Project Report

On

Customer Churn Prediction using

Orange Telecom's Customer Churn Data

Submitted by:
Bhoj Raj Thapa

2021 February

Note: I would be grateful if you take your time to read this report.

1. Introduction

Customer churn is defined as the loss of the customer using the certain telecom service due to the wide range of circumstances such as disliking the data services, call service prices, customer service, or due to the lack of network strength in particular regions etc. I have put the jupyter notebook file in the following link:
https://drive.google.com/file/d/1QrFgZ5FMBcxp_8Rp416QX03l7nh8Wjs1/view?usp=sharing

2. Objective of the Analysis

- To predict if a particular user will churn or not.

3. Scope/Benefits of the Analysis

- Could be helpful on deciding the marketing strategies to lessen the amount of customer churn.
- Could be helpful on deciding services or plans to provide for better customer retention.
- Could be helpful to identify future network issues, competitive threats and at-risk customers.
- Could be helpful to find the trends of services that customers require.

4. The Dataset

This is the Telecom Churn dataset provided by the Orange Telecom Company which I found on Kaggle.com. The dataset was uploaded by the author named Baligh Mnassri. The dataset is divided into two groups in terms of percentage: 80% for the training and cross validation, and 20% for testing and final performance evaluation. The names and shape of the two files are as follows:

i) **churn-bigml-20.csv**
Rows: 2666
Columns: 20

ii) **churn-bigml-80.csv**
Rows: 667
Columns: 20

The columns are as follows:

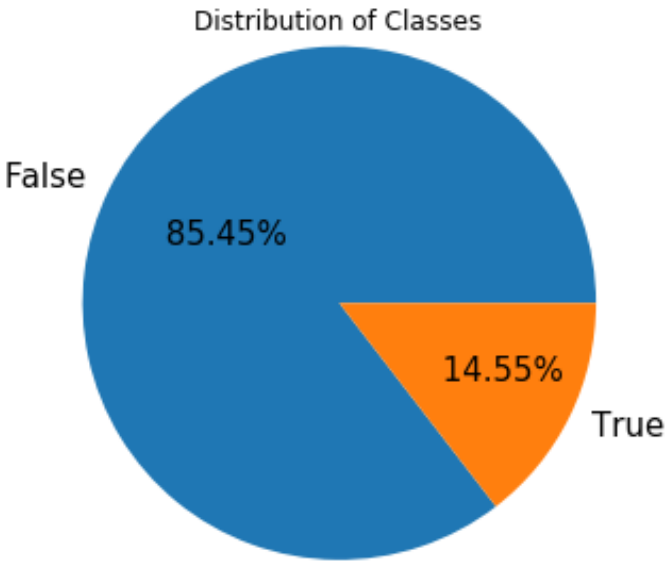
Column Name	**Data Type**
State	String
Account Length	Integer
Area Code	Integer
International Plan	String
Voice Mail Plan	String
Number Vmail Messages	Integer
Total Day Minutes	Double
Total Day Calls	Integer

Total Day Charge	Double
Total Eve Minutes	Double
Total Eve Calls	Integer
Total Eve Charge	Double
Total Night Minutes	Double
Total Night Calls	Integer
Total Night Charge	Double
Total Intl Minutes	Double
Total Intl Calls	Integer
Total Intl Charge	Double
Customer Service Calls	Integer
Churn	String

From the above dataset, our **target column** is the **"Churn"** column which has to be predicted. The dataset only contains the voluntary customer churn. s

5. Data Exploration

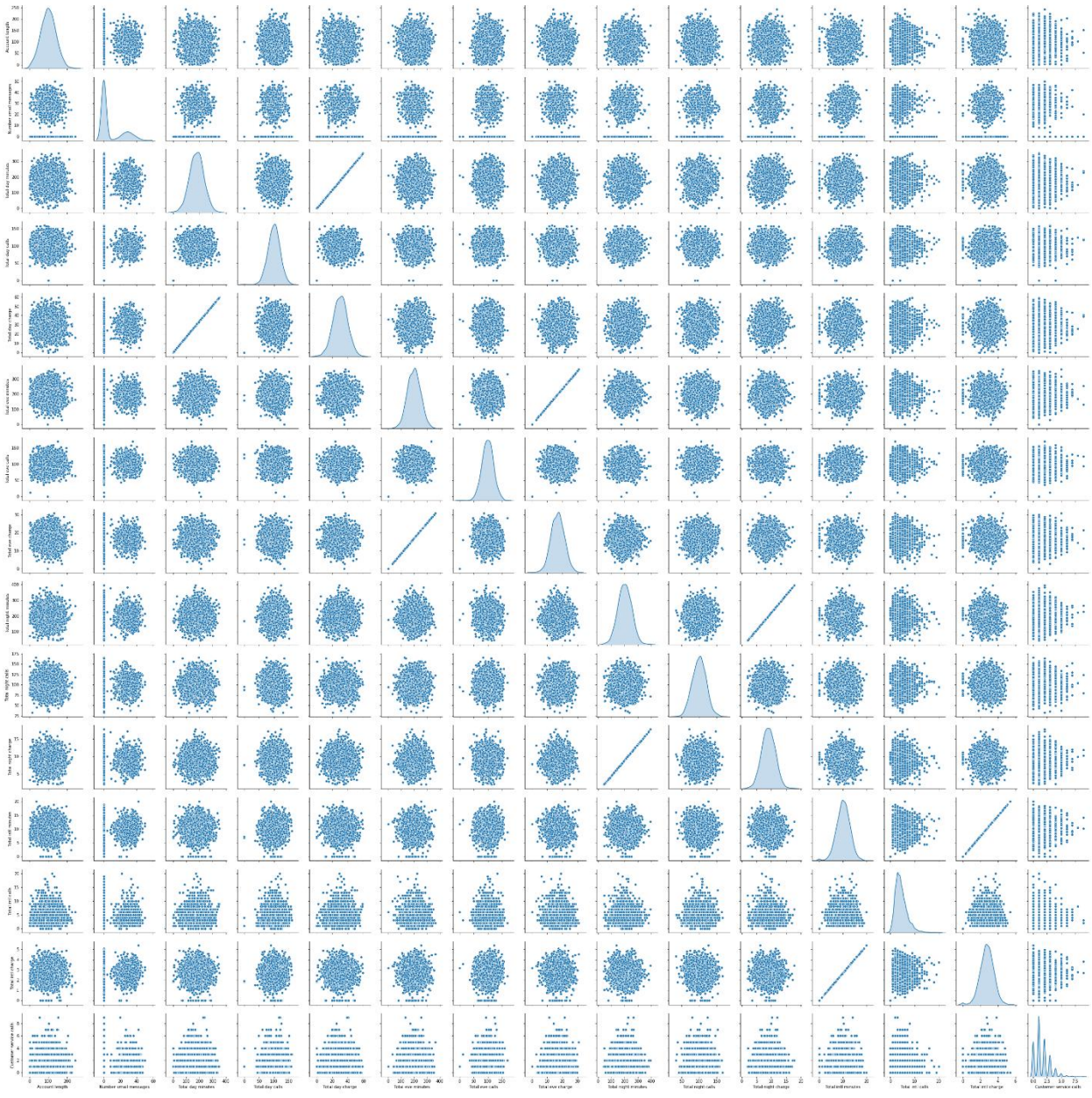
At first, I have to know if the data is well divided in terms of target column, "Churn." For that, I made a pie chart representing the number of different rows (customers) with true value and false value which is given below:



From the above figure, I can say that the number of customers with "Churn" value false is 2278 as compared to that of true value which is 388 out of the total 2666 customers, which means the dataset is skewed.

Then I plotted the pairwise correlation of each feature column in the dataset using pairplot function of seaborn module. I will only use specific features which have numerical values and

meaningful to plot. I removed the five columns for the pairplot: ‘State’, ‘Voice mail plan’, ‘Churn’, ‘Area code’, ‘International plan’. After that, the pairplot could be seen as follows:



The figure might not be as clear to be seen in this report. But I can conclude from the pairplot that four pair of features are highly correlated to each other and they are: i) total day minutes and total day charge, ii) total eve minutes and total eve charge iii) total night minutes and total night charge, and iv) total intl minutes and total intl charge. And I can take one of the above pairs for the feature.

6. Data Preprocessing

For the preprocessing of the data to make ready to train, we have to remove the columns that add no value to the feature space, which are:

- State
- Area code
- Total day charge
- Total eve charge
- Total night charge
- Total intl charge

The next step is to separate the categorical value and numerical value in the refined dataset which contains only 16 features including "Churn." The categorical values were selected based on the column with less than 6 unique values. From the categorical feature columns, feature column with binary value is separated from that of feature columns with more than 2 type of unique values. Although we don't have the feature columns with non-binary values. So, in our dataset, the feature columns are divided as:

1. Categorical Columns: "Churn", "International plan" and "Voice mail plan"
2. Numerical Columns: "Account Length" , "Number vmail messages", "Total day minutes", "Total day calls", "Total eve minutes", "Total eve calls", "Total night minutes", "Total night calls", "Total intl calls", "Total intl calls", and "Customer service calls".

Then the categorical columns were encoded using LabelEncoder function of sklearn module. After that, scaling was done using the StandardScaler method of sklearn module. The reason we do scaling is to remove the high varying data (high variance between the data features). And then the unscaled data from the dataset was replaced by the scaled dataset for the Numerical Columns. Preprocessing was done for the both training and test dataset. And the final dataset after preprocessing could be seen as follows:

	International plan	Voice mail plan	Churn	Account length	Number vmail messages	Total day minutes	Total day calls	Total eve minutes	Total eve calls	Total night minutes	Total night calls	Total intl minutes	Total intl calls	Customer service calls
0	0	0	0	0.347127	-0.601245	0.064036	-0.193167	2.983872	-1.081478	0.324092	-0.501749	-0.548297	-0.212747	-0.423098
1	0	0	1	-0.927731	-0.601245	-0.934756	1.769398	0.506113	-0.923033	0.183311	0.540053	0.877350	0.593516	1.828550
2	0	0	1	1.425853	-0.601245	2.739500	-1.665090	2.303545	-0.183625	-0.786067	1.383417	-1.724455	1.802910	1.828550
3	0	0	0	0.200028	-0.601245	-1.271894	0.101218	-1.329563	0.080450	-0.202831	0.242395	-0.904709	0.593516	0.327451
4	0	0	0	-1.319994	-0.601245	-1.111438	0.788115	0.236397	0.450154	-0.422048	-0.501749	0.307091	-1.422141	-0.423098

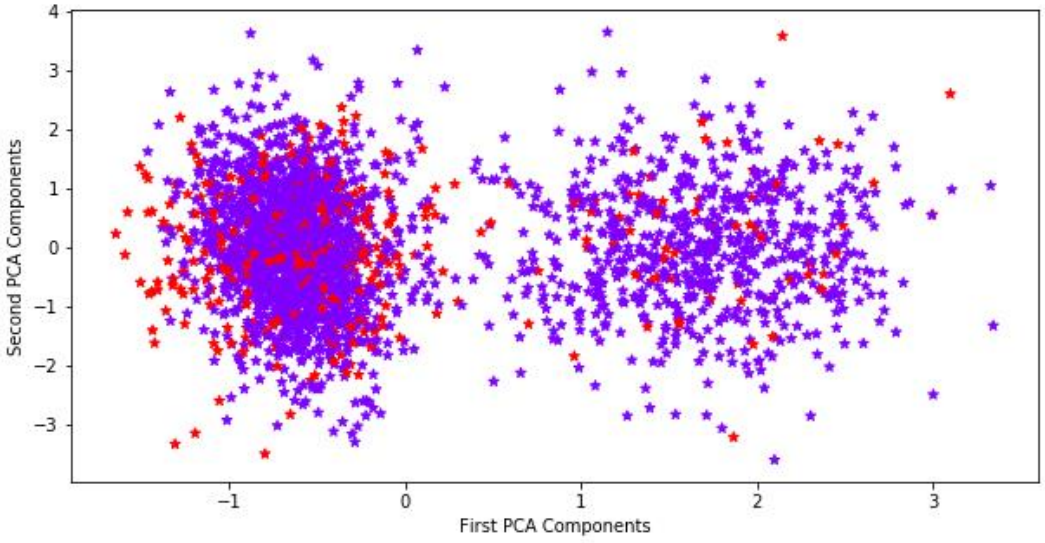
And the summary of this dataset looked like:

feature	count	mean	std	min	25%	50%	75%	max
International plan	2666.0	0.10	0.30	0.00	0.00	0.00	0.00	1.00
Voice mail plan	2666.0	0.27	0.45	0.00	0.00	0.00	1.00	1.00
Churn	2666.0	0.15	0.35	0.00	0.00	0.00	0.00	1.00
Account length	2666.0	-0.00	1.00	-2.52	-0.70	-0.02	0.67	3.60
Number vmail messages	2666.0	-0.00	1.00	-0.59	-0.59	-0.59	0.81	3.08
Total day minutes	2666.0	0.00	1.00	-3.31	-0.67	0.01	0.67	3.16
Total day calls	2666.0	0.00	1.00	-5.02	-0.67	0.03	0.69	2.99
Total eve minutes	2666.0	-0.00	1.00	-3.93	-0.69	0.01	0.68	3.21
Total eve calls	2666.0	0.00	1.00	-4.96	-0.65	-0.00	0.69	3.47
Total night minutes	2666.0	0.00	1.00	-3.10	-0.67	-0.00	0.70	3.82
Total night calls	2666.0	-0.00	1.00	-3.46	-0.68	-0.01	0.66	3.39
Total intl minutes	2666.0	-0.00	1.00	-3.67	-0.62	-0.01	0.67	3.50
Total intl calls	2666.0	-0.00	1.00	-1.82	-0.60	-0.19	0.62	6.33
Customer service calls	2666.0	0.00	1.00	-1.19	-0.43	-0.43	0.33	5.67

From the above figure of summary, we can say that they dataset is well preprocessed and encoded for the modeling.

6.1 Data Analysis using Principle Component Analysis

After transforming the preprocessed data using the PCA, the figure below could be observed:



In the above figure, the red dots represent PCA components of “churn” data and purple dots represent PCA components of “not churn” data. From the figure, we can’t clearly separate the dataset in two classes. It seems like there is a division of the dataset in two clusters but not necessarily the division in terms of target classes.

7. Building the Model

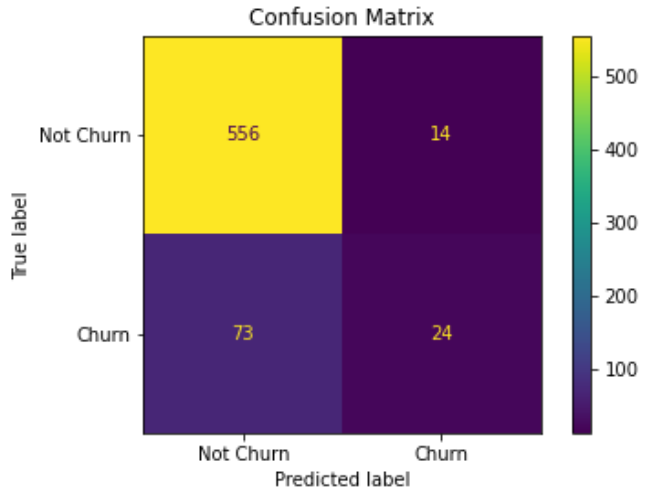
I made a function called “churn_prediction_training” which need 5 arguments: object of the classification algorithm, x_test data, y_test data, x_train data, y_train data. Following metrics were observed in each model:

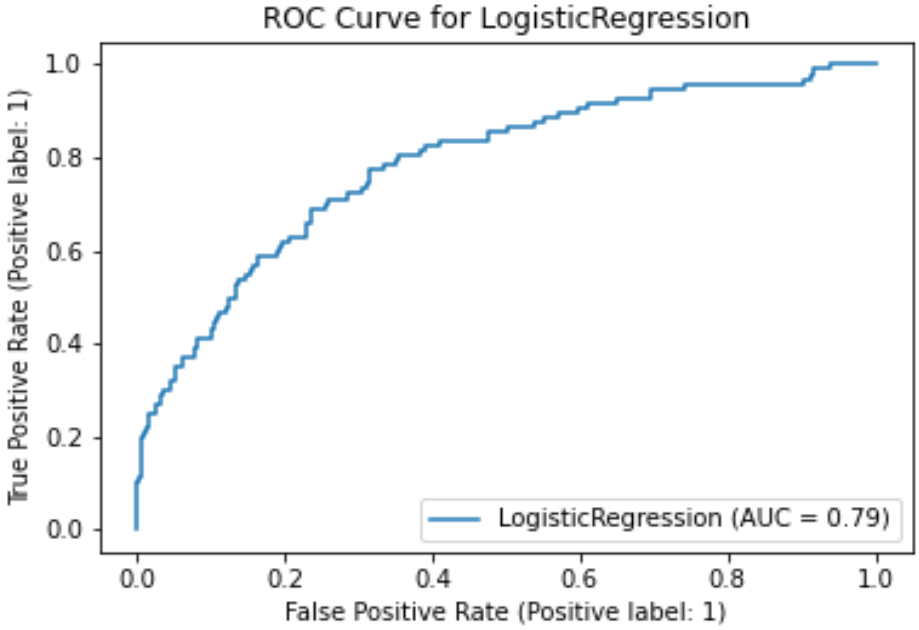
- Accuracy of the classification method : Accuracy is defined as the percentage of correct predictions for the test data.
- Area under the curve (AUC): Area under the curve is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model as distinguishing between the positive and negative classes.
- Precision and recall for the both the classes: Precision quantifies the number of positive class predictions that belong to the positive class whereas recall quantifies the number of positive class prediction made out of all positive examples in the dataset.
- Confusion matrix for the training set: Confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically supervised machine learning algorithm.
- ROC curve: ROC curve is a graph showing the performance of the classification model at all classification thresholds which is plotted between true positive rate and false positive rate.

7.1 Baseline Model

Then I used the function (churn_prediction_training) to classify the training dataset using Logistic Regression which is also the baseline model for our analysis. Then we got the following output.

```
Algorithm: LogisticRegression

Classification Report:
               precision    recall  f1-score   support

   Not Churn       0.88      0.98      0.93       570
       Churn       0.63      0.25      0.36        97

    accuracy                           0.87       667
   macro avg       0.76      0.61      0.64       667
weighted avg       0.85      0.87      0.84       667

Accuracy Score: 0.8695652173913043
Area Under the Curve: 0.7863266413456322
```



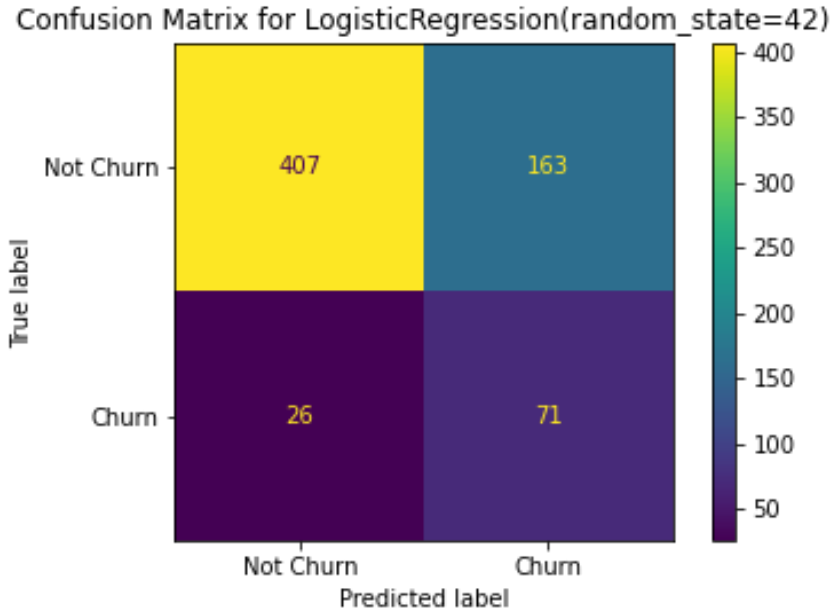


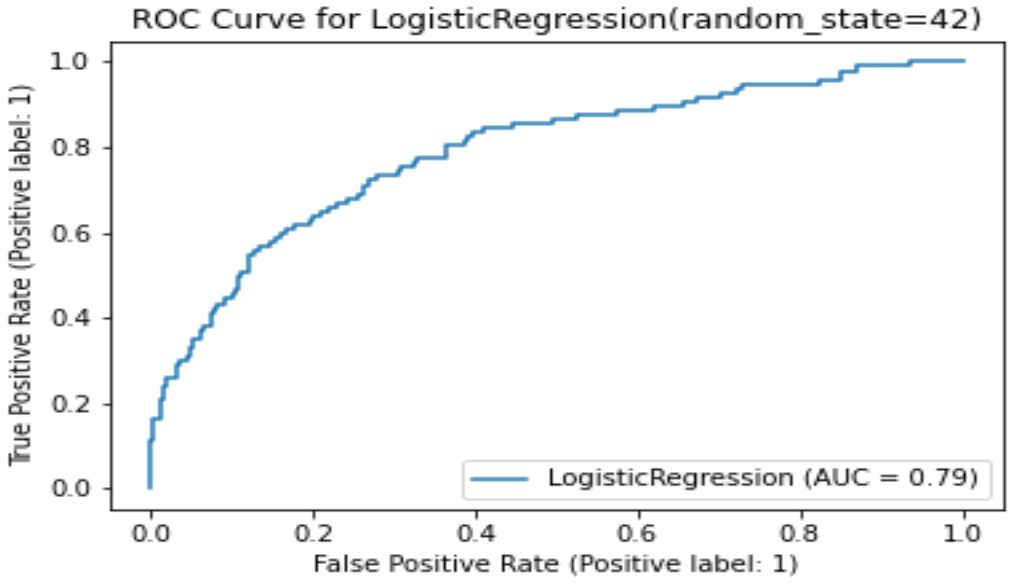
From the above table, we can see that the accuracy is comparatively lower. Even though, for the not churn samples, the recall, precision and f1 score is considerably higher but is not the same for churn samples. The report also described the confusion matrix as well. In the confusion matrix, out of 570 not churned samples, the model predicted 556 samples right and 14 samples wrong. And out of 97 churned samples, the model predicted 73 samples right and 24 samples wrong. The ROC curve presents the tradeoff between the sensitivity and specificity. If the area under the curve is 1, the model is best and it is around 0.5 the model is not good. In our case, the area under the curve is 0.61 which is not so good for any classification model. The f1 score for our skewed dataset seems little lower for churn class which is not good for the model because the model will not generalize well with this performance. The reason behind the low metrics of the model might be the skewed dataset. Therefore, we will try to resample the dataset and then train and test the dataset using other classification models.

7.2 Over Sampling the Dataset

Since our dataset is skewed or unbalanced, we will oversample the dataset using adaptive synthetic algorithm (ADASYNC) to make the number of samples for two classes equal. After training the Logistic Regression Model with oversampled dataset using adasyn function, following classification metrics could be seen:

```
Algorithm: LogisticRegression

Classification Report:
              precision    recall  f1-score   support

   Not Churn       0.94      0.71      0.81       570
       Churn       0.30      0.73      0.43        97

    accuracy                           0.72       667
   macro avg       0.62      0.72      0.62       667
weighted avg       0.85      0.72      0.76       667

Accuracy Score: 0.7166416791604198
Area Under the Curve: 0.7894194248507866
```





From the above report and confusion matrix, we can see that the accuracy score is decreased. But the recall value for churned class is increased. Precision microaverage is also decreases which means our model is less likely to predict 1 as 1 or 0 as 0. But the recall micro average is decreased which means the model is less likely to classify all the samples as their original class.

Since, oversampling did not refined the accuracy and other performance metrics of the model, we will not use oversampled data in the later phase.

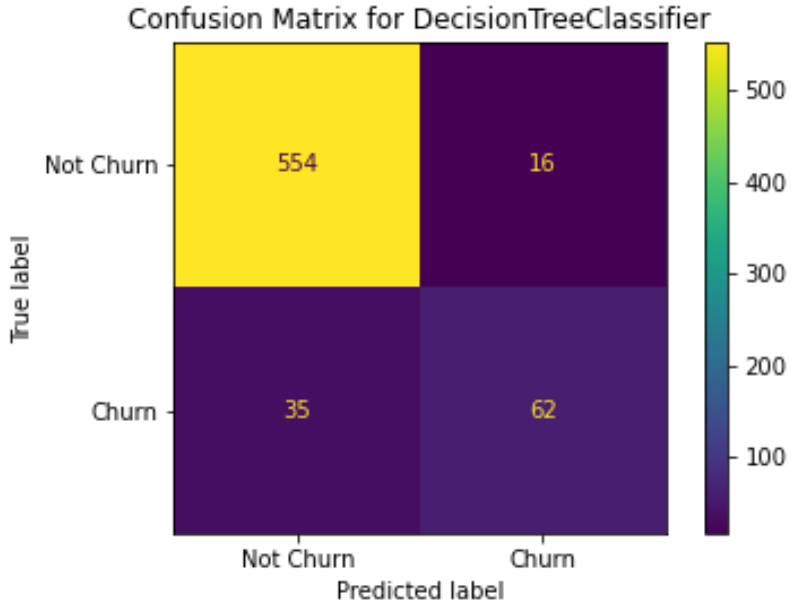
7.3 Decision Tree Classifier

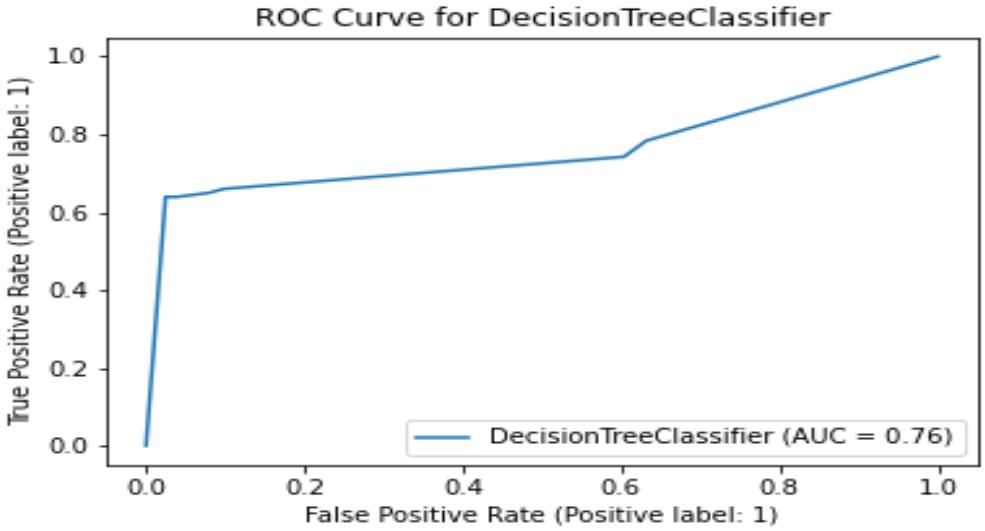
I also tried decision tree classifier with maximum depth of 9, gini criterion, and best splitting. And after the training, the performance of the model could be seen as follows:

```
Algorithm: DecisionTreeClassifier

Classification Report:
               precision    recall  f1-score   support

   Not Churn       0.94      0.97      0.96       570
       Churn       0.79      0.64      0.71        97

    accuracy                           0.92       667
   macro avg       0.87      0.81      0.83       667
weighted avg       0.92      0.92      0.92       667

Accuracy Score: 0.9235382308845578
Area Under the Curve: 0.7596129499005244
```

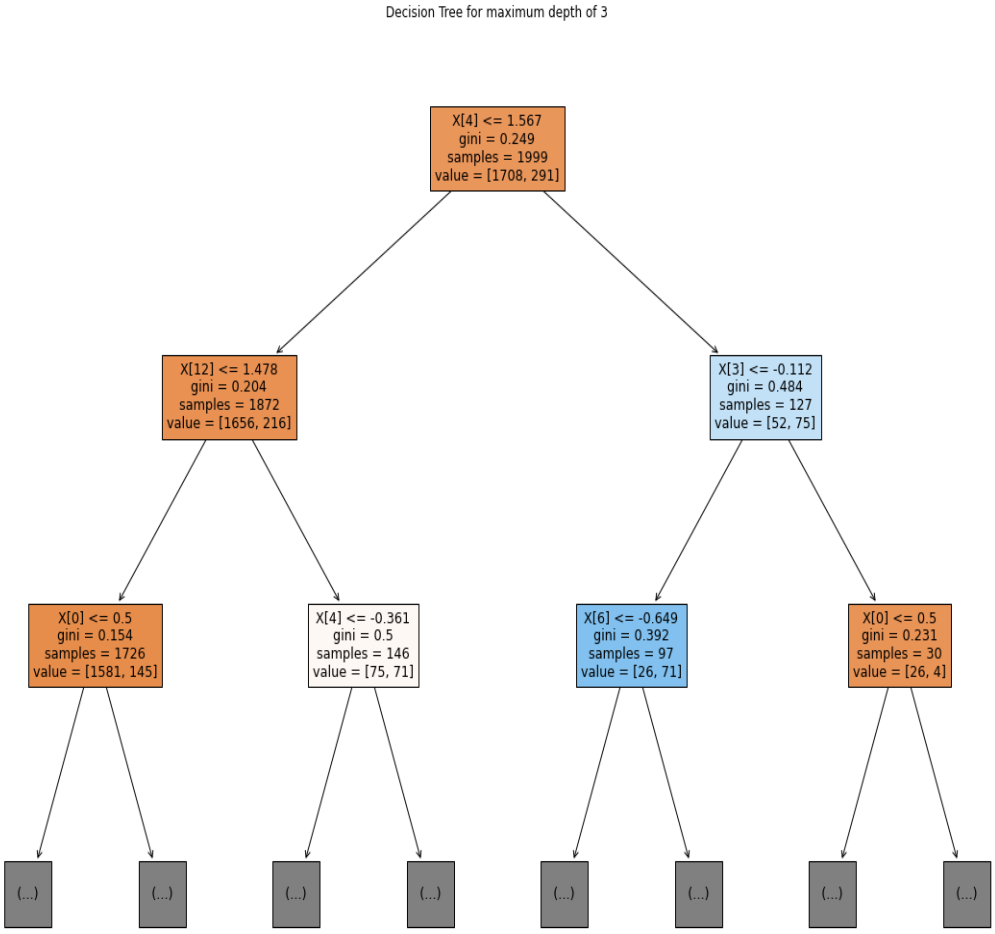
From the above classification report, we can see that the accuracy is increased as compared to logistic regression classifier model and the area under the curve is also increased. Better the auc

score, better the performance of the model. The precision, recall and f1 score increased significantly. This implies the better overall performance of the classification model.





In the above confusion matrix, recall for churn class is decreased which can be seen by 35 wrong prediction of the churn samples.

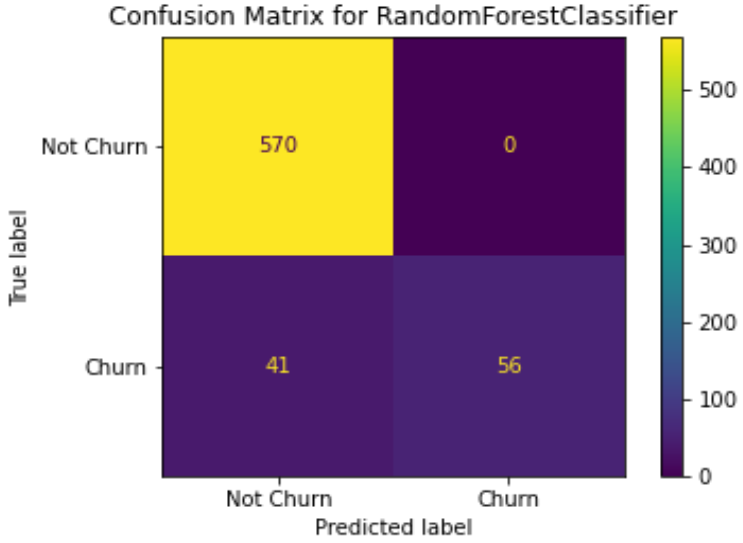


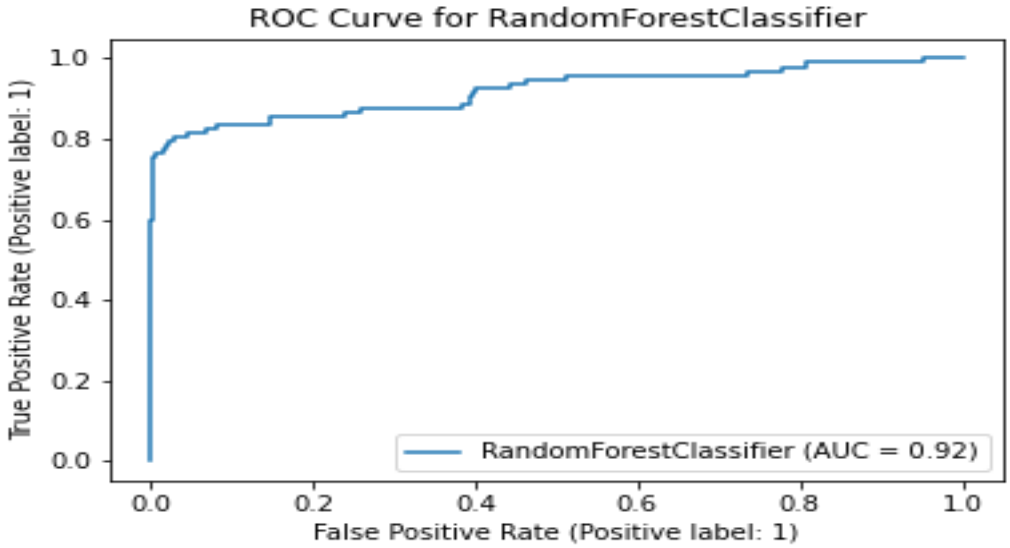
The above tree is the decision tree of the classification model mentioned above.

7.4 Random Forest Classifier

Random forest classifier is a meta-estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The performance metrics for Random Forest Classifier are as follows:

```
Algorithm: RandomForestClassifier

Classification Report:
               precision    recall  f1-score   support

   Not Churn       0.93      1.00      0.97       570
       Churn       1.00      0.58      0.73        97

    accuracy                           0.94       667
   macro avg       0.97      0.79      0.85       667
weighted avg       0.94      0.94      0.93       667

Accuracy Score: 0.9385307346326837
Area Under the Curve: 0.919714234038705
```





In the above classification report, we can see that the overall accuracy was improved to 0.94 whereas recall was not up to the mark. Recall for the negative class was 100 percent but for the positive class, recall was 0.58. And the precision for positive class was 100 percent. And the area under the curve is around 0.92 which is pretty good as compared to LR and Decision Tree Classifier.

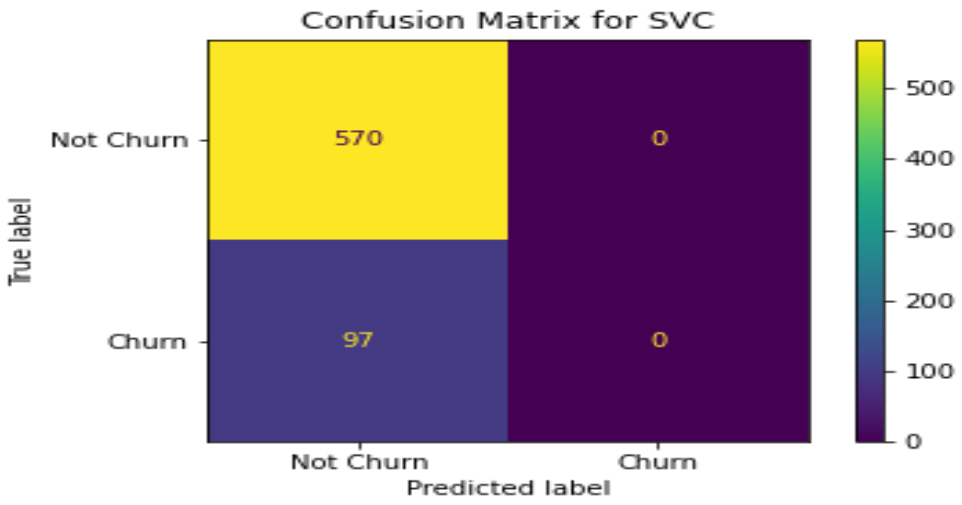
7.5 Support Vector Machine

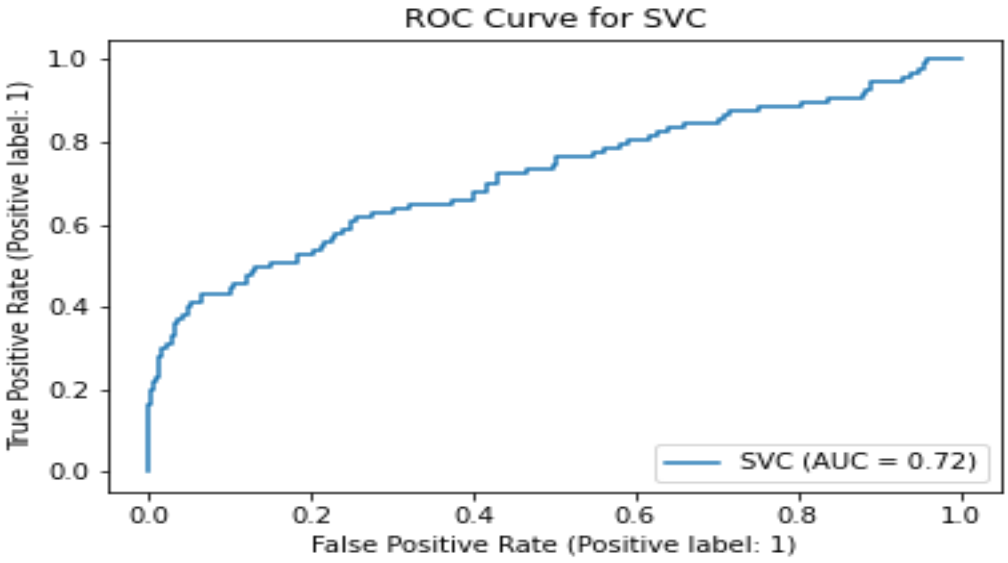
Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression problems. It is mostly used for classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then we perform classification by finding the hyper-plane that differentiate the two classes. We will use two different kernels for SVM.

7.5.1 Linear Kernel

The performance metrics of the SVM model using linear kernel are as follows:

```
Algorithm: SVC

Classification Report:
                precision    recall  f1-score   support

    Not Churn       0.85      1.00      0.92       570
        Churn       0.00      0.00      0.00        97

     accuracy                           0.85       667
    macro avg       0.43      0.50      0.46       667
weighted avg       0.73      0.85      0.79       667

Accuracy Score: 0.8545727136431784
Area Under the Curve: 0.7183396635919695
```



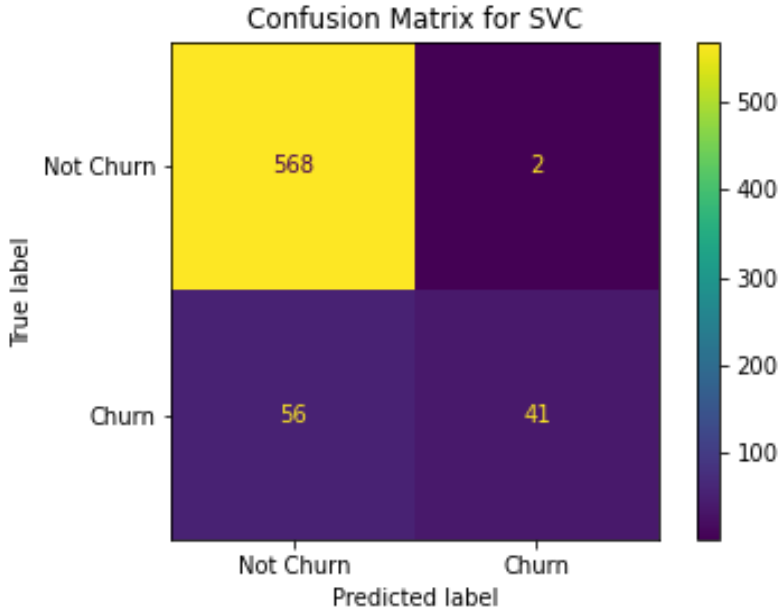


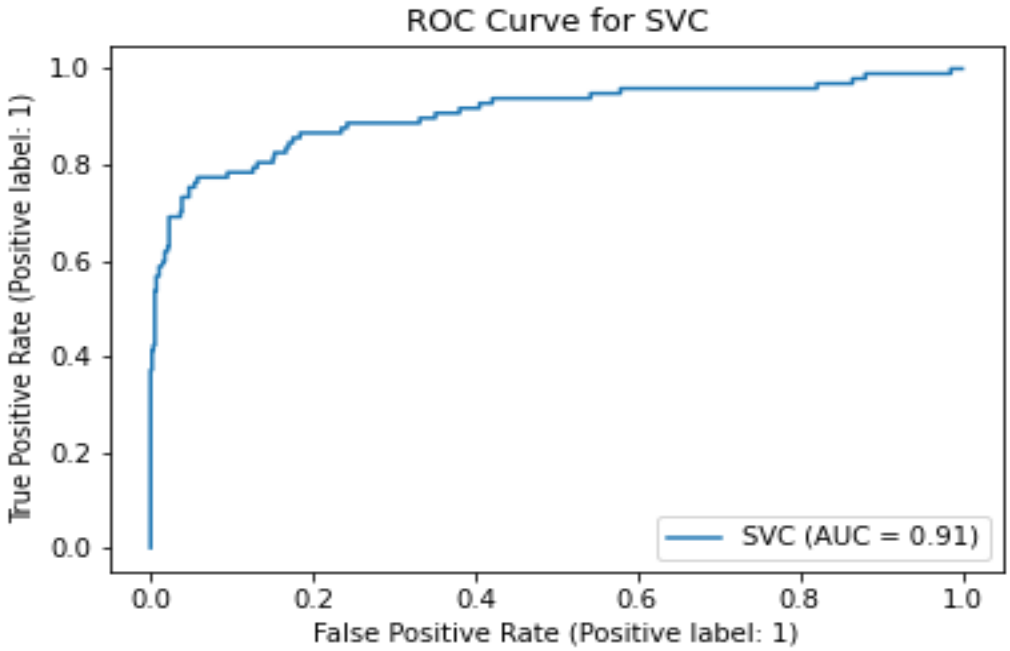
From the above classification report and confusion matrix, we can see that the precision and recall for churn class is 0 which is very bad. And the model did not predict well for churn class which is the model predicted 0 samples as churn out of 97 churn samples. Even though the area under the curve and accuracy seems okay but the model is very bad for this application since the precision, recall and f1 score is quite bad.

7.5.2 RBF kernel

Now, I will use rbf kernel with gamme value of 0.1. And the performance metrics for this model are as follows:

```
Algorithm: SVC

Classification Report:
               precision    recall  f1-score   support

   Not Churn       0.91      1.00      0.95       570
       Churn       0.95      0.42      0.59        97

    accuracy                           0.91       667
   macro avg       0.93      0.71      0.77       667
weighted avg       0.92      0.91      0.90       667

Accuracy Score: 0.9130434782608695
Area Under the Curve: 0.9064839934888769
```



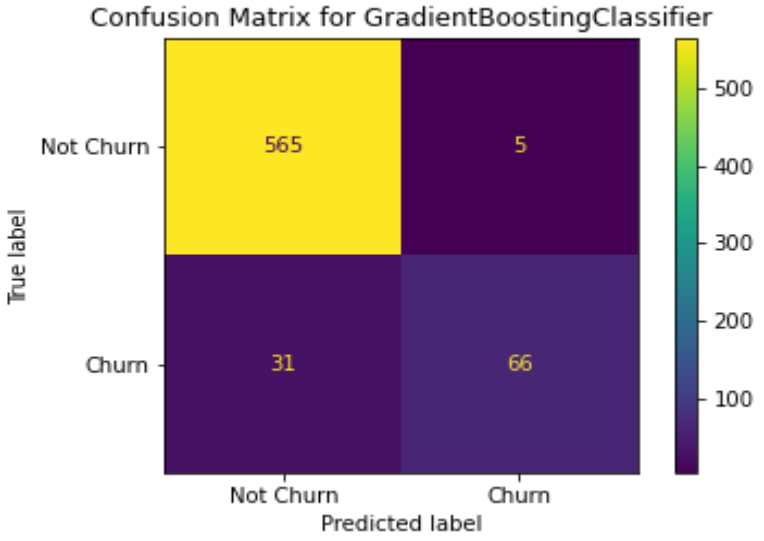


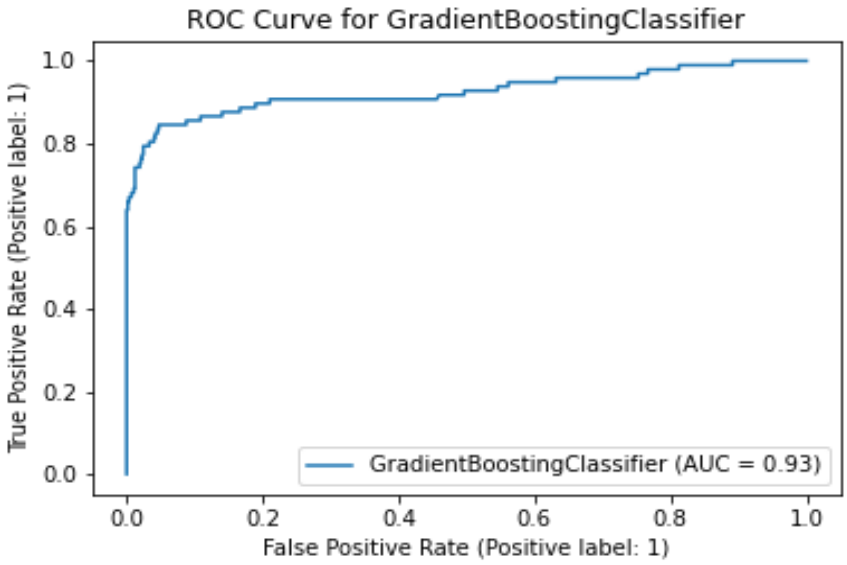
From the above classification report and confusion matrix, it could be seen that the model with rbf kernel is better than the svm with linear kernel. The area under the curve seems significantly higher than svm with linear kernel. In this model, recall and precision for churn class is better but is not up to the mark. And the f1 score is 0.9 which seems okay.

7.6 Gradient Boosting Classifier (Ensemble Method)

I trained the model using gradient boosting classifier of sklearn module using all the default values of loss, learning rate, and criterion. And the performance metrics could be seen as follows:

```
Algorithm: GradientBoostingClassifier

Classification Report:
               precision    recall  f1-score   support

   Not Churn       0.95      0.99      0.97       570
       Churn       0.93      0.68      0.79        97

    accuracy                           0.95       667
   macro avg       0.94      0.84      0.88       667
weighted avg       0.95      0.95      0.94       667

Accuracy Score: 0.9460269865067467
Area Under the Curve: 0.9255018990775908
```





From the above classification report, we can see that all the metrics like recall, precision and f1 score improved. The area under the curve is also increased as a whole as compared to other classifiers.

7.6 Comparison Table of all the above models

Model Name	Accuracy	Precision (Weighted Average)	Recall (Weighted Average)	F1 Score (Weighted Average)	Area under the ROC Curve
Logistic Regression (LR)	0.87	0.85	0.87	0.84	0.79
LR with ADASYN	0.72	0.85	0.72	0.76	0.79
Decision Tree	0.92	0.72	0.92	0.92	0.76
Random Forest	0.94	0.94	0.94	0.93	0.92
SVM with linear kernel	0.85	0.73	0.85	0.79	0.72
SVM with RBF	0.91	0.92	0.91	0.9	0.91
Gradient Boosting	0.95	0.95	0.95	0.94	0.93

From the above comparison table of all the classification model trained on a split of training dataset and tested on the test split of the same dataset, we could see that the performance of Random Forest Classifier, SVM with RBF and Gradient Boosting Classifier are close to each other and close to 100 percent in all metrics.

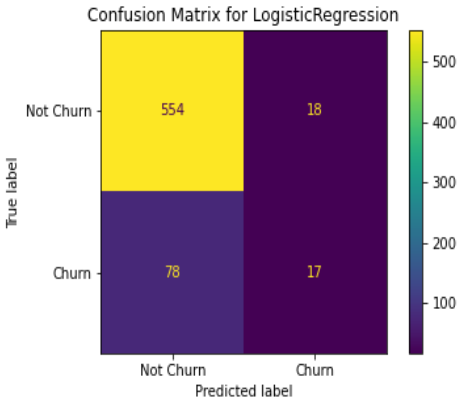
8. Performance metrics for the principle test dataset

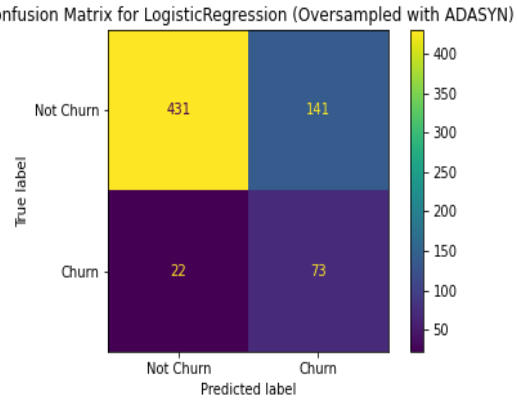
For the testing of principle test dataset, we used pre-processed principle training dataset of size 2666*13 and its target values to train, and preprocessed testing dataset of size 667*13 and its target values to test. Previously, classification was done using the train_test_split function of sklearn on principle training dataset.

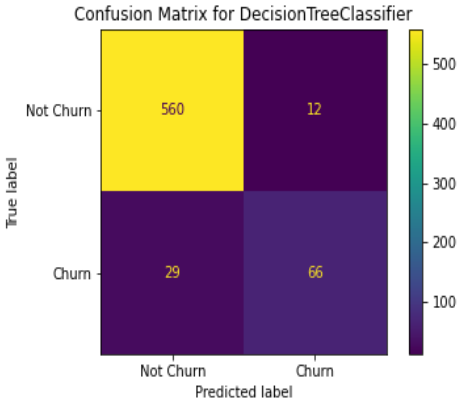
After training and testing the models with above mentioned dataset, the performance metrics could be seen as follows:

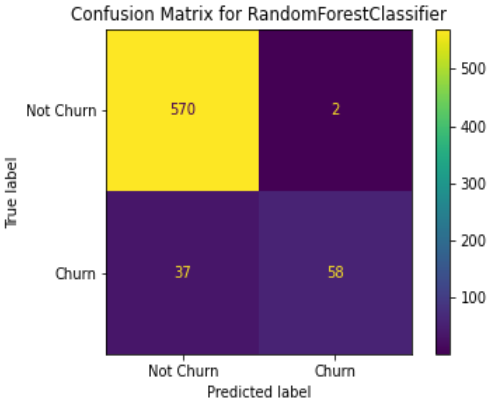
Model Name	Accuracy	Area under the ROC Curve	Precision		Recall		F1 Score	
			Not Churn	Churn	Not Churn	Churn	Not Churn	Churn
Logistic Regression (LR)	0.86	0.83	0.88	0.49	0.97	0.18	0.92	0.26
LR with ADASYN	0.76	0.82	0.95	0.34	0.75	0.77	0.84	0.47
Decision Tree	0.94	0.82	0.95	0.85	0.98	0.69	0.96	0.76
Random Forest	0.94	0.92	0.94	0.97	1	0.61	0.97	0.75
SVM with linear kernel	0.86	0.78	0.86	0.00	1	0.00	0.92	0.00
SVM with RBF	0.91	0.93	0.91	0.90	0.99	0.40	0.95	0.55
Gradient Boosting	0.95	0.93	0.95	0.94	0.99	0.66	0.97	0.78

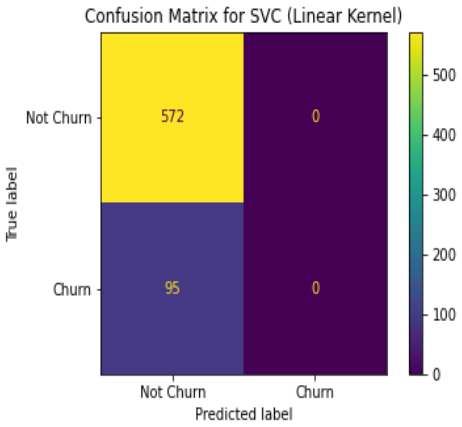
And the confusion matrixes for all the models are as follows:

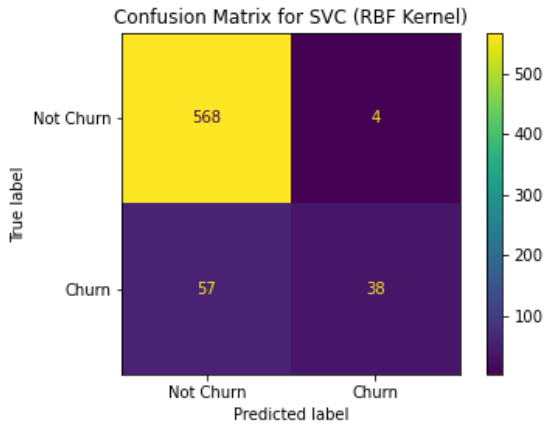

Confusion Matrix for LogisticRegression
Not Churn
Churn
True label
554
18
78
17
Not Churn
Churn
Predicted label
500
400
300
200
100

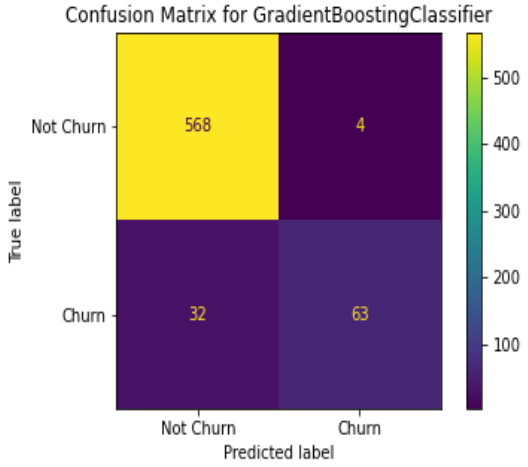

Confusion Matrix for LogisticRegression (Oversampled with ADASYN)
Not Churn
Churn
True label
431
141
22
73
Not Churn
Churn
Predicted label
400
350
300
250
200
150
100
50


Confusion Matrix for DecisionTreeClassifier
Not Churn
Churn
True label
560
12
29
66
Not Churn
Churn
Predicted label
500
400
300
200
100

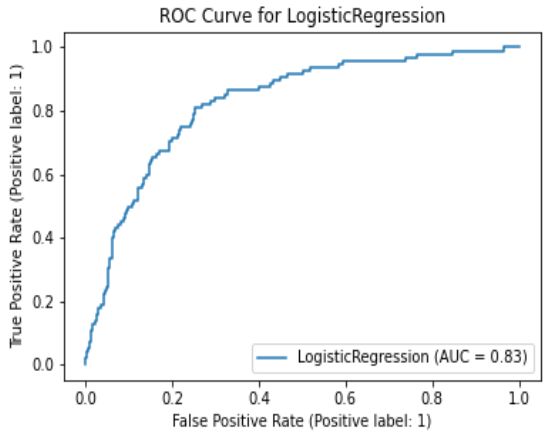

Confusion Matrix for RandomForestClassifier
Not Churn
Churn
True label
570
2
37
58
Not Churn
Churn
Predicted label
500
400
300
200
100

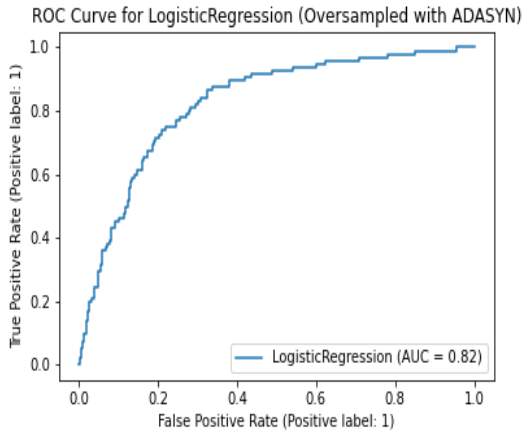

Confusion Matrix for SVC (Linear Kernel)
Not Churn
Churn
True label
572
0
95
0
Not Churn
Churn
Predicted label
500
400
300
200
100
0

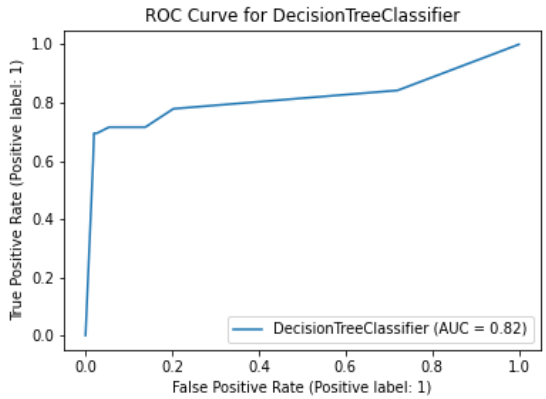

Confusion Matrix for SVC (RBF Kernel)
Not Churn
Churn
True label
568
4
57
38
Not Churn
Churn
Predicted label
500
400
300
200
100

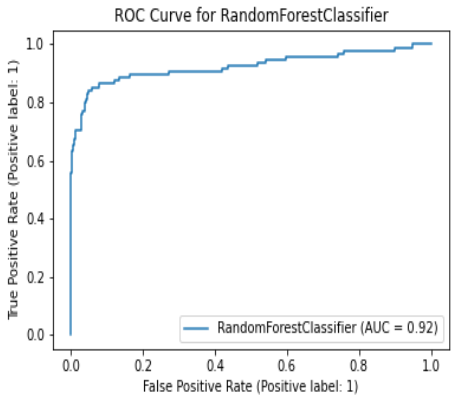


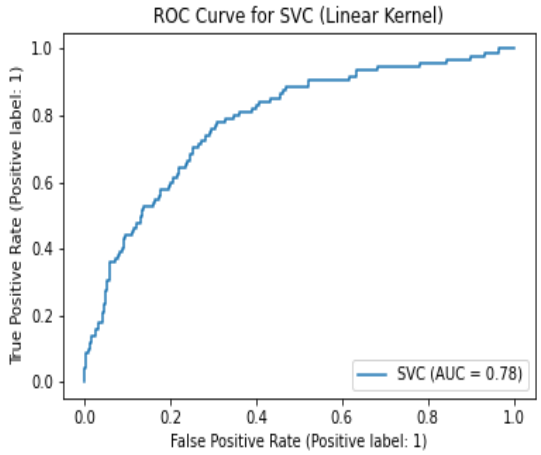
And the ROC curve for all the models are as follows:

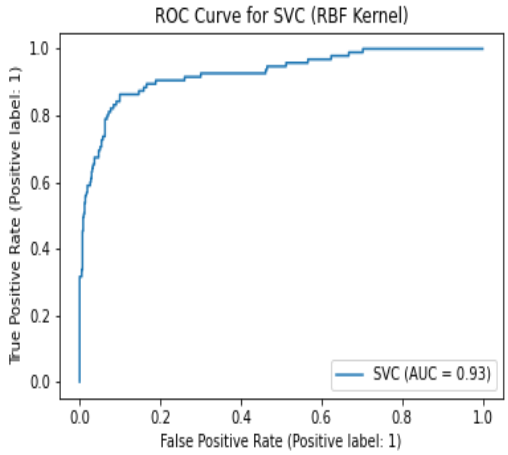


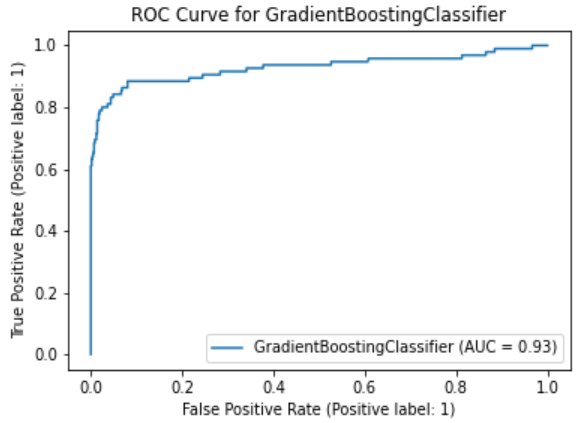












9. Recommendation

Since our objective is to classify and predict the churn and not churn customers, we will evaluate the performance metrics of all classification models. According to the above classification report (precision, recall and f1 score), confusion matrix and area under the ROC curve, I would highly recommend Gradient Boosting Algorithm. I was confused between Decision Tree Classifier, Random Forest Classifier and Gradient Boosting Classifier. In the case of skewed dataset where one class has less samples as compared to other in binary classification, the recall for “churn” class is important because we want our model to predict all the churn samples as churn samples which we can get only when the recall is higher as possible. We would be okay to compromise the precision which means that the predicted churn classes might not be churn classes. It is better to predict not churn class as churn rather than to predict churn class as not churn. Therefore, out of three classifiers, I would recommend Gradient Boosting Algorithm because it has highest recall value as compared to other models and additionally, it also has higher values in other metrics (precision, areas under the curve, f1 score and accuracy) as well.

10. Key Findings and Insights

From the given two datasets: principle training dataset and principle testing dataset, we had to be able to predict the customers who would churn the telecom service. So we had two classes: churn

and not churn. But our major problem in the dataset was its skewness. The class samples were not distributed uniformly. I also plotted the correlation matrix for all the 19 features and found out that 6 pairs of the features seemed to have direct correlation with each other. So I removed one of the each correlated feature pair. After that, to see the variance in the dataset, I applied Principal Component Analysis too which did not give me the clear idea where the decision boundary locates. This clearly gives me the idea that the decision boundary might be non-linear. I first tried to train the classification model with logistic regression which gave us bad performance. So I tried to resample the dataset to make the class distribution even by oversampling using the method ADASYN (Adaptive Synthetic oversampling) thinking that the skewness was the problem. Even after applying the oversampling method, the performance of the model with logistic regression did not improve. So, I came to the conclusion that data distribution is not the problem. It might be the classification model and its decision boundary criteria or its feature's non-linear properties. So I tried 4 different classification models, which are:

- Decision Tree Classifier
- Random Forest Classifier
- SVM with linear kernel
- SVM with rbf kernel
- Gradient Boosting Algorithm

Each classifier has their own specific properties to handle the dataset which could be also studied in the course itself. For the nature of the dataset, Decision Trees, Random Forest, SVM with rbf kernel and Gradient Boosting algorithm seemed good in terms of performance. And out of these, Gradient Boosting algorithm has the best performance metrics.

Therefore, I concluded that Gradient Boosting Classifier will give us the best results to predict if the future customer will churn or not churn the telecom service.

11. Suggestions

Even though we concluded that Gradient Boosting Algorithm seems the best among the above mentioned models, the model does not give us the ideal results. There seems to be problems:

- Under-fitting, which can be improved by using different methods such as adding features, adding polynomial features, try decreasing the regularization parameter etc.
- Run time, the time is not the concern for our dataset because our dataset is small and does not pose problems. But if the dataset is extremely high and run time becomes the concern, we can also look into the run time of each algorithm simultaneously with other performance metrics.

To get even better performance and minimize the problems, we can implement following tweaks in the overall algorithm:

- Can use other resampling methods such as random oversampler, SMOTE, oversampler, undersmpling techniques.
- Can use larger dataset to improve the generalization for the churn class instead of oversampling.
- Can use other classification models such as ensemble methods, KNN classifier, AdaBoost classifier, other bagging classifier, etc.
- Can also use feature elimination method to better the performance by only including the important features and use other feature processing algorithms.
- Can also use different feature scaling methods.
- Can use different SVM kernels to improve the performance of SVM classifier.
- Can use different regularization parameters in models.
- And many more algorithms.

References

[1] Telecom Churn Dataset, https://www.kaggle.com/mnassrib/telecom-churn-datasets.

[2] Wikipedia, https://www.wikipedia.org/