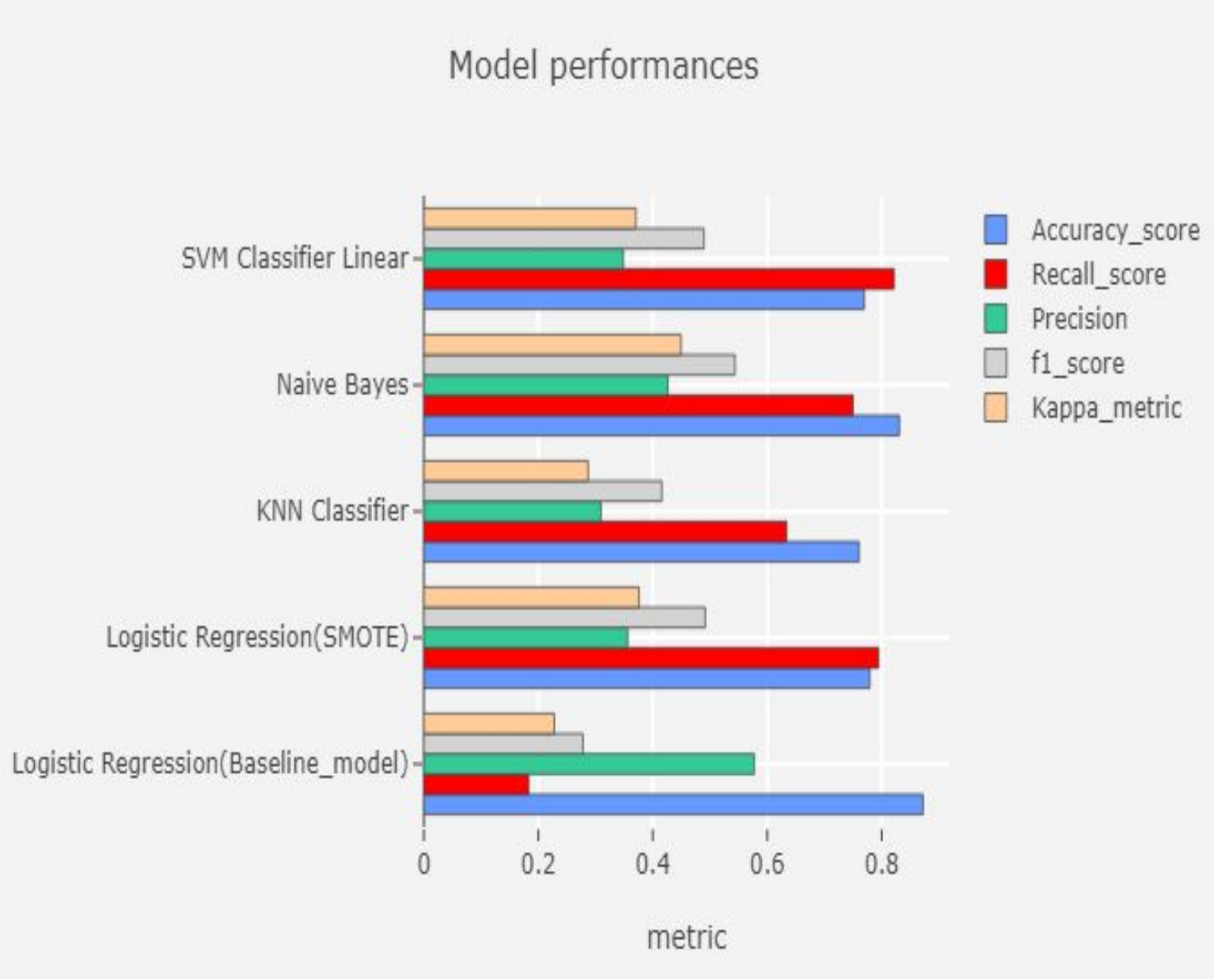
Now we will see and compare Model Performances with respect to the metrics

```
from sklearn.metrics import f1_score
from sklearn.metrics import cohen_kappa_score


model1 = model_report(logit,train_X,test_X,train_Y,test_Y,
                      "Logistic Regression(Baseline_model)")

model2=model_report(logit_smote,os_smote_X,test_X,os_smote_Y,
                          test_Y, "Logistic Regression(SMOTE)")

model3= model_report(knn,os_smote_X,test_X,os_smote_Y,test_Y,
                      "KNN Classifier")

model4= model_report(gnb,os_smote_X,test_X,os_smote_Y,test_Y,
                      "Naive Bayes")

model5= model_report(svc_lin,os_smote_X,test_X,os_smote_Y,test_Y,
                      "SVM Classifier Linear")
```

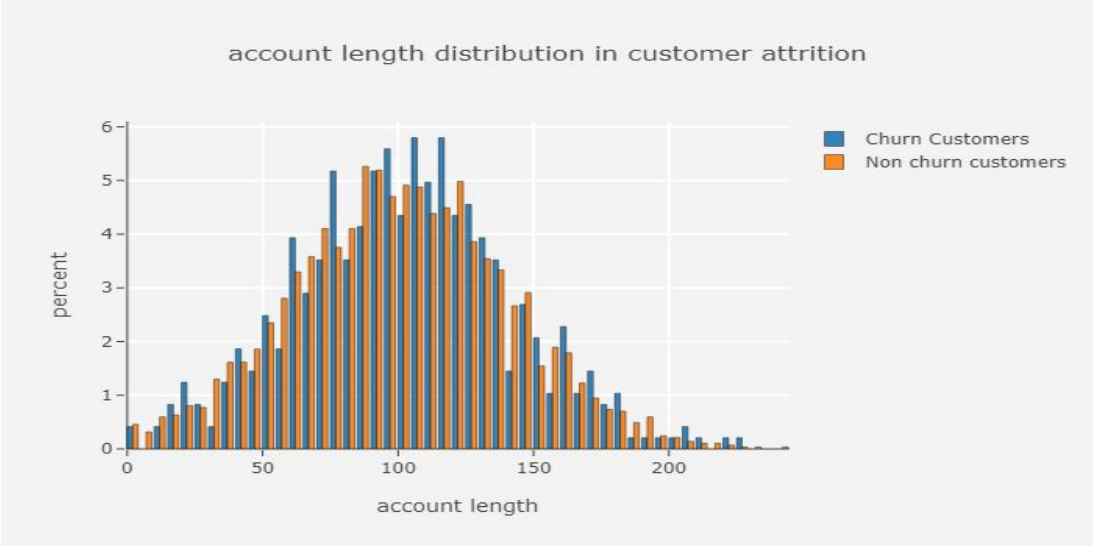
Model	Accuracy_score	Recall_score	Precision	f1_score	Area_under_curve	Kappa_metric
Logistic Regression(Baseline_model)	0.8722	0.183	0.5775	0.278	0.5811	0.228
Logistic Regression(SMOTE)	0.7792	0.7946	0.356	0.4917	0.7857	0.3759
KNN Classifier	0.7606	0.6339	0.3094	0.4158	0.7071	0.287
Naive Bayes	0.8308	0.75	0.4264	0.5437	0.7967	0.4493
SVM Classifier Linear	0.7696	0.8214	0.3485	0.4894	0.7915	0.3706

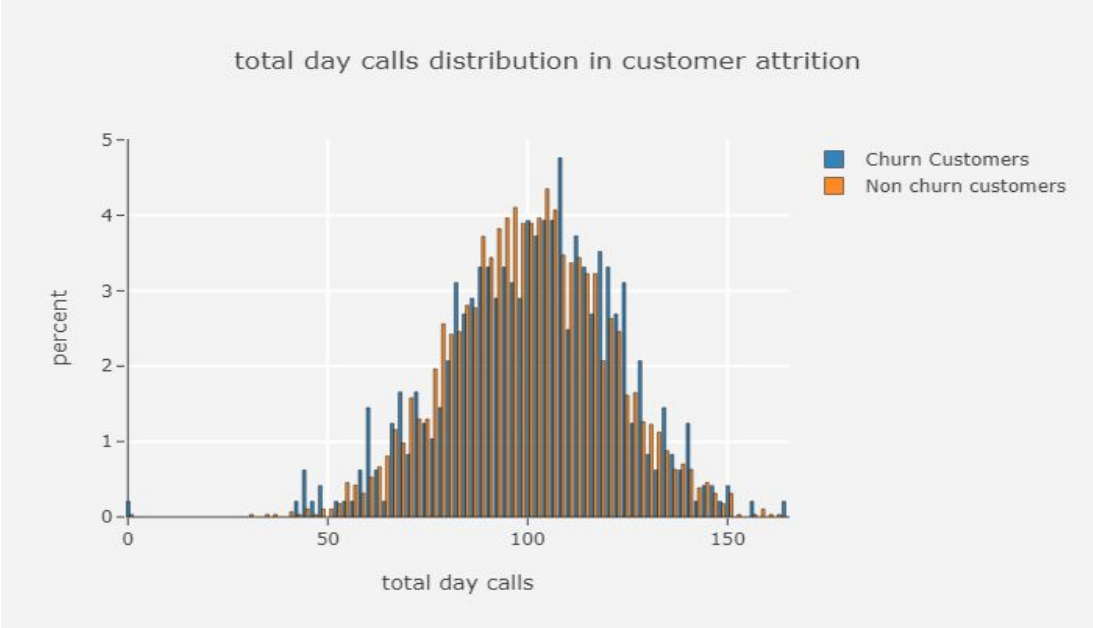
6.3 Confusion matrix Comparison

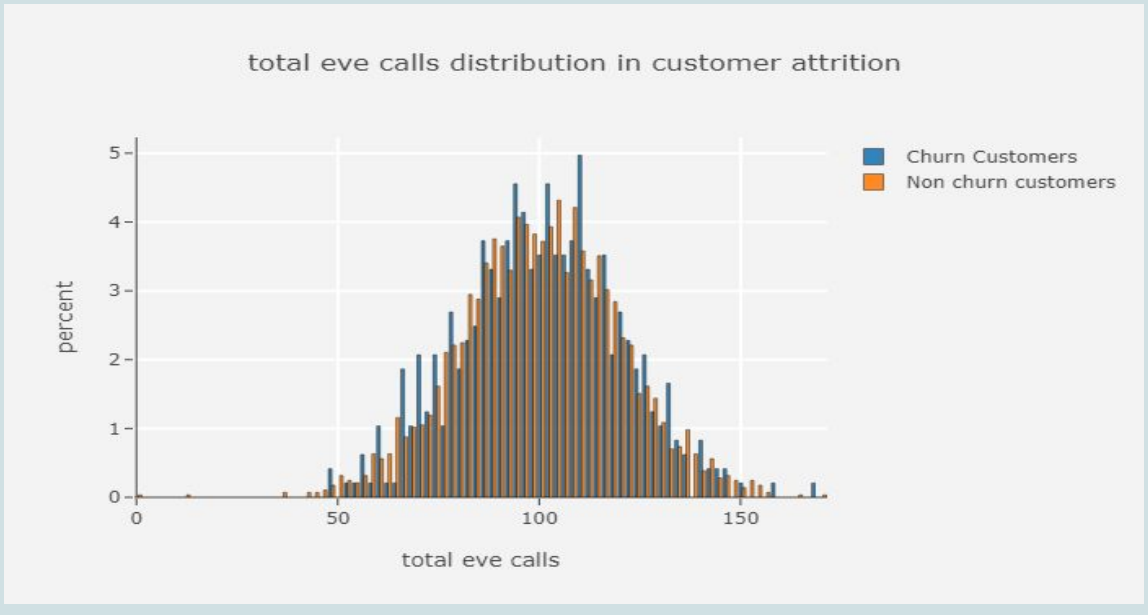


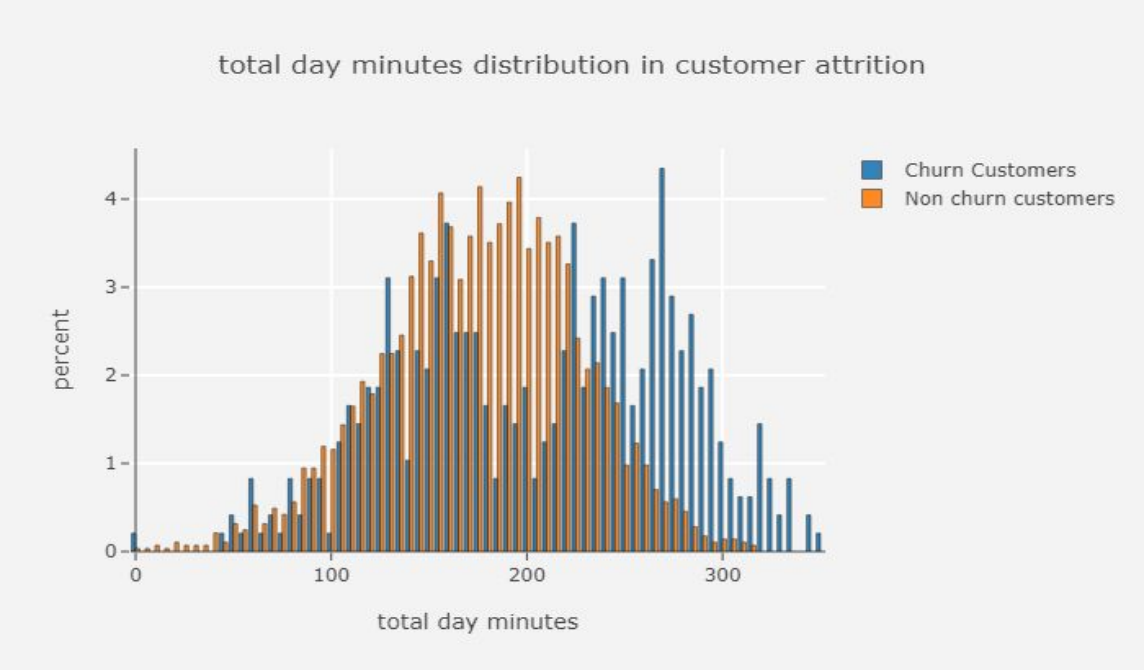

Model performances
SVM Classifier Linear
Naive Bayes
KNN Classifier
Logistic Regression(SMOTE)
Logistic Regression(Baseline_model)
0
0.2
0.4
0.6
0.8
metric
Accuracy_score
Recall_score
Precision
f1_score
Kappa_metric

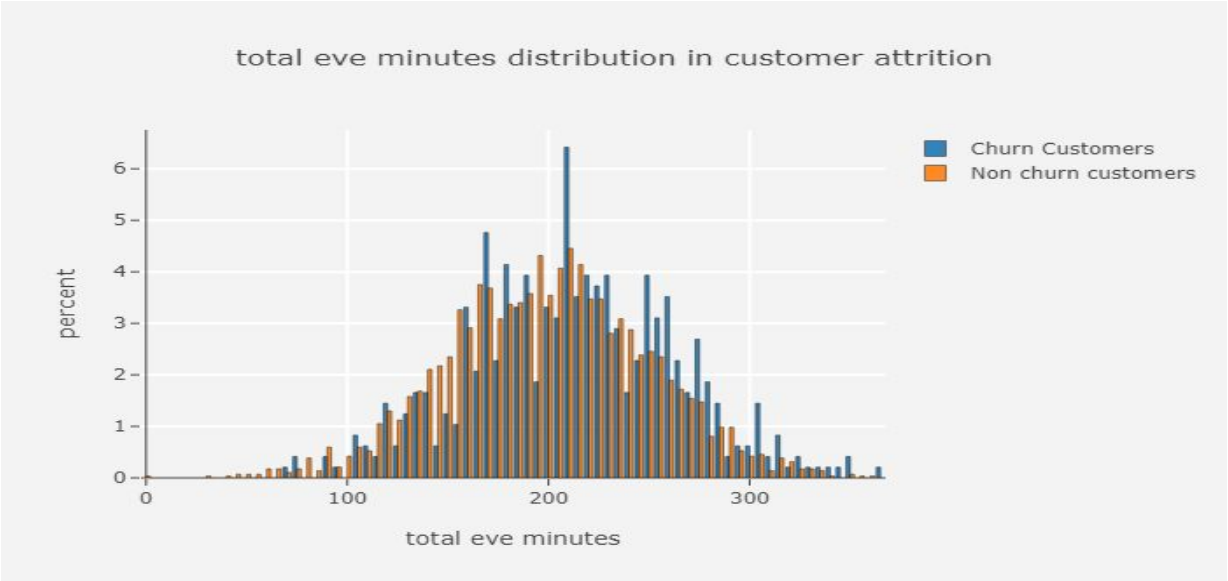
7. Appendix A - Extra Figures

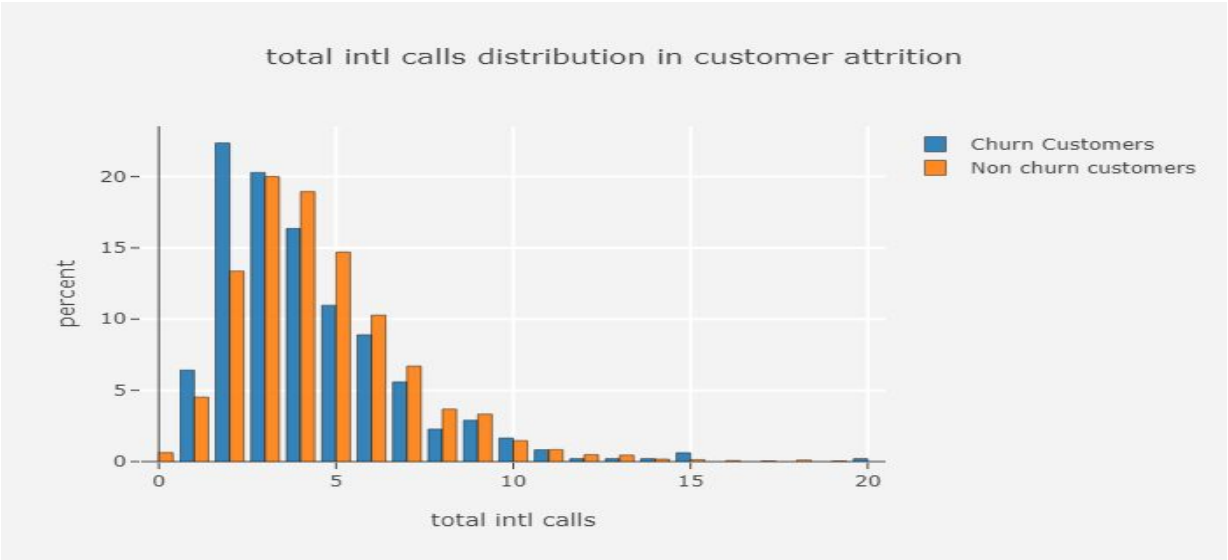


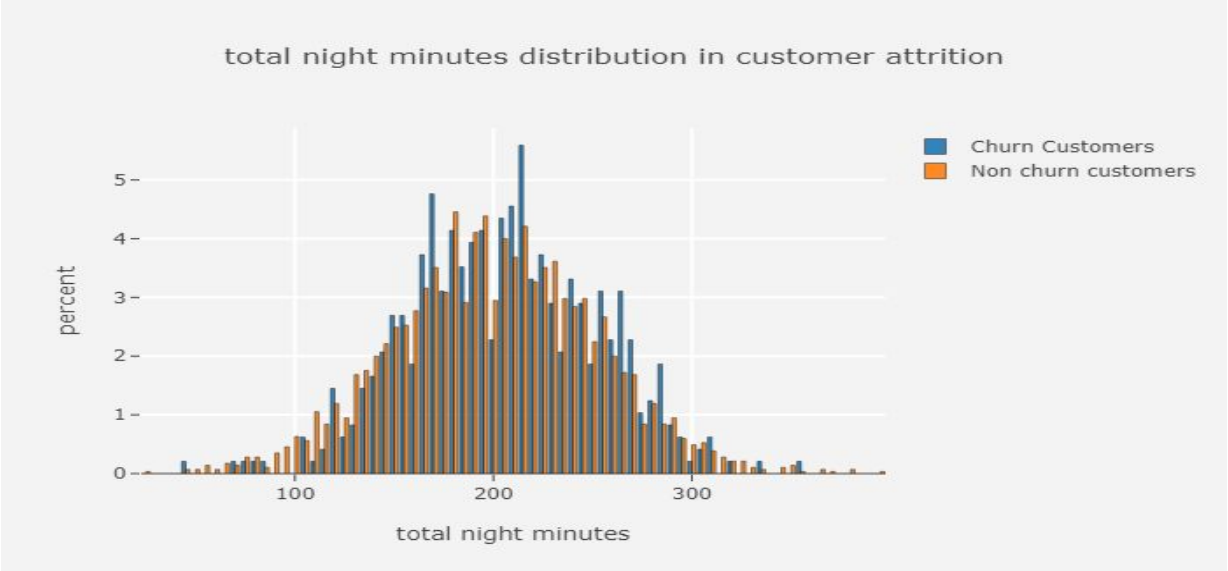



total eve calls distribution in customer attrition
Churn Customers
Non churn customers
percent
0
1
2
3
4
5
0
50
100
150
total eve calls


total day minutes distribution in customer attrition
Churn Customers
Non churn customers
percent
0
1
2
3
4
0
100
200
300
total day minutes


total eve minutes distribution in customer attrition
Churn Customers
Non churn customers
percent
0
1
2
3
4
5
6
0
100
200
300
total eve minutes


total intl calls distribution in customer attrition
Churn Customers
Non churn customers
percent
0
5
10
15
20
0
5
10
15
20
total intl calls


total night minutes distribution in customer attrition
Churn Customers
Non churn customers
percent
0
1
2
3
4
5
100
200
300
total night minutes