Springboard—Data Science Career Track
Capstone Project 2
Predicting Customer Churn
By Sheema Murugesh Babu
September 2019

1 Introduction

Churn quantifies the number of customers who have unsubscribed or canceled their service contract. Customers turning their back to a service or product are no fun for any business. It is very expensive to win them back once lost. Keeping the right customers can be quite valuable for a company. Not only because customer acquisition is much more expensive, but as more and more business models are shifting towards subscription plans, a customer can be worth thousands of dollars in the future. Reducing churn ultimately leads to a sustainable growing business.

1.1 Problem Statement

The challenge here is to build a model that identifies customers with the intention to leave a service in the near future. The data contains Demographic information like gender, age range, and whether they have partners and dependents, contains customer account information, services that each customer has signed up for and lastly customers who left within the last month – the column is called Churn.
My solution looks into building machine learning models to predict customer churn. Given the dataset, I could estimate whether a customer may or may not be unsubscribed to a service.

2 Approach

2.1 Data Acquisition and Wrangling

About the Dataset

The data was found from the "Telcom Customer Churn" dataset provided by Kaggle's website. https://www.kaggle.com/blastchar/telco-customer-churn and was saved into the local as '**Telco-Customer-Churn.csv**'.

This is the current data they have available:

Variable	Definition
customerID	Customer ID
gender	Sex of User
SeniorCitizen	Senior Citizen
Partner	Partner
Dependents	Dependents
tenure	Tenure
PhoneService	Phone Service
MultipleLines	Multiple Lines
InternetService	Internet Service

OnlineSecurity	Online Security
OnlineBackup	Online Backup
DeviceProtection	Device Protection
TechSupport	Tech Support
StreamingTV	Streaming TV
StreamingMovies	Streaming Movies
Contract	Contract
PaperlessBilling	Paperless Billing
PaymentMethod	Payment Method
MonthlyCharges	Monthly Charges
TotalCharges	Total Charges
Churn	Churn

Steps on Data-Wrangling

First, I imported the required packages that I would need for Data Wrangling. Then, I used pandas read_csv module to import the csv file (i.e. 'Telco-Customer-Churn.csv') which was saved in my local repository. Upon executing the read_csv function using the info() method, the csv file has 7043 entries and 21 columns with different formats of data. Also, it seems that there are no missing values. Digging deeper into it using the describe() method along with the head(). Next step would be to find the distinct values in the dataset.

Second, I used nunique() method to find the count of distinct values in the dataset. All the columns except tenure, MonthlyCharges and TotalCharges have numerical values and all the other columns would be considered as categorical. Note: 'tenure' column is also categorical only its values are numerical.

Third step would be to perform some Data Manipulation like, changing the type of 'TotalCharges' column from object type to integer type. When I tried to do that, it gave me an error. There were actually some blank spaces in that column. I replaced the blank spaces with the mean. Also, changed the data in few other columns like 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'MultipleLines' and 'SeniorCitizen'. Added a column called 'Tenure_group' to group in data from the tenure column. Dropped the irrelevant column, separated the churn, non-churn customers, categorical and numerical columns.

2.2 Storytelling and Inferential Statistics

After I wrangled and cleaned the dataset, I started to explore the data in detail. To put great visualizations, I used 'Plotly' library. Let's see some of the visualizations here.

1. Plot for 'Percentage of churn in the dataset':

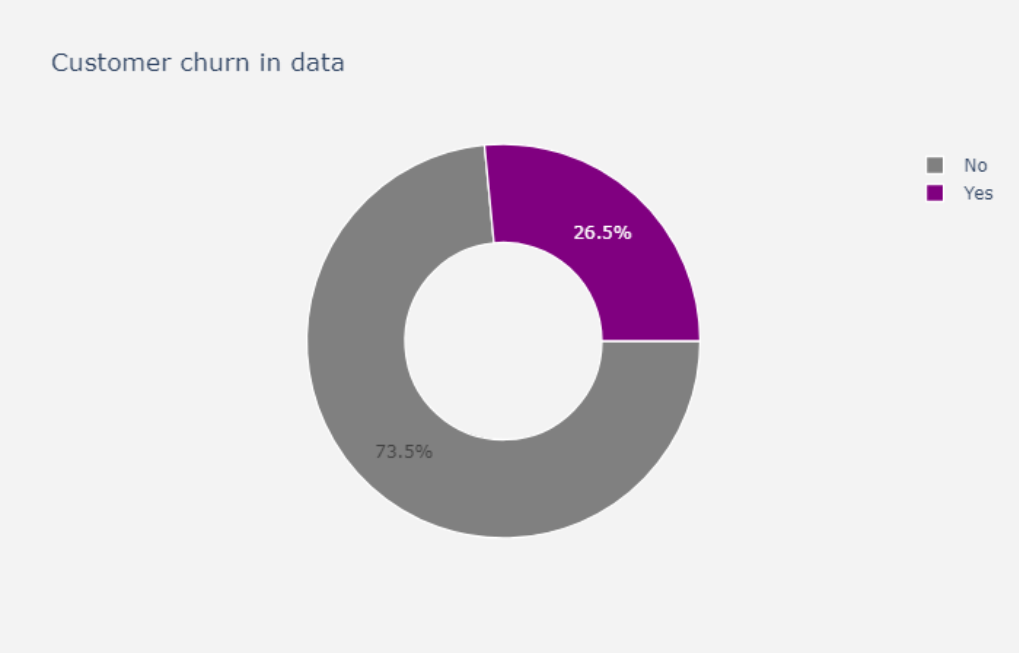


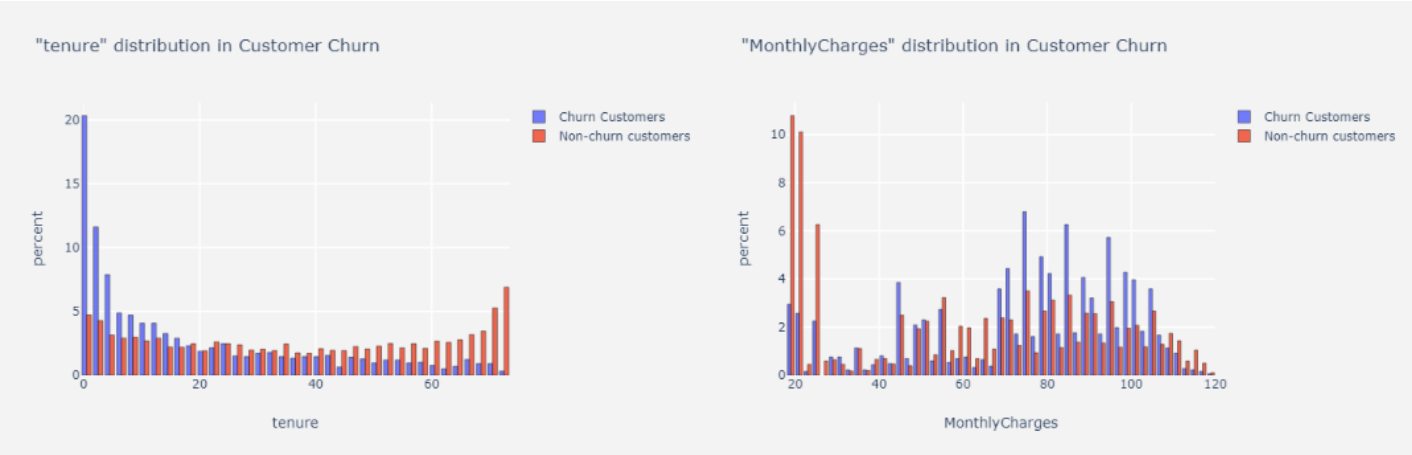
Figure: Customer Churn in data

From the above pie chart, we can see that only 26.6% of the data represents churn customers and the majority are the non-churn customers. This also shows that we might be dealing with a class imbalance problem as there are more non-churned customers than the churned ones.

2. Plots for some of the categorical columns:

Figure: Pie Plots for few columns.

3. Histograms for all the numerical columns:



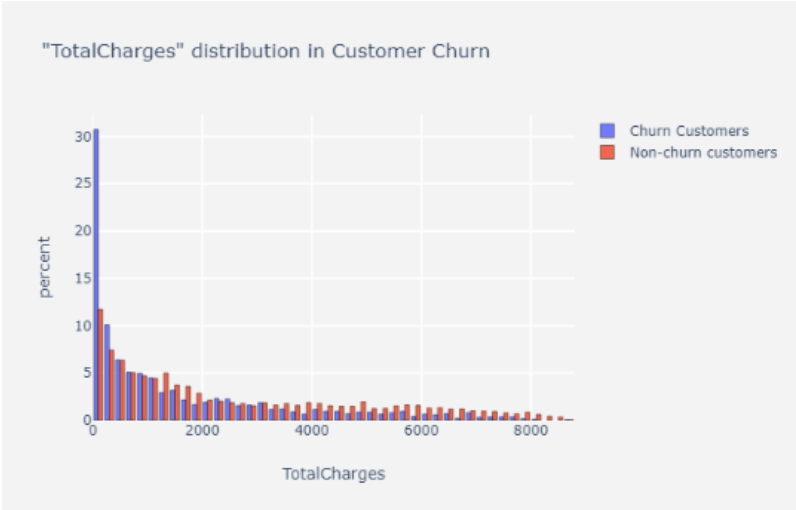


Figure: Histogram plot for the numerical columns.

4. Customer Churn in Tenure groups:

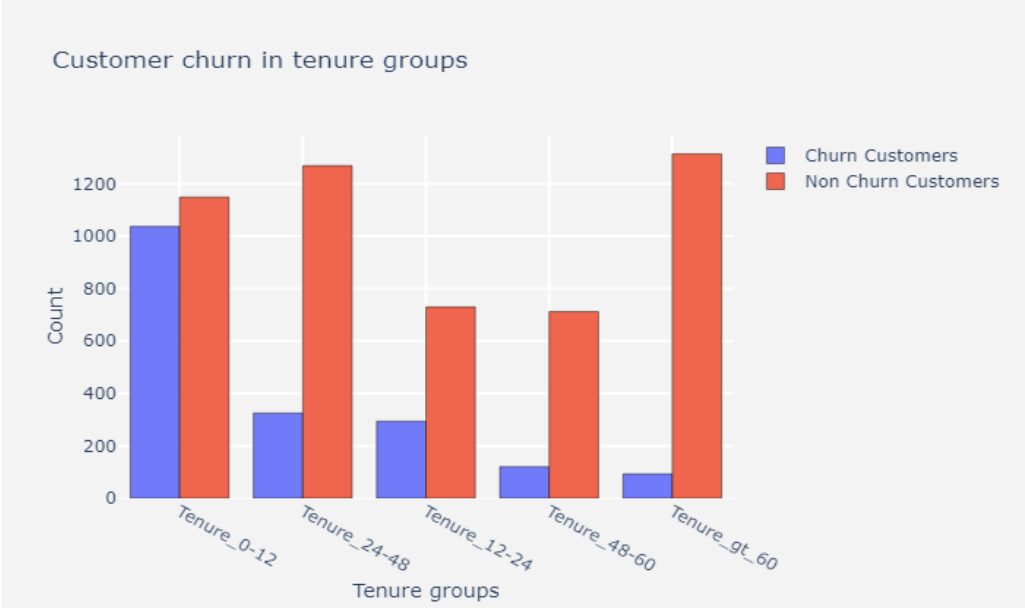


Figure: Customer Churn in Tenure groups

5. Monthly Charges and Total Charges by Tenure group and Churn group:

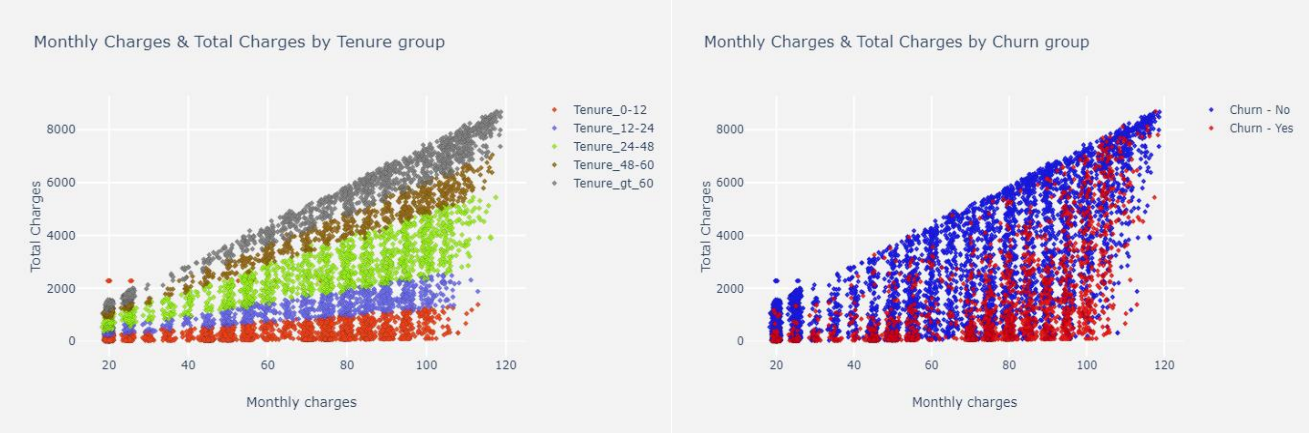


Figure: Monthly Charges and Total Charges by Tenure group and Churn group

6. Average charges by Tenure Groups:

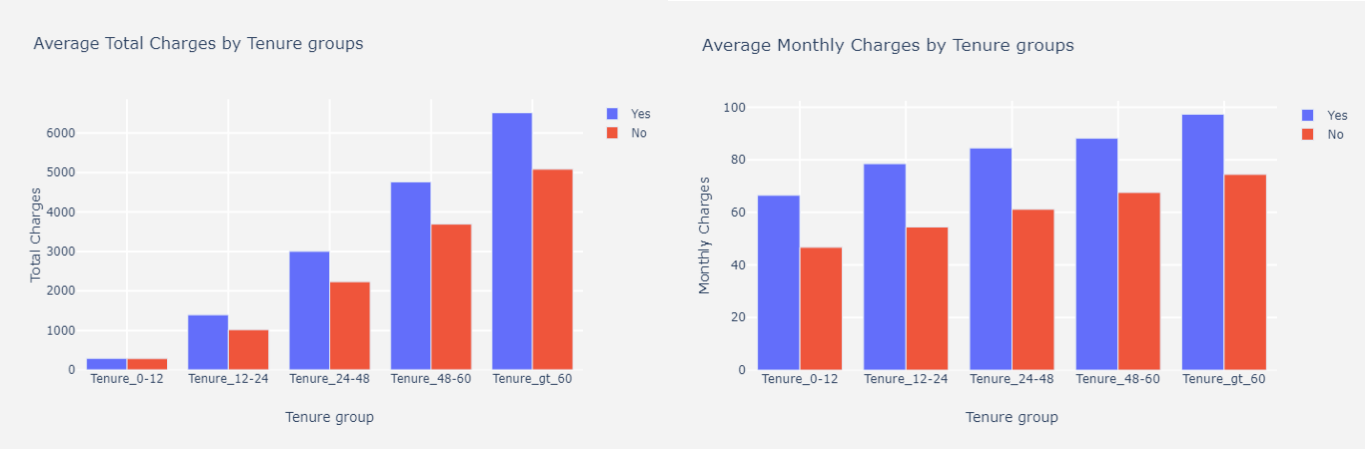


Figure: Average charges by Tenure Groups

The next step would be to provide steps on Inferential Statistics. I have performed Hypothesis testing using the Chi – Square test.

Hypothesis Test:

Chi-Square Test

The Pearson's Chi-Squared test, or just Chi-Squared test is a statistical hypothesis test that assumes (the null hypothesis) that the observed frequencies for a categorical variable match the expected frequencies for the categorical variable. The test calculates a statistic that has a chi-squared distribution, named for the Greek capital letter Chi (χ) pronounced "ki" as in kite.

The Chi-Squared test uses something called a contingency table, by first calculating the expected frequencies for the groups, then determining whether the division of the groups, called the observed frequencies, matches the expected frequencies. The result of the test is a test statistic that has a chi-squared distribution and can be

interpreted to reject or fail to reject the assumption or null hypothesis that the observed and expected frequencies are the same.

Imported the 'chi2' and the 'chi2_contingency' from the scipy.stats library and wrote a function for Contingency table for all the categorical columns and the numerical columns. The function returned a Contingency table for each categorical/numerical column against the target variable i.e. Churn, degrees of freedom, expected values, the test statistics such as Probability, Critical values, Chi-square statistic, significance and the p-value. If the p-value < 0.05, it would mean there is a relationship between the 2 categorical variables.

Categorical Columns		**Numerical Columns**	
Column Name	**Significance**	**Column Name**	**Significance**
SeniorCitizen	YES	tenure	YES
Partner	YES	MonthlyCharges	YES
Dependents	YES	TotalCharges	NO
MultipleLines	YES		
InternetService	YES		
OnlineSecurity	YES		
OnlineBackup	YES		
DeviceProtection	YES		
TechSupport	YES		
StreamingTV	YES		
StreamingMovies	YES		
Contract	YES		
PaperlessBilling	YES		
PaymentMethod	YES		
Tenure_group	YES		
Gender	NO		
PhoneService	NO		

Table shows the significance of all Categorical and Numerical Columns.

2.3 Baseline Modeling

At first, I used the Label encoding for all the binary columns and created dummy variables for them using the get_dummies method of Pandas to create dummy features for all the categorical data. Then, I imported all the necessary packages that I would be using for the machine learning analysis. Then, I divided the dataset into input 'X' and label 'y'. The variable 'X' contains all columns except the target variable and the variable 'y' contains only the target variable. By the help or using input 'X', we will find out, predict, or classify the label 'y'. Then, I divided the X and y into training and testing data set. The training set is 75% of our total data and training set is 25% of the data.

Next step would be to instantiate a Logistic Regression object. Here again I, fitted the model on the training set, predicted on the test set as well as the training set. I calculated different scoring metrics to measure the performance of my model. I stored each score to compare which method was the best. The metrics in

consideration here are the accuracy_score, precision_score, recall_score, confusion_matrix, classification_report, roc_auc_score, roc_curve and f1_score.

A **Classification report** contains precision, recall, f1-score, support with the accuracy, macro avg and the weighted average.

- The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier to not label a sample as positive if it is negative.
- The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.
- The F score can be interpreted as a weighted harmonic mean of precision and recall, where an F score reaches its best value at 1 and worst score at 0. The F score weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.
- The support is the number of occurrences of each class in y_test.

A **Confusion matrix** is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

Ideally recall is the ability of a model to find all the relevant cases per class. True positives are data point classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect). In the churn case, true positives are correctly identified churn customers, and false negatives would be churn customers incorrectly classified. Recall can be thought as of a model's ability to correctly classify data points according to their class.

Before considering Recall as the metric for evaluation, it should be discussed with the business team since choosing Recall can lead to less precision] at the expense of correctly classifying Churn cases.

Classification Report Summary for Logistic Regression (LR):

The class of interest which is the Churn class for the training set has only 67% precision and 53% recall score and the test set has 68% precision and 51% recall score. Our model for the churn classes has not performed great. But the overall global accuracy scores were much higher than the individual scores for the churn class. I then plotted a ROC Curve.

The receiver operating characteristic (ROC) curve is a common tool used with binary classifiers. The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner). Higher the AUC, better is the model.

The ROC curve as shown in the figure below is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

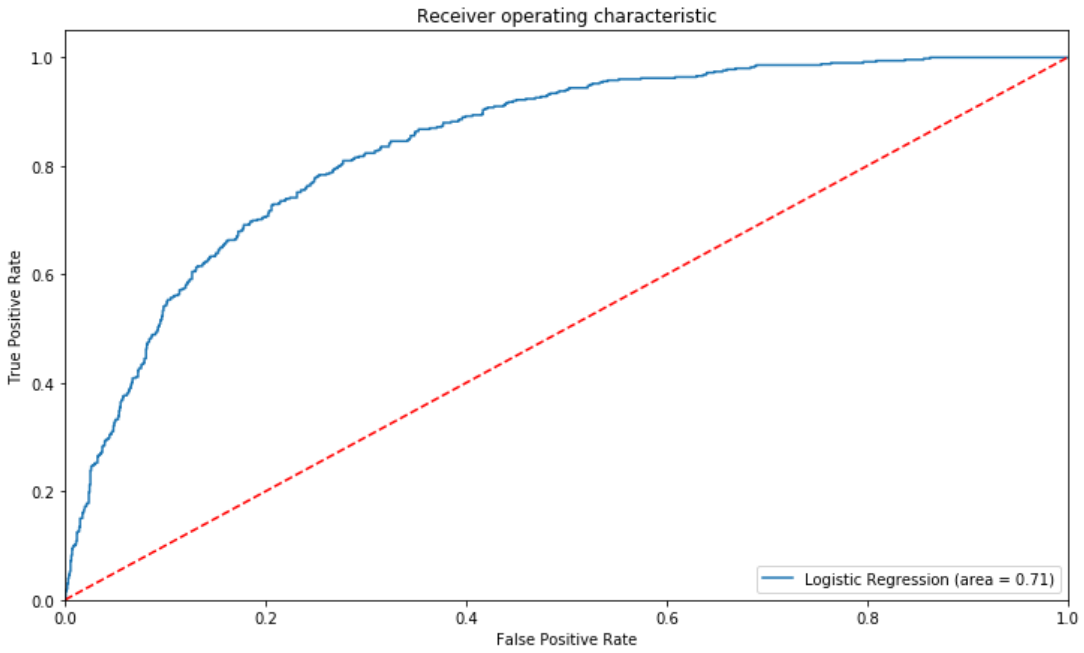


Figure: ROC curve for Logistic Regression

The Area Under Curve for the Logistic Regression model is 0.706359628925045.

2.4 Extended Modelling

This section includes all the other machine learning models. First, I tried applying some sampling methods to see if this gives us better metrics than the baseline model, as a way of addressing the inherent imbalance in the dataset. Here I have used 'Random Undersampling'. I imported all the necessary packages, instantiated a RandomUnderSampler() and fitted it on the train set.

Some numbers to note are,

- Length of undersampled data is: 4110
- Number of non-churn customers in undersampled data: 2740
- Number of churn customers: 1370

After obtaining the undersampled sets, I then used them as my train sets for the base model i.e. Logistic Regression model. I performed the same steps as before, this time using the undersampled sets. Here again, I calculated different scoring metrics to measure the performance of my model. I stored each score to compare which method was the best.

Classification Report Summary for LR with Undersampling:

The class of interest which is the Churn class for the train set now has 69% precision and 64% recall score and that of the test set has 64% precision and 63% recall score.

Looking at the recall scores for both the models, the model with undersampling technique has a better recall score than the later. Once again, I plotted a ROC Curve.

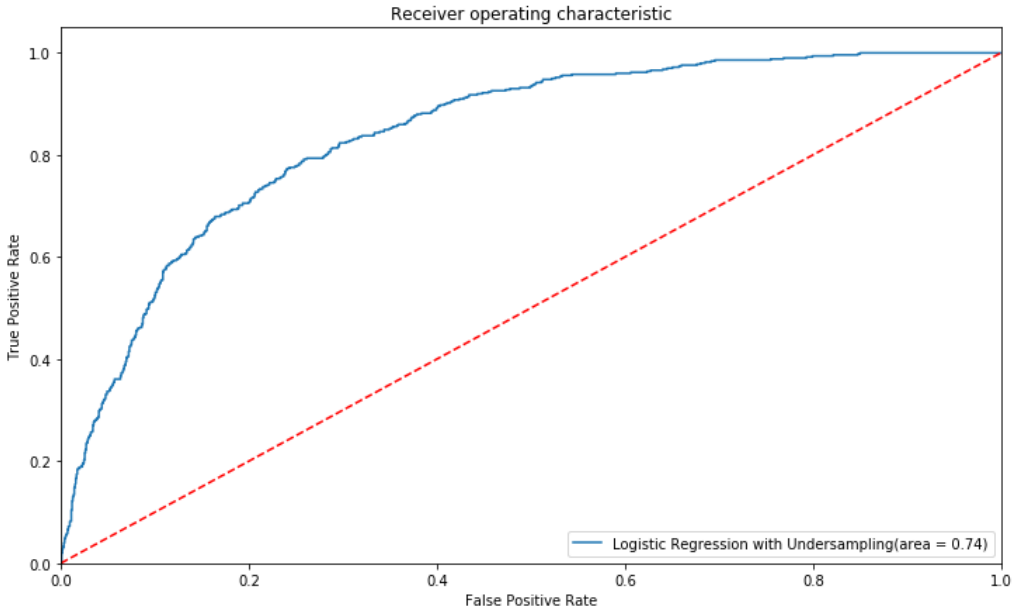


Figure: ROC curve for Logistic Regression with Undersampling

The Area Under Curve for the Logistic Regression with undersampling is 0.7424984676166914.

Just to check that our model was not overfitting, I performed some hyperparameter tuning using GridSearchCV.

I imported the necessary packages. Here, I used my LR with undersampling object. Set up a hyperparameter grid with different values for 'C'. Instantiated GridSearchCV, fitted on the train set and predicted on the test set. Calculated different scoring metrics and stored them.

The Classification Report Summary for LR with Undersampling and GridSearchCV. There wasn't much of a difference in the recall scores for the two. Hence, we could say there is no overfitting. ROC Curve for GridSearchCV is as shown in the figure below.

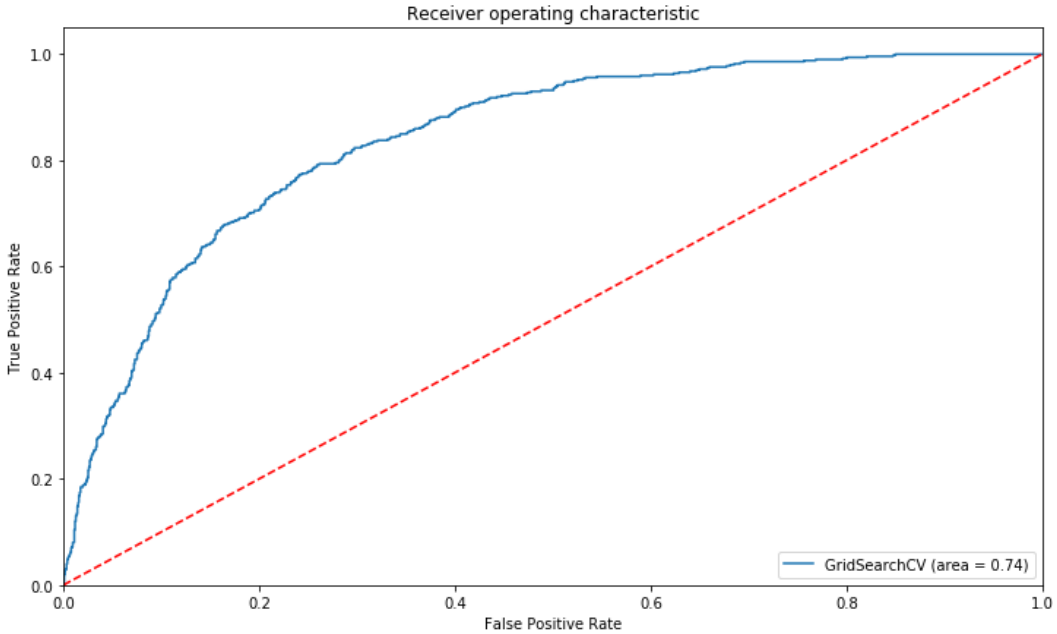


Figure: ROC curve for GridSearchCV

The Area Under Curve for the GridSearchCV model is 0.7414964636086754.

The next type of model I choose was the XGBoost model,
which is one of the implementations of the Gradient Boosting concept, but what makes XGBoost unique is that it uses “a more regularized model formalization to control over-fitting, which gives it better performance,” according to the author of the algorithm, Tianqi Chen. Therefore, it helps to reduce overfitting.

I combined the Synthetic Minority Oversampling TEchnique (SMOTE) method with the XGBoost model.

I imported all the necessary packages for SMOTE, instantiated a SMOTE() object and fitted it on the train set.

Some numbers to note are,

- Length of oversampled data is: 7824
- Number of non-churn customers in oversampled data: 3912
- Number of churn customers: 3912

After obtaining the oversampled sets, I then used them as my train sets for the XGBoost model. I performed the same steps as before, this time using the undersampled sets. Here again, I calculated different scoring metrics to measure the performance of my model. I stored each score to compare which method was the best.

Classification Report Summary for XGBoost Model:

The class of interest which is the Churn class for the train set for XGBoost model has 88% precision and 87% recall score and that of the test set has 64% precision and 58% recall score. ROC Curve for XGBoost Model is as shown in the figure below.

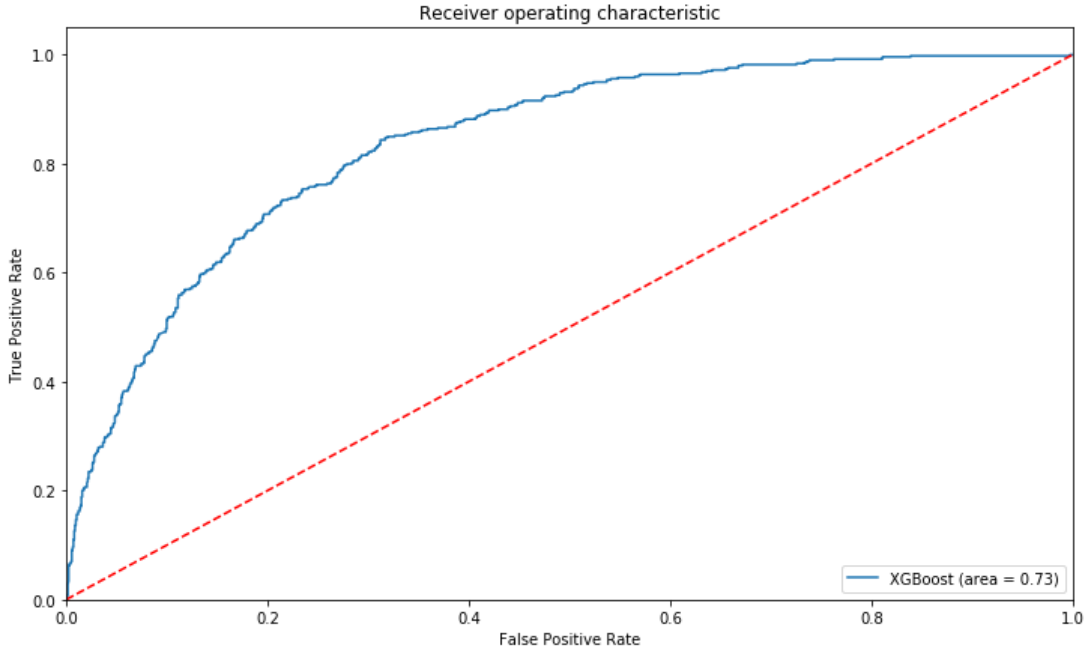


Figure: ROC curve for XGBoost Model.

The Area Under Curve for the XGBoost model is 0.7252087376019867.

Performed some hyperparameter tuning for XGBoost model to improve our metrics i.e. recall scores and in turn

improving performance.

First, I set up a parameter grid of XGBoost, instantiated an XGB classifier and applied GridSearchCV. Fitted it on the train set and predicted it on the train set as well as the test set. Here again, I calculated different scoring metrics to measure the performance of my model. I stored each score to compare which method was the best.

Classification Report Summary for XGBoost with GridSearchCV:

The class of interest which is the Churn class for the train set for XGBoost with GridSearchCV has 92% precision and 88% recall score and that of the test set has 65% precision and 55% recall score. ROC Curve for XGBoost with GridSearchCV is as shown in the figure below.

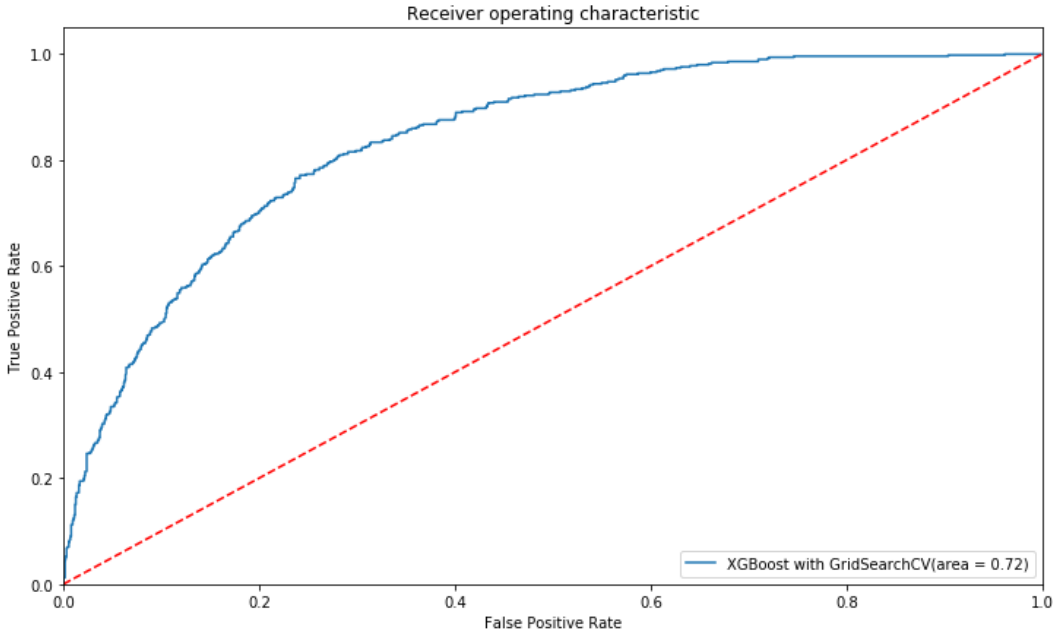


Figure: ROC Curve for XGBoost with GridSearchCV

The Area Under Curve for the XGBoost model with GridSearch is 0.7161216251838066.

3 Findings

So far, we saw how to predict customer churn using different machine learning algorithms. It's time to summarize our findings. We calculated different scoring metrics to measure the performance of the model.

I used the Classification report, confusion matrix and ROC curve as my metrics. In the classification report, I mainly concentrated on the Recall scores since Recall identifies the relevant cases in a dataset.

Firstly, I put up a Logistic regression model as my base model, performed all the necessary steps. The class of interest which is the Churn class for the test set has 68% precision and 51% recall score.

Secondly, I performed Logistic regression with undersampling with all the necessary steps. The class of interest which is the Churn class for the test set has 64% precision and 63% recall score. This model performed better than the previous one.

Thirdly, I performed Logistic regression with GridSearchCV with all the necessary steps. The class of interest which is the Churn class for the test set had 64% precision and 62% recall score. There isn't much of a difference

in the recall scores for the tuned model.

Fourthly, I performed the XGBoost with SMOTE. The class of interest which is the Churn class for the test set has 64% precision and 58% recall score.

At last, I performed the XGBoost model with hyperparameter tuning to improve our metrics. The Churn class for the test set has 65% precision and 55% recall score.

Algorithms	Precision	Recall
Logistic Regression (LR)	68%	51%
LR with Undersampling	**64%**	**63%**
LR with GridSearchCV	64%	62%
XGBoost with SMOTE	64%	58%
XGBoost with GridSearchCV	65%	55%

4 Conclusions and Future Work

In the last section, we jotted down the metrics for all the machine learning models. We can conclude that Logistic Regression with undersampling performed better for the recall score which is our metric of interest.

Going forward I would like to improve the performance of the XGBoost model by using more parameters and also by trying out different algorithms.

5 Recommendation for the Client

In this section, I performed some feature analysis, to provide an idea for the clients on which variables affect the churn.

This is in continuation with the XGBoost model, I used the plot_importance function of the xgboost. The result of the function gave me features (as numbers like f1, f2, f3 and so on) with the F - score. Higher the F - score more importance for the features. The figure below shows the feature importance plot for the XGBoost Model.

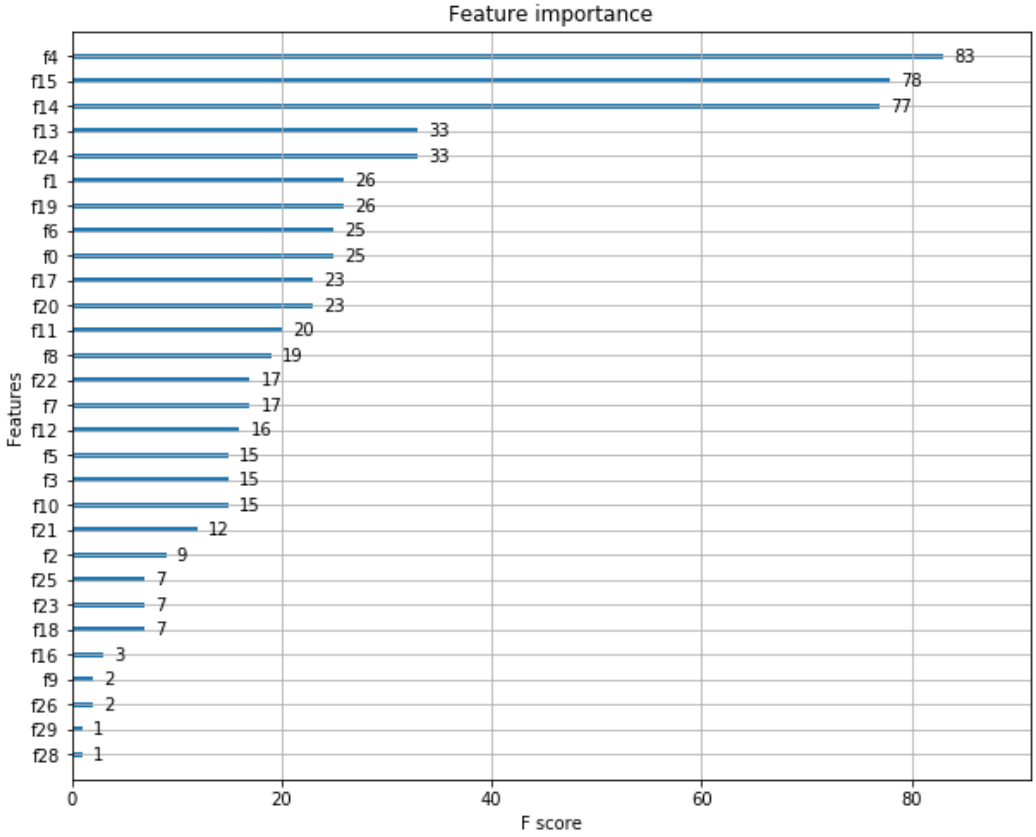


Figure: Feature Important Plot for XGBoost Model

From the above plot we can see that the top 5 features are f4, f15, f14, f13, f24 and so on. Future work involves finding out the exact feature names for these numbers.