**Predicting Churn of Wireless Customers**

By

Ravi Kallepalli

A Capstone Project Paper Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

In

Data Science

University of Wisconsin – Stevens Point

Stevens Point, Wisconsin

May 2020

**DEDICATION**

To my wife Poornima and my daughter Naina, for constant support and encouragement during this program.

# **ABSTRACT**

Predicting Churn of Wireless Customers

Ravi Kallepalli

In telecom industry churn represents percentage of subscribers leaving the service provider in a given period of time. In telecom industry costs for acquiring a new customer is way more expensive than retaining the existing customers. So all service providers try to reduce churn and its one of the most important KPI’s (key performance indicator) companies and stock holders emphasize on. This is also one of the key metrics released every quarter during financial reporting to investors. By predicting which customer might leave beforehand, company can take appropriate actions to avoid customer leaving the company. This can save companies millions of dollars from new customer acquisition costs and increased revenue and profits by keeping their existing customers.

Using the telecom churn dataset from Kaggle, this projects builds a machine learning model to predict churn. There are about 80 different features in the data set for each customer and an output field churn which is a binary function which tells whether the customer churned or not. Using this data the project builds a classification model to predict whether the customer will churn or not which can be used on new data to

Keywords: churn, machine learning, classification, prediction, wireless.

# **TABLE OF CONTENTS**

[ABSTRACT iv](#_Toc39819209)

[TABLE OF CONTENTS v](#_Toc39819210)

[LIST OF FIGURES v](#_Toc39819211)

[1. INTRODUCTION: 1](#_Toc39819212)

[2. OBJECTIVE 3](#_Toc39819213)

[3. PROJECT OVERVIEW 4](#_Toc39819214)

[4. MACHINE LEARNING LIFE CYCLE 6](#_Toc39819215)

[**4.1 BUSINESS PROBLEM CASE** 6](#_Toc39819216)

[**4.2 MACHINE LEARNING PROBLEM FRAMING** 7](#_Toc39819217)

[**4.3 DATA COLLECTION** 8](#_Toc39819218)

[**4.4 DATA VISULIZATION AND ANALYSIS** 9](#_Toc39819219)

[**4.5 DATA PREPARATION** 13](#_Toc39819220)

[**1. Handling Missing Values:** 13](#_Toc39819221)

[**2. Handling Outliers:** 18](#_Toc39819222)

[**3. Data Transformation:** 19](#_Toc39819223)

[**4. Splitting Data into Training and Test sets:** 26](#_Toc39819224)

[**4. 6 FEATURE ENGINEERING / FEATURE SELECTION** 27](#_Toc39819225)

[**4.7 MODEL TRAINING** 35](#_Toc39819226)

[**4.7.1 Initial Model Evaluation:** 37](#_Toc39819227)

[**4.7.2 Model Tuning:** 45](#_Toc39819228)

[**4.8 MODEL EVALUATION** 49](#_Toc39819229)

[5. CONCLUSION 52](#_Toc39819230)

[6. REFERENCES 52](#_Toc39819231)

[APPENDIX A 54](#_Toc39819232)

# 

# **LIST OF FIGURES**

[Figure 1.1 : US Wireless Providers Churn Rate by Quarter 2](#_Toc37153252)

[Figure 2.1: Machine Learning Process 1 4](file:///C:\Users\Ravi\Desktop\UW-MS\DS785-Capstone\Project%20Report.docx#_Toc38453326)

[Figure 2.1: Machine Learning Process 2 5](file:///C:\Users\Ravi\Desktop\UW-MS\DS785-Capstone\Project%20Report.docx#_Toc38453327)

[Figure 4.4 1 Data Set Shape 9](#_Toc38620847)

[Figure 4.4 2 Feature Types 9](#_Toc38620848)

[Figure 4.4 3 Categorical Features 9](#_Toc38620849)

[Figure 4.4 4 Continuous Features 9](#_Toc38620850)

[Figure 4.4 5 Count of Missing Values in each Feature 10](#_Toc38620851)

[Figure 4.4 6 Data Distribution 12](#_Toc38620852)

[Figure 4.5 1 Deleting Columns 14](#_Toc38620862)

[Figure 4.5 2 Deleting Rows with Missing Values 1 14](#_Toc38620863)

[Figure 4.5 3 Continuous Variables Missing Value Count 15](#_Toc38620864)

[Figure 4.5 4 Deleting Rows with Missing values 2 15](#_Toc38620865)

[Figure 4.5 5 Categorical Variables Imputing missing values 16](#_Toc38620866)

[Figure 4.5 6 Continuous variables Missing Values 17](#_Toc38620867)

[Figure 4.5 7 Continuous Variables missing value imputation using Iterative imputer. 17](#_Toc38620868)

[Figure 4.5 8 Handling outliers using Z score 18](#_Toc38620869)

[Figure 4.5 9 Removing rows with Negative values 19](#_Toc38620870)

[Figure 4.5 10 Transforming Negative Values 20](#_Toc38620871)

[Figure 4.5 11 Log Transformation 20](#_Toc38620872)

[Figure 4.5 12 Data Distribution post log transformation 22](#_Toc38620873)

[Figure 4.5 13 Normalizing Numerical Features 23](#_Toc38620874)

[Figure 4.5 14 One-Hot Encoding 24](#_Toc38620875)

[Figure 4.5 15 One-Hot Encoding Output 24](#_Toc38620876)

[Figure 4.5 16 Splitting Data into training and test sets 25](#_Toc38620877)

[Figure 4.6 1 Feature Creation 27](#_Toc39130641)

[Figure 4.6 2 Feature Selection - Removing Constant Features 30](#_Toc39130642)

[Figure 4.6 3 Correlation Matrix 31](#_Toc39130643)

[Figure 4.6 4 Removing Correlated features 31](#_Toc39130644)

[Figure 4.6 5 Feature Selection- Wrapper Method 34](#_Toc39130645)

[Figure 4.7 1 Naive Predicator 36](#_Toc39130660)

[Figure 4.7 2 Training and prediction pipeline 43](#_Toc39130661)

[Figure 4.7 3 Execute the models 44](#_Toc39130662)

[Figure 4.7 4 Models Performance Metrics 44](#_Toc39130663)

[Figure 4.7 5 Hyper-parameter tuning Grid vs Random search 46](#_Toc39130664)

[Figure 4.7 6 AdaBoost Tuning 47](#_Toc39130665)

[Figure 4.7 7 XGBoost Tuning 48](#_Toc39130666)

[Figure 4.7 8 Final Model 48](#_Toc39130667)

[Figure 4.8 1 Final results 51](#_Toc39130674)

# **1. INTRODUCTION:**

Churn in telecom industry is the percentage of subscribers moving from existing wireless service provider to another wireless service provider. In telecom industry cost of customer acquisition is much higher than retaining existing customers. So companies work very hard and provide incentives to keep their existing customers from switching to other carriers. For example T-Mobile is providing Netflix subscription for free of cost to all its customers. Sprint is providing Hulu and Tidal subscription to all its customers. Companies want to keep their customers happy and engaged so they don’t move to a different service provider.

In the United States mobile penetration has saturated meaning most of the population already have a wireless phone. The only way wireless providers can get new customers is either from new devices like watch, tablets etc. or poach customers from other wireless providers, this makes keeping churn lower very hard. So carriers provide incentives to keep existing customer and also provide more or better incentives for new customers so customers from other wireless provides can move and become their customers.

To avoid bad customer experience, wireless providers have to constantly improve customer experience by improving their network quality and investing more money in the next generation technologies and also investing a lot of money in customer experience centers. Wireless companies have to constantly keep improving or else they will experience churn. For example Sprint bought Nextel in mid-2000, post-merger they were not able to keep customers happy as they were not able to seamlessly integrate both the networks. Since then the brand image has fallen down and even today Sprint has to offer more incentives to attract customers from other providers. T-Mobile on the other hand in early 2013 under the leadership of New CEO they tries to improve customer satisfaction by understanding what customers do not like about the exiting services offered by wireless companies. Based on this study they started un-carrier movement which eliminated contracts, started unlimited voice and data and started to gain more customers. Since then T-Mobile was able to add more than 5million customers every year. This gives an idea how companies innovate to keep the churn low and how to gain new customers.

The below chart shows churn by Quarter for all the wireless providers in United States. (Statista 2020 [15])

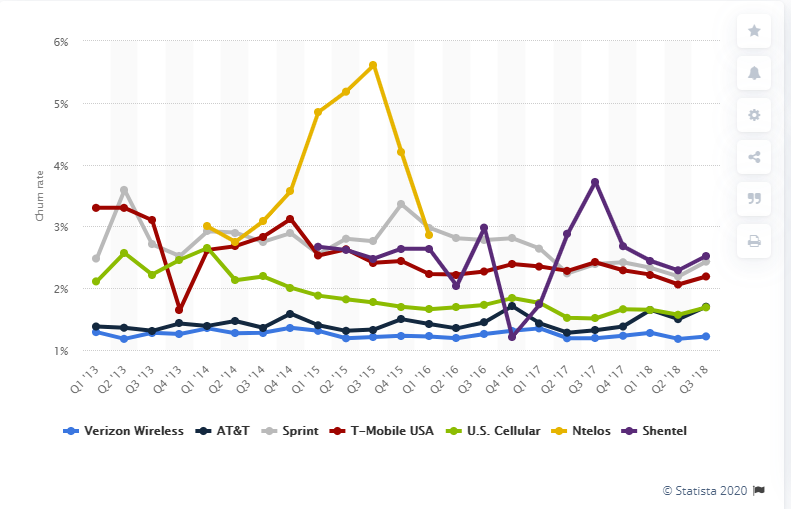


Figure 0‑1.1: US Wireless Providers Churn Rate by Quarter

# **2. OBJECTIVE**

The objective of this project is to predict if a customer will be churning or not based on the different features. If model can predict with some accuracy, the company can take precautionary measures and improve the customer satisfaction before the customer churns. This problem is a high recall problem meaning we are OK with some degree false positives meaning if the model predicts customer might churn but in reality the customer is not churning but NOT OK with false negatives, meaning when customer is churning the model predicts customer is not churning. This would help companies act and resolve any customer issues before customer churns, hence increasing revenues and profits.

The objective/goals of this project could be summarized below with a demonstrated case study

1. Find telecom Churn data from a public domain. Data from a wireless provider cannot be available due to federal rules on customer data.

2. Understand the data, clean and preprocess the data.

3. Prepare data for modelling.

4. Try different modelling techniques for this problem statement.

5. Select the best model and tune the model.

6. Test the model on test data.

7. Summarize the results.

# **3. PROJECT OVERVIEW**

The general approach for this project could be summarized as below, this is common to most of machine learning projects.

Figure 2.1: Machine Learning Process 1

Figure 2.1: Machine Learning Process 2

# **4.** **MACHINE LEARNING LIFE CYCLE**

## **4.1 BUSINESS PROBLEM CASE**

For any given project the first step is to identify the problem statement clearly before proceeding into the project. Business team is the one responsible for defining the key business processes and identifying KPI’s (Key Performance indicators) which are used to measure the business processes. Business users are the one who can communicate the questions they are trying to find the answer for and what decisions they are trying to make. Business teams are the one trying to leverage existing data to find answers to their questions. So the first step in any project is to site with business users understand the problem statement, understand what data they have with respect to problem statement and how that data can be used to answer the questions.

In our case this is a case study project, so there are no business users per say we identified the problem statement and metrics used to evaluate the problem. The problem statement here is to predict telecom customers who will be potentially moving to another service provider, based on the data available for those customers. Identifying potential churn customers for telecom companies is very important. Companies can reach to these customers, identify what their problem is and resolve the issue before the customer’s moves to a different provider, this would help the company keep the customer which in turn increases revenue and profit. In our case our evaluation metric will be F-score and accuracy. Our model should be a high recall model meaning we are OK with False Positives but Not OK with False Negatives. We do not want our model to predict a customer who will be churning as not churning but we are ok to predict few instances where customer will be not churning as churn.

## **4.2 MACHINE LEARNING PROBLEM FRAMING**

This is the step where we convert the business problem into a machine learning problems which can be solved with an algorithm.

**Problem to Solve:**

Create a way to classify whether the existing customer will churn or not based on existing data.

**Business Metric:**

* Reduction in Churn rate.
* Increased Customer count, increased revenue and profits due to reduction in churn.

**ML Problem:**

**Supervised Learning:** We take a set of input features and predict an output. In our case the input features will be specific customer data and output is whether the customer churned or not.

**Classification:** Classification is asupervised learning method where a model learns from the existing data and uses knowledge from that data to classify on new data. In our case we are using customer data to classify whether that particular customer will churn or not.

**Binary Classification:** In binary classification the objective is to assign the data into one of the two buckets. In our case its 1-Churn & 0-Not Churn

## **4.3 DATA COLLECTION**

Data collection not only happens at the start of Machine learning process but can happen as we move along the process. Sometimes during the process you can realize that more data is required or some additional features are required during creating stage of process.

There will scenarios where towards the end of machine learning process when you evaluate results and business goals, if the goals are not met then you might have to go back and re-evaluate your data and check if more data is required.

You also need to collect more data after the model is deployed in production. After model is deployed in production and as the new data comes in your model has to be accountable for new data also, sometimes fine tuning the model is required as you get new data, sometime you might have to remove the model from production if the model is not performing well on new data.

In short data collection happens throughout the machine learning process not just in the beginning of the process. Depending on the problem data can come from various sources from logs, databases, websites, surveys, data providers and many other sources.

For this case study project we are collecting data from public domain. Since the customer data at telecom companies are not shared due to legal and privacy concerns, data collected for this project is based from data available on public domain.

Below is the source of the data, this data is in tabular format. No additional data will be collected for this case study project. Modelling will be solely based on the below data.

<https://www.kaggle.com/abhinav89/telecom-customer>

## **4.4 DATA VISULIZATION AND ANALYSIS**

This is the step where you do exploratory data analysis, understanding the data, find what’s missing in the data and where data cleaning is required. In this section we will visualize and analyze the telecom churn data set.

**Shape of the dataset**: This data set has 90 columns and 100,000 rows



Figure 4.4 1 Data Set Shape

**Feature Types:** This data set has 68 features which are float, 10 features which are integers and 12 features which are categorical.



Figure 4.4 2 Feature Types

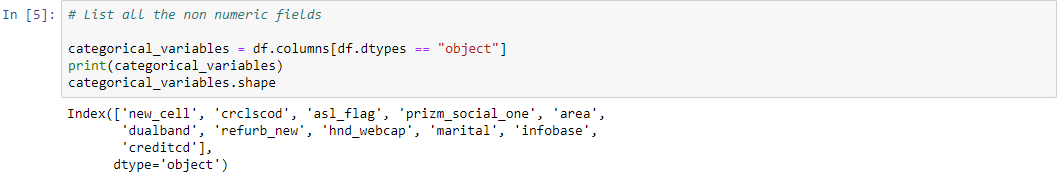
**Categorical Features:** Below are list of features which are Categorical.

Figure 4.4 3 Categorical Features

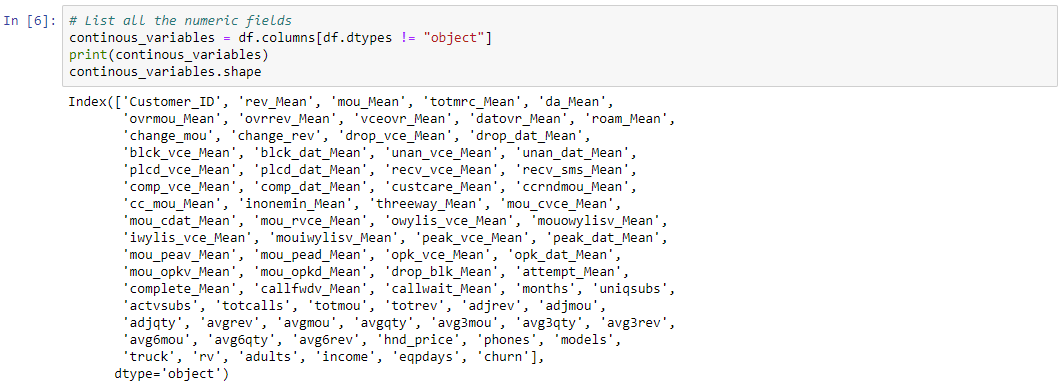
**Continuous features:** Below are the list of feature which contain continuous data.

Figure 4.4 4 Continuous Features

**Features with Missing Values**: There are about 33 features with various degrees of missing values. All these missing values have to be taken care of.

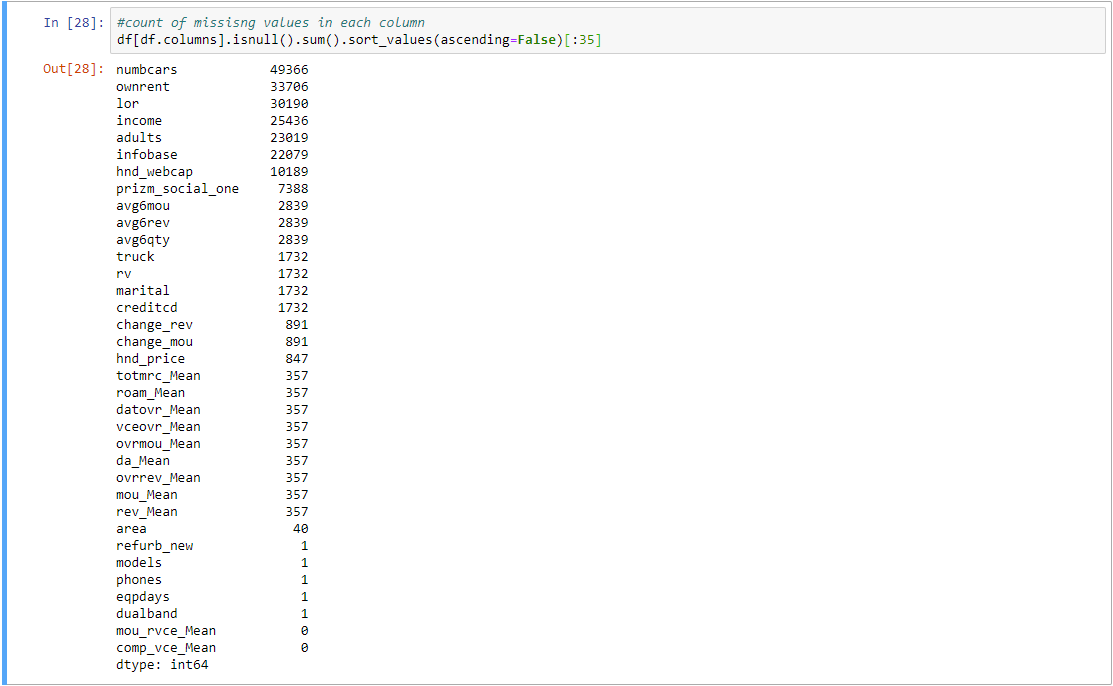
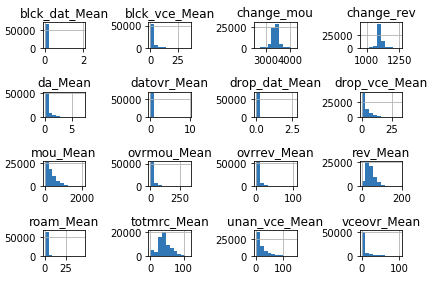
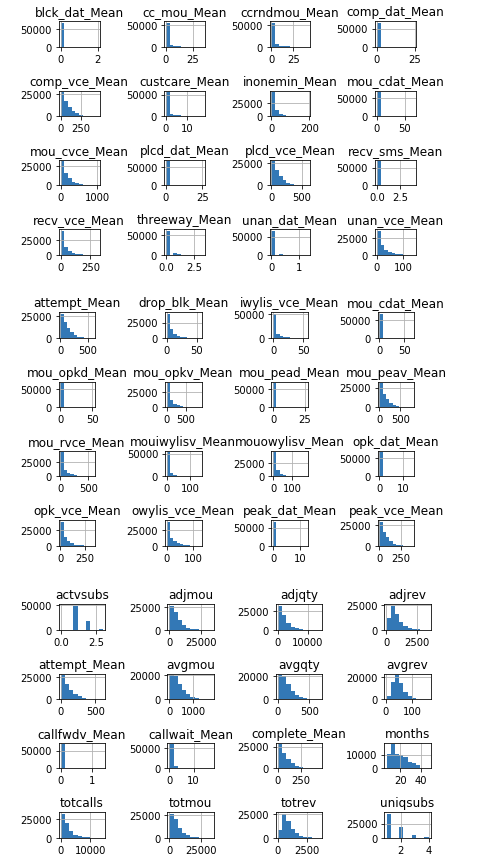


Figure 4.4 5 Count of Missing Values in each Feature

**Distribution of Data for each Feature:** Looking at histogram for all the numerical variables, will give an idea of data distribution and if any outliers are existing. Based on the below data we can see that multiple features have outliers.





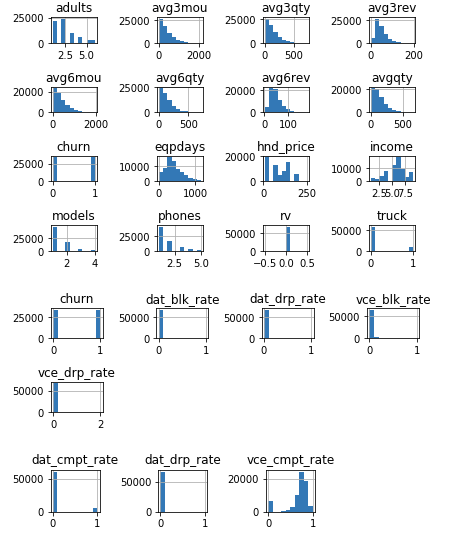


Figure 4.4 6 Data Distribution

Based on the data analysis, the data set has missing values, outliers and distribution which are not normal and there might be features which are highly correlated. Data cleaning and data transformation is required on the dataset before we forward with any modelling. Next step is Data preparation where we clean the data and transform the data and get the dataset ready for modelling.

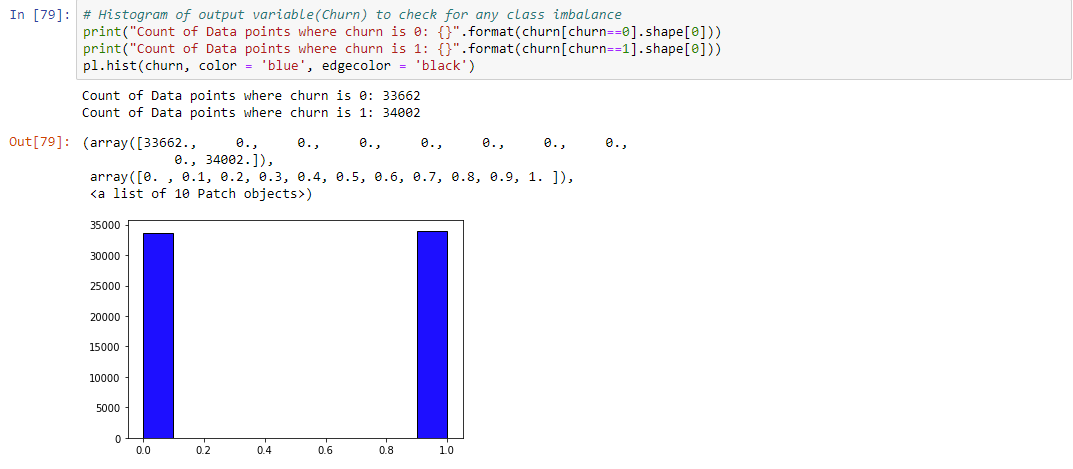


Figure 4.4 7 Target Variable histogram

## **4.5 DATA PREPARATION**

In this stage you prepare the data so that data is ready for modelling.

For this project data preparation can be divided into 4 sections:

1. Handling missing values

2. Handling Outliers

3. Data Transformation

4. Splitting data into training and testing datasets

### **1. Handling Missing Values:**

Missing values can be handled in multiple ways depending on the data. One way is to go back to data source and see if any corrections are required and correct missing data if available. But for this project since the data source is fixed and no changes can be made we will stick with just handling missing values based on the below techniques.

a. Deleting columns with missing values

b. Deleting rows with missing values

c. Imputing data into missing values

**a**. **Deleting columns with missing values:**

If a certain feature has high number of missing values then that feature might not be helpful in the analysis. Deleting all those rows or imputing values might not be helpful. So in these cases we can delete these features from the dataset. For this dataset if any columns has more than 30% of the data missing we will delete those columns from our analysis.

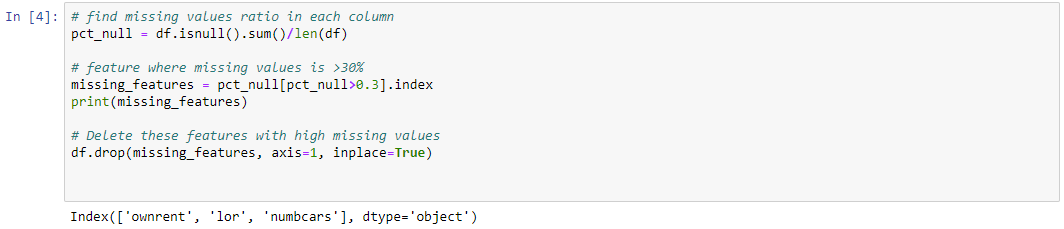


Figure 4.5 1 Deleting Columns

**b. Deleting rows with missing values:**

Deleting all rows when one of the feature is missing a value might lead to loss of valuable information. Deleting rows should be done very selectively to avoid loss of information.

In the churn data set there are 357 rows which are missing data from Col B to Col AW. Since majority of the features are missing data in these specific rows we will delete those rows from data set.

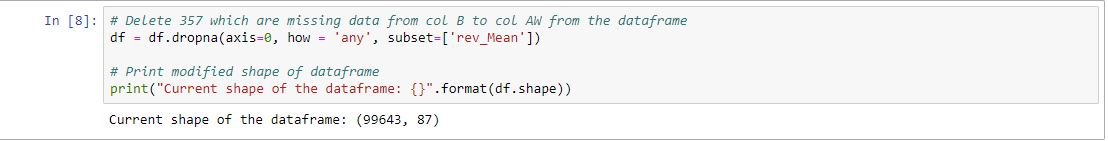


Figure 4.5 2 Deleting Rows with Missing Values 1

After deleting these rows, let’s check how many more rows missing values are present in the continuous variables.

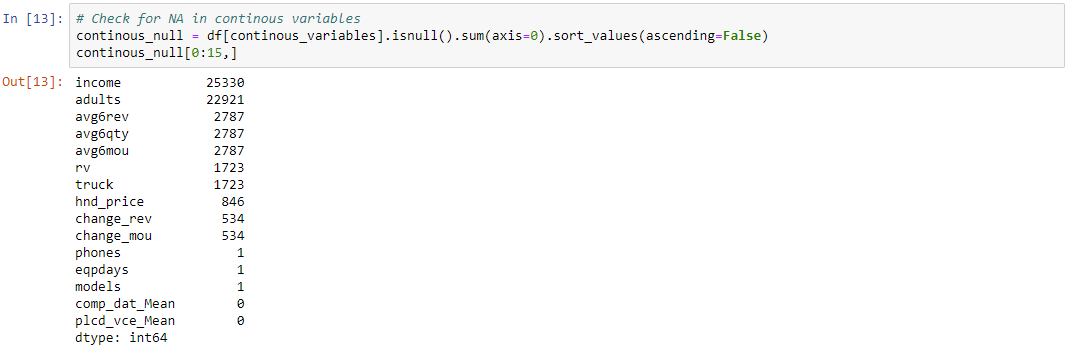


Figure 4.5 3 Continuous Variables Missing Value Count

Based on Figure 4.5 3 income column and adult column have >20K fields with missing values, we cannot delete these rows as deleting these rows will cause information loss. The other columns where we have missing data we will delete those rows.

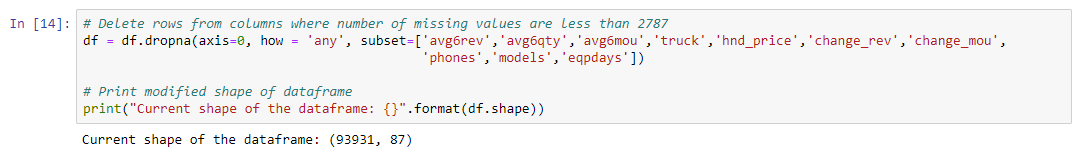


Figure 4.5 4 Deleting Rows with Missing values 2

**b. Imputing Data into missing values:**

Imputation can be done in different ways for continuous and categorical variables.

For categorical variables we can handle missing values by

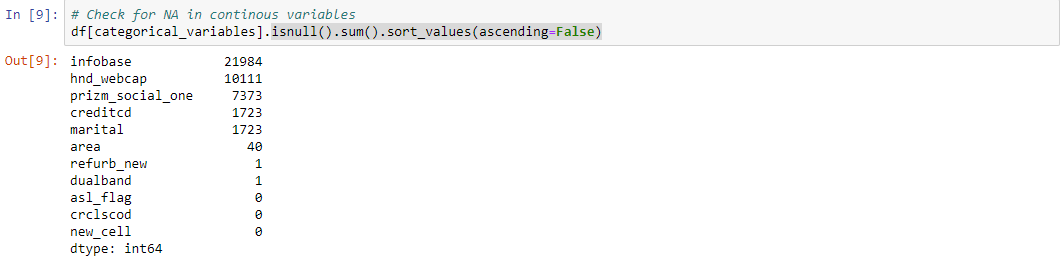
1. Ignore/delete observations with missing values when the dataset is large and there are few records of missing values.

2. Replace with a statistic like mean, median or mode

3. Develop model to predict missing variables

4. Treat missing data as just another category.

For this dataset we have variables with high number of missing values, so for this analysis we will treat missing data just another category. So we will replace all the missing values with MD.



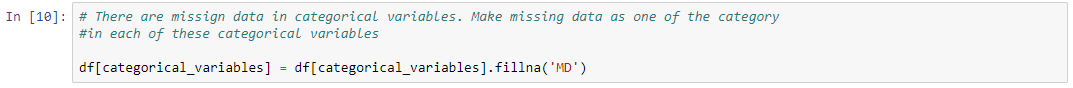


Figure 4.5 5 Categorical Variables Imputing missing values

For continuous variables we can handle missing values by

1. Ignore/delete observations with missing values when the dataset is large and there are few records of missing values.

2. Replace with a statistic like mean, median or mode

3. Develop model to predict missing variables

For this dataset we use iterative imputer from Sklearn library. Iterative imputer takes features with missing values, develops a model as function of other fetaures, estimate the missing values and imputes the missing the value. In does in a iterative manner, where it takes one feature with missing values, treats that as output features and considers all the remaining features as input variables. The features with missing values is output y and rest of the features are input variable X. Then a model is fit to predict y using allt he variables X. This happens in a iterative manner for each variable with missing values.[17].

The only features missing have values are income and adults features. Based on the exiting data both these feature have values between 1 and 9 and all the values are integers. So in the iterative imputer we can input these values so the predicted values will be between 1 and 9 and convert them to integers.

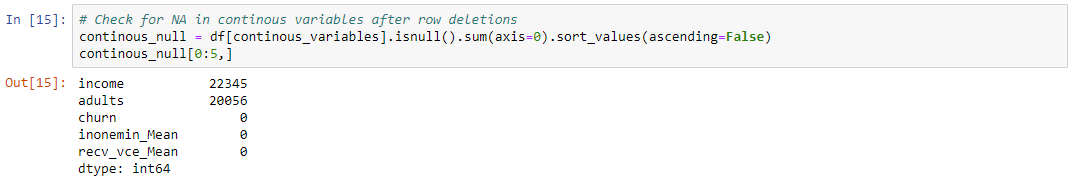


Figure 4.5 6 Continuous variables Missing Values

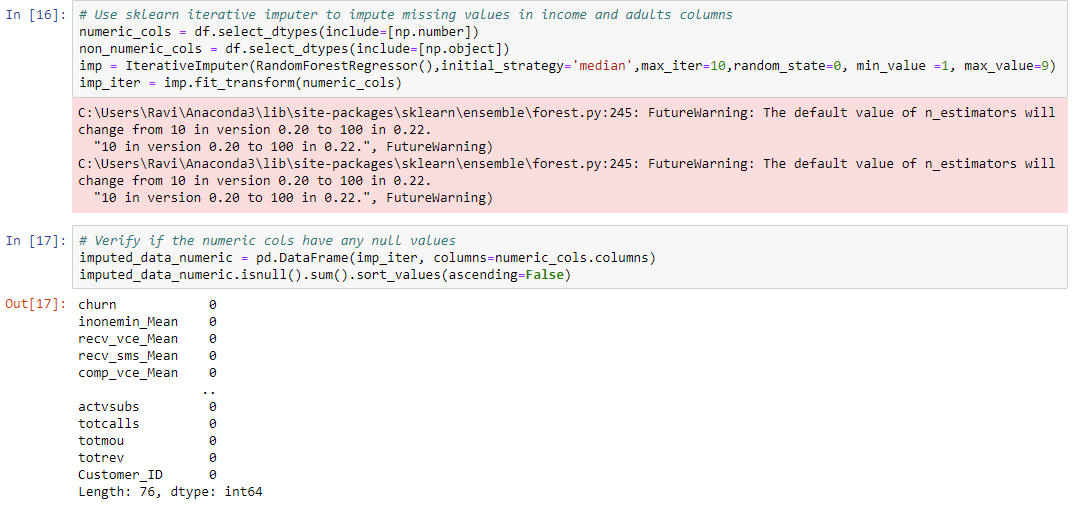


Figure 4.5 7 Continuous Variables missing value imputation using Iterative imputer.

After this step all feature with missing data points are addressed in both continuous and categorical variables. There should be no missing values. The new dataframe has 93931 rows, meaning we have deleted close to 7K rows out of 100K rows on the data set. We will use this new data frame for the next steps of data preparation. The next step in the data preparation process is handling outliers.

### **2. Handling Outliers:**

The performance of a machine learning problem completely depends on data quality. Outliers are observations which are significantly different from other observations. When outliers are present the models assumes those points and can severely underperform if outliers are not cleaned. There are several way to identify and clean outliers. You can check for outliers using histogram, scatter plot and box plot. From **Figure 4.6 6** we have histograms for each feature and we can see outliers in those features. We need to clean those outliers.

Outliers can be dealt with in different ways. We can use IQR (Inter quartile range) identify outliers cap them, we can identify outliers based on Z score and remove them or we can replace outliers with the median value. For this data set we will identify outlier using Z-score and remove them. If Z-score is greater than +/- 3 then we consider those points an outlier and we remove them.

To do this Z-score is calculated for each data point in a column, using column mean and standard deviation. Then absolute Z score is taken for each of these datapoints since direction doesn’t matter, if the z score is below absolute (3) in all columns then those points are kept and ignore any points above that. Post this we have about 68K points for our modelling.

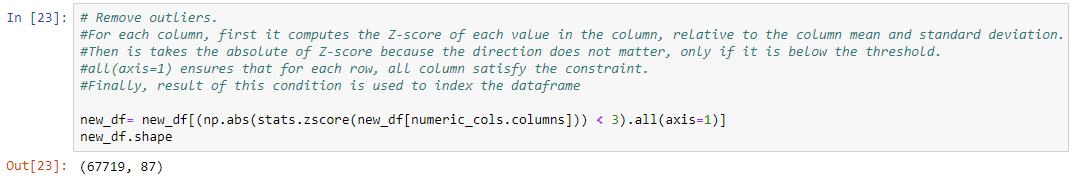


Figure 4.5 8 Handling outliers using Z score

There are some feature with negative values. Negative values for these features are not valid. We will remove these rows. This will be our final step in data cleaning.

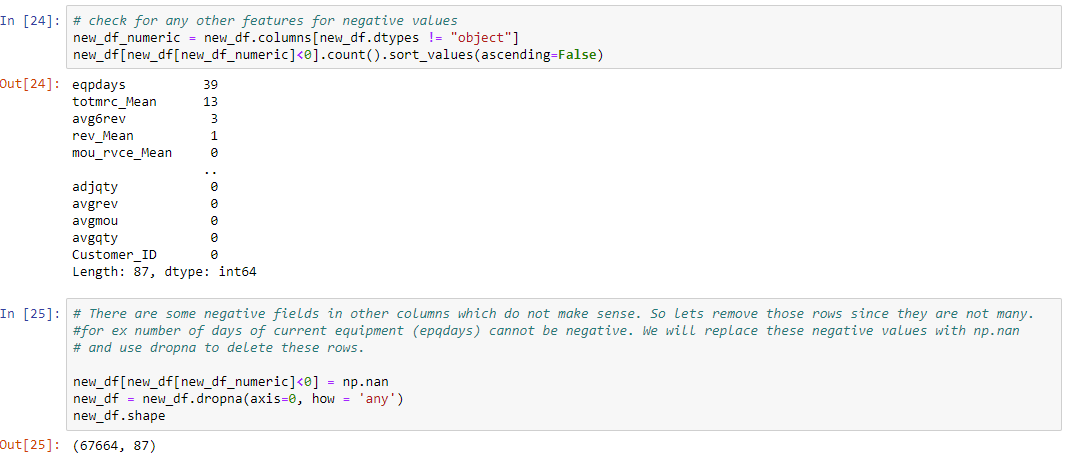


Figure 4.5 9 Removing rows with Negative values

### **3. Data Transformation:**

Dataset may contain features with different units of measurements which might vary largely. For example the data set might have cost which might be thousands and other variables which might be in single digits. Algorithms are sensitive to these kind of data as they treat both these values equally in terms of priority, especially algorithms which are distance based. To avoid this the feature should be normalized so all the features are given equal priority.

**Log Transformation:**

For highly-skewed feature distributions, one of the widely used methods is applying log transformation on the feature so very large or very small values do not impact the performance of the model. Log transformation will reduce the range of the values caused my outliers. But since log of 0 is infinity or not defined, we have to make sure we convert the 0 values by adding a small number to the whole column to avoid this situation.

In this case from **Figure 4.6 6** looks like most of the columns need log transformation, so let’s apply log on all the columns.

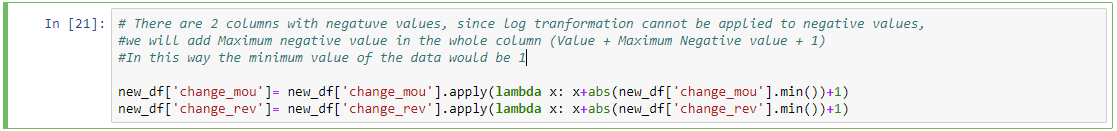
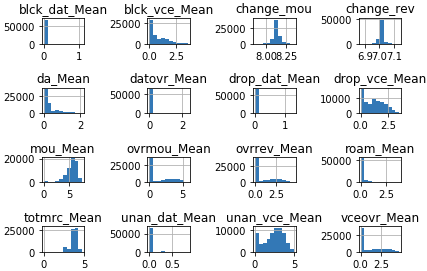
There are 2 columns with valid negative values, since log transformation cannot be applied to negative values, we will add Max negative value to the whole column plus 1 (Value + Maximum Negative value + 1) so, the minimum value in the feature will be 1.

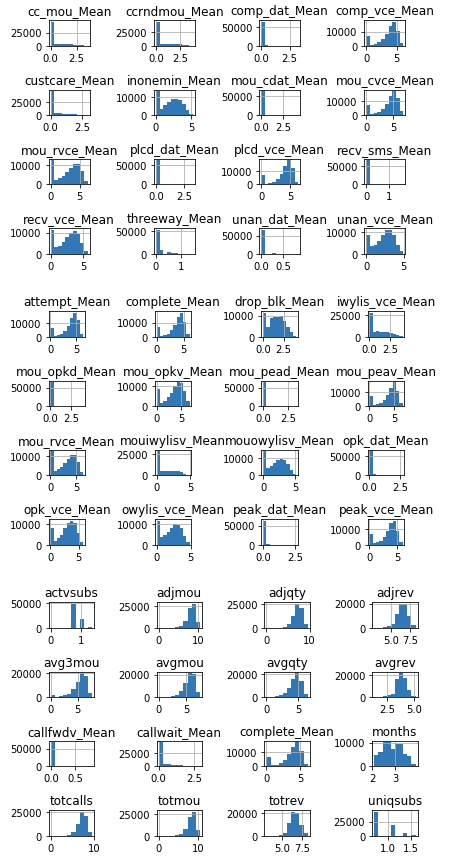
Figure 4.5 10 Transforming Negative Values

Now all the features are ready for log transformation. Log transformation will be only done on numeric columns.

Figure 4.5 11 Log Transformation

After applying log function to features the range of values have significantly reduced on all the features.





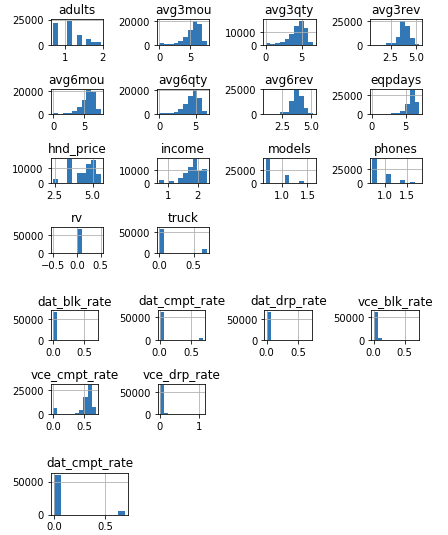


Figure 4.5 12 Data Distribution post log transformation

**Normalizing Numerical Features:**

Log transformation will reduce the range of values within each features, on top this it’s good to apply some sort of scaling on all the continuous variables. Scaling the feature will not change the shape of the distribution. Normalization is one of the scaling methods, doing this each variable is treated equally when applying the algorithms.

For normalizing numerical features we will use minimax scaler from Sklearn. Applying min max scaler will reduce the range of each feature between 0 and 1. Min max scaler serves better when the feature distribution is not normal. In our case since most of the feature distribution are not normal we will apply min max scaler.

MinMax transformation is given by

**(X – X (min)) / (X (max)- X (min)**

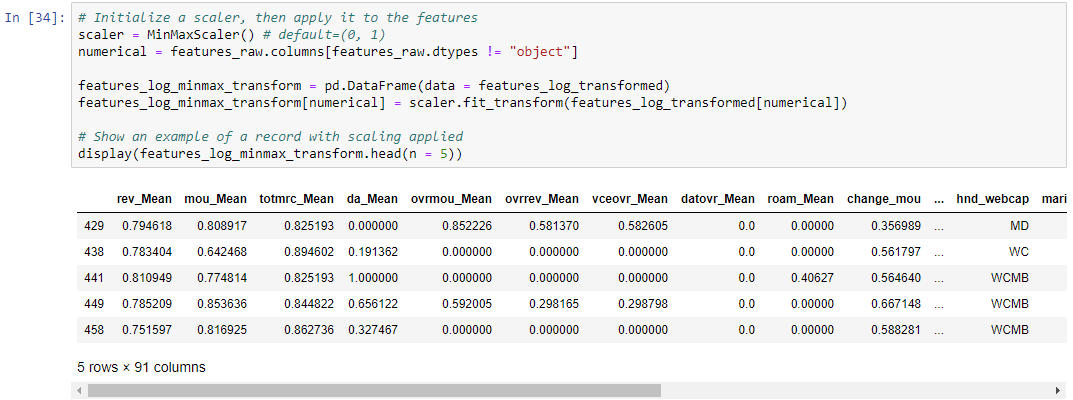


Figure 4.5 13 Normalizing Numerical Features

**Data Preprocessing:**

The next step in data transformation is data preprocessing. Many of the machine learning algorithm input to be numeric, but we have several features which are not numeric. This is the step where we convert non-numerical values to numerical values.

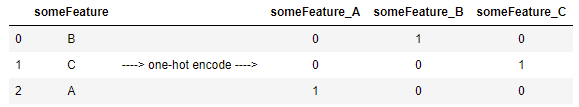
From the **figure 4.15 13** there are several features which are not numeric. There are features which are categorical. Most of algorithms required data to be numeric, so we need to convert these features which are not numeric to numeric. The most commonly used process for this conversion is one-hot encoding from pandas library. This process creates a dummy variable for each category in that variable. For example, let’s assume that a feature called grade has 3 categories A, B and C. One-hot encoding converts this into 3 features, fgrade\_A, grade\_B and grade\_C and assigns 0 if that category was not present for that row and 1 if it’s present.

Figure 4.5 14 One-Hot Encoding

We can use **pandas.get\_dummies ()** to done one-hot encoding on

'features\_log\_minmax\_transform' dataset. The output label churn is numeric with 0 representing not churn and 1 representing churn, so we don’t have to convert this.

Figure 4.5 15 One-Hot Encoding Output

After applying this process we now have 181 features. So we have the data ready for training.

### **4. Splitting Data into Training and Test sets:**

Now that the data is ready for modelling we need to divide the data into training and testing sets. Training set data is used to train the model and testing dataset is use to testing the model. In our case we will split the data randomly to assign 80% of the data as training data and 20% of the data as testing data.

We will use sklearn.train\_test\_split () get training and testing data sets.

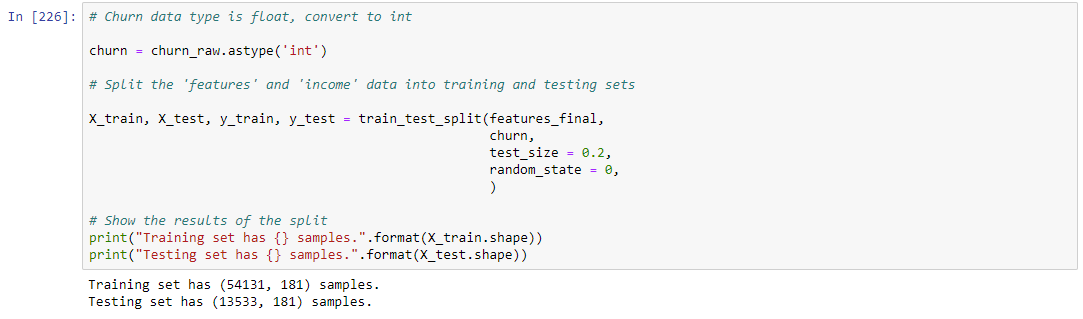


Figure 4.5 16 Splitting Data into training and test sets

From above we now have 54131 rows and 181 columns for training data and 13533 rows and 181 columns for testing.

Having 181 features in your data set will be computationally very expensive, so the next step is feature engineering where we can create new features from exiting variables and feature selection where we select subset of 181 features for modeling.

## **4. 6 FEATURE ENGINEERING / FEATURE SELECTION**

Feature engineering is a process where domain knowledge can be used to create new features from the existing dataset. These additional feature’s which help in the predictive power of machine learning algorithm.

In our dataset we create additional feature’s as below

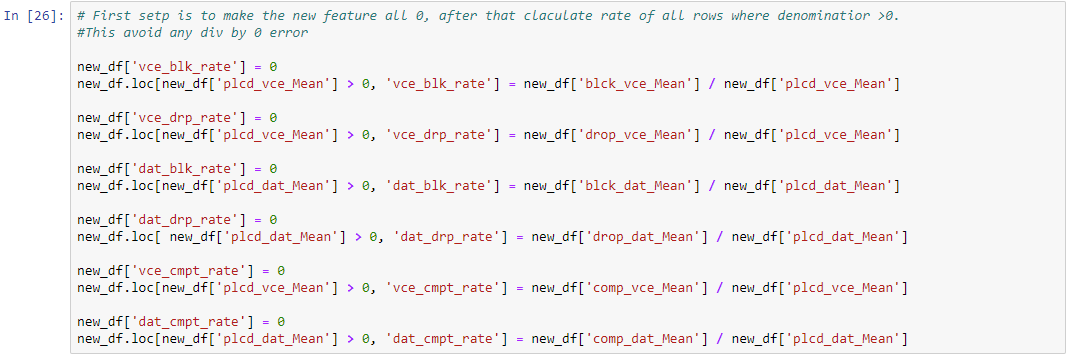


Figure 4.6 1 Feature Creation

Network experience is one of main reasons whether a customer will churn or not, if customer is experience networks issues there is a high probability that customer would churn. So having better understanding on network issues in the data set is important for modeling.

The raw integer values of drops, blocks, and completions do not give a full picture. For example a customer can make 10000 calls and can have 100 drops which equates to 1% of drops or blocks. Other customer could make 100 calls and have 10 drops or blocks which equated to 10% drop or block rates. So going by the raw integers do not tell if the customer is experiencing any network issues or not. So creating features with rates of drop, block and completion will give a better idea of network quality for each customer.

After feature engineering and on-hot encoding the total features in the dataset is at 181. Trying to use all 181 features for modeling provide to be very computationally expensive. Running a model or fine tuning the model using gridsearchCV proved to be very computationally expensive. So in this section we will reduce the number of features actually used for modelling.

Machine learning algorithms learn from different features with the data set. The training and performance of the model depends on the features. Having redundant features or features which are highly correlated or features which have the same value for any output are no help to modelling and will slow down the training and might over fit. [12]

Feature selection is the process of selecting the most important features for model training.

Feature selection provides several advantages in model training:

1. Models with less features are simpler and easy to explain.

2. Easy to implement, less computationally expensive.

3. Reduces overfitting and easy to generalize to new data

4. Removes data redundancy.

5. Faster training time.

6. Less errors.

Below are the 2 methods which are used for feature selection on this dataset.

**1. Filter Methods:**

Filter methods check for data uniqueness, evaluates the data and selects features which are unique, this can be done by checking the variance. This methods will rank the feature by a threshold like variance and removes the n features according to the threshold. The verification is done at a feature level. Whereas multivariate methods will take other features in account and delete features which are redundant or correlated. This process should be applied to training data in order to avoid overfitting. [11]

**a. Removing Constant Features:**

Constant features are the variables where all the data points have same value for any output. These kind of features offer no value to the algorithm since any data point in the feature will lead to the same result. So as part of feature reduction process its better we removed these features from the data.

To do so we will use Variance Threshold function. This function will require a threshold, if the function meets the threshold conditions and then the feature will be deleted. We can use sklearn.feature\_selection import VarianceThreshold for this. [11]

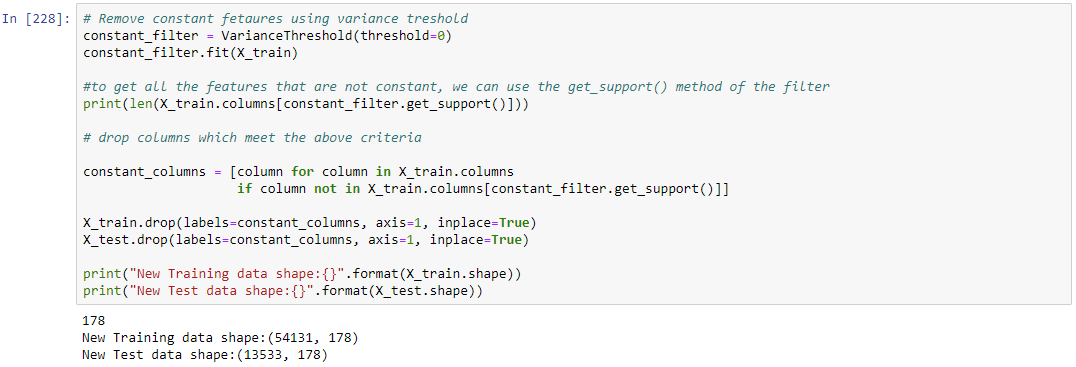


Figure 4.6 2 Feature Selection - Removing Constant Features

Based on below result, 3 of the features in the dataset are constant. Transform method can be used to remove feature with constant values.

**b. Removing Correlated Features:**

For any model, features with correlation between with output variable is very useful in predicting the output variable. But if there is high correlation between the input variables then data is becomes redundant to predict the output. So it’s a good practice to remove one input features with high correlation and only retain only one of the two variables as input feature for prediction.

Corr() method can be used to remove correlated features. This methods creates a correlation matrix between all the features in the dataframe. Once we have the matrix we can loop through this and find features which meets the correlation threshold criteria and add them into the correlated features variable. We can then delete all the features in this variable. [11]

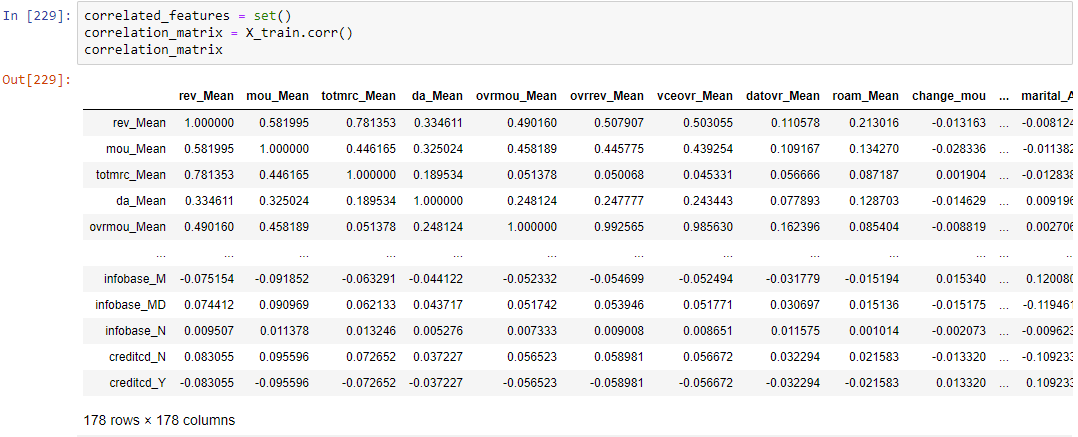


Figure 4.6 3 Correlation Matrix

In the next step we will remove any feature which has correlation of more than 80% to any other feature.

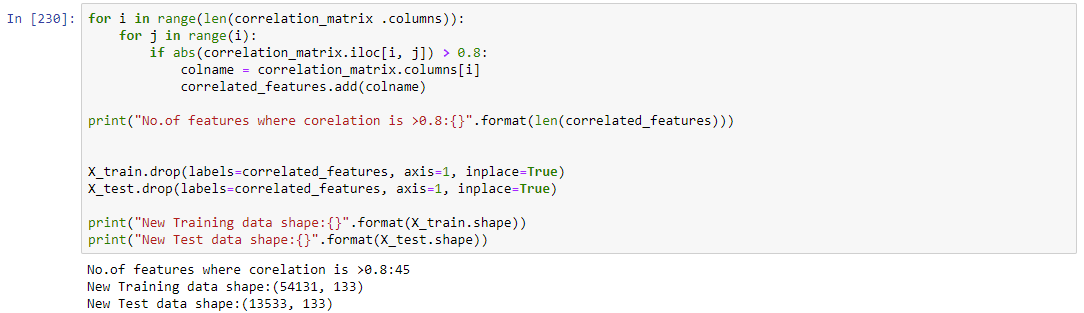


Figure 4.6 4 Removing Correlated features

Based on the above we have about 45 features where the correlation is greater than threshold which is 80%. After removing highly correlated features and constant features we are now down to 133 features from 181 features. 133 features are still high number of features and can be computationally expensive so we need to reduce these features further for less model complexity and to avoid any overfitting. To further reduce the number of features we will use wrapper methods which will be discussed in next section.

**2. Wrapper Methods:**

Filter methods do not require a machine learning model to evaluate performance it more based on the statistics. Wrapper methods need machine learning for performance evaluation. It uses machine learning model to find the best features for model performance. There are several wrapper methods like forward selection, backward elimination and recursive elimination. With high number of features in our current dataset forward and backward elimination will be highly computationally expensive so we will use Recursive feature elimination. [11]

**Recursive Feature elimination:**

Recursive feature elimination uses greedy search to find the best performing features. In each iteration it creates the model and selects best features in each iteration. In the next round it will create subsequent model with the left over features and so on and selects the best N features mentioned in the algorithm

[https://scikitlearn.org/stable/modules/generated/sklearn.feature\_selection.RFE.html#sklearn.feature\_selection.RFE](https://scikitlearn.org/stable/modules/generated/sklearn.feature_selection.RFE.html%23sklearn.feature_selection.RFE)

Let’s go over the difference between filter and wrapper methods for better understanding.

* Filter methods do not use machine learning model to decide whether a feature is good or not whereas wrapper methods use machine learning model in its decision making process to select the best N features.
* Filter methods are computationally less expensive and faster since any trianing is not involved in the decision making process, where as wrapper methods are computationally expensive and take longer time. When the dataset is large wrapper methods might not be an efficient solution to consider.
* In case of less data filter methods might fail to find the best set of feature where it’s hard to find the statistical correlation of features due to less data, whereas wrapper methods will all find the best features
* There is a possibility that using wrapper methods can lead to overfitting since selection is based on training a model, filter methods will not run into this scenario. [11]

Using wrapper method we will choose the best 15 features for our data set.

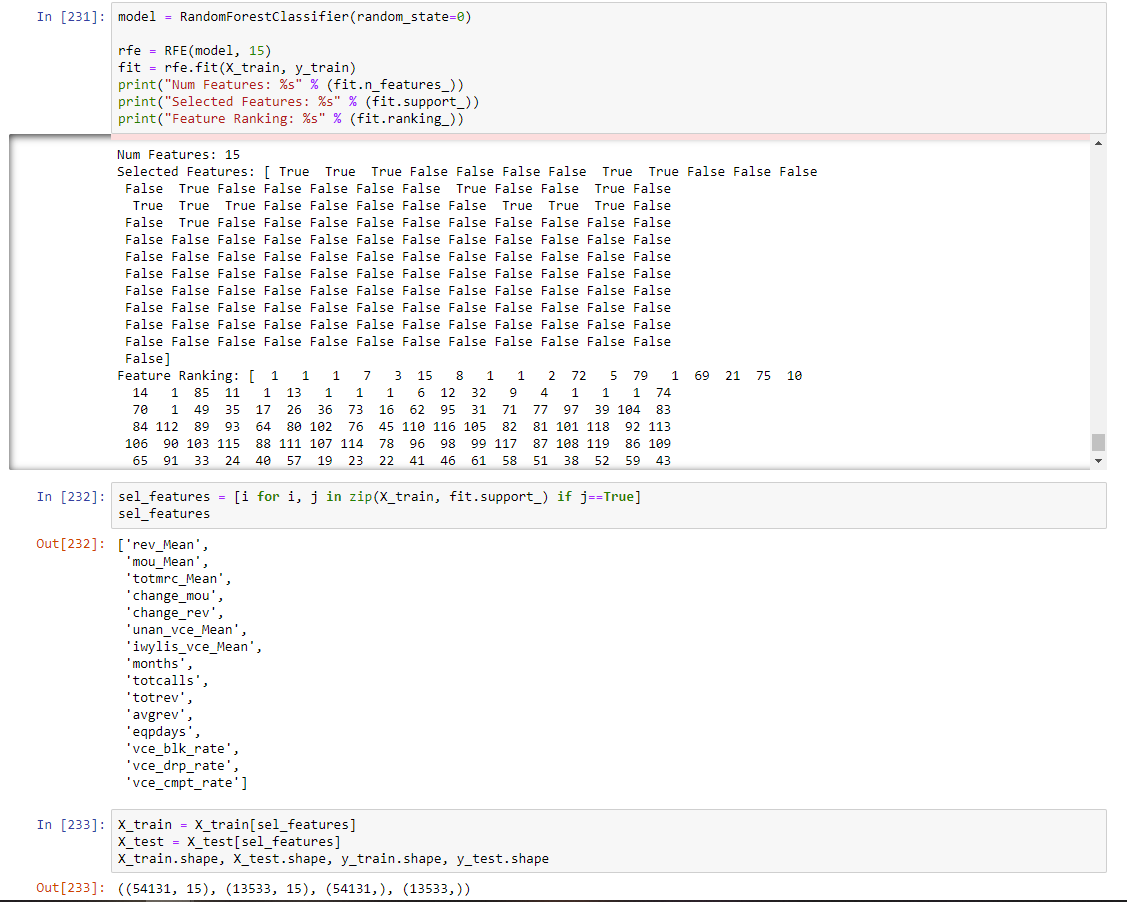


Figure 4.6 5 Feature Selection- Wrapper Method

Based on the above we are ready to do modelling based on the 15 features. We will use these features for training and then test the model on test set. Since no one algorithms for all data sets, we will try multiple algorithms choose the best algorithm and tune that algorithm using these 15 features in our dataset.

## **4.7 MODEL TRAINING**

Now that the data is cleaned and feature selection is complete we will use this new dataset to start modeling. In the model training step, we will try different models and select the best performing model and tune the hyper –parameters of that model. Multiple model are tried since no one models works the best for different data sets.

To efficiently predict and evaluate each model performance we will create a function that will make it easier to predict, test and evaluate all the model at once using different training data set sizes

Our implementation here will do the following:

* Import all the required metrics from Sklearn. For this project we will use f-score and accuracy as our evaluation metrics.
* Fit each learner to training data and record the training time
* Perform prediction for testing set.
* Check prediction time for each learner.
* Select beta value for f-score based on the problem at hand.
* Calculate F-score and accuracy for each learner.

Since we do not have a base model to compare the predictions against, we compare against Naïve predictor. The reason for using naïve predictor is we want check how the model performs with no intelligence. In real world scenarios we normally have a older model or any model based on research paper o compare our model performance against.

When the model has no intelligence, let’s say the model always predicts that customer is going to churn, in this case we do not have any TN or FN, since our model make no predictions as not churn. In this case accuracy becomes equal to precision (TP/(TP+FP)). The denominator here is actual number of data points. Our recall will be (TP/(TP+FN)) will be 1 as there are no FN

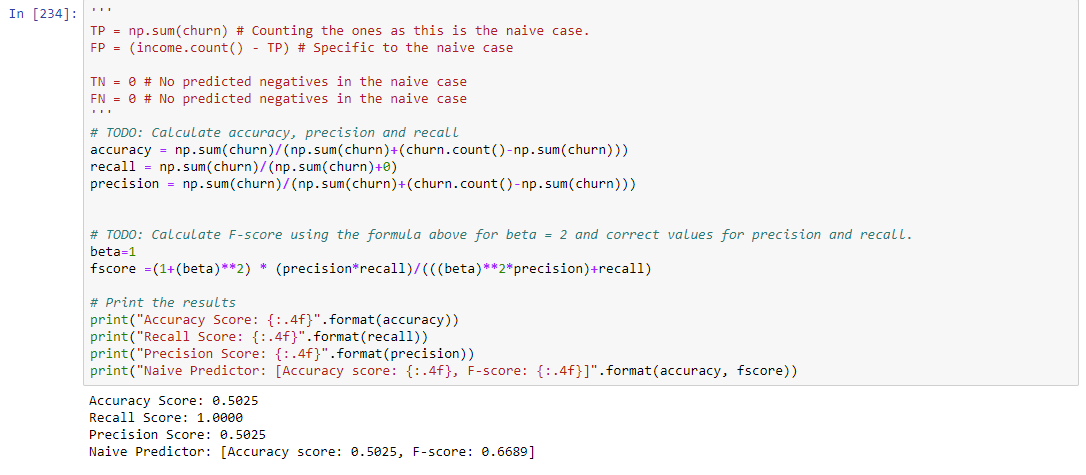


Figure 4.7 1 Naive Predicator

Based on above our Naive predictor accuracy is at 50% and f-Score of 66%. We will use this as baseline to compare performance of our different models. The next step is to train models on training set and test those modes on our test set to evaluate the performance.

### **4.7.1 Initial Model Evaluation:**

In this step will try different models with our data set and check which model performs the best on this dataset. We will pick the best 2 models and tune the hyper-parameters to optimize the performance. Below are the list of models

**1. Logistic Regression:**

Logistic Regression is one of the basics regression models and used as a baseline model for classification problems. Logistic regression is mainly used for 2-class classification models. Since our problem is a 2-class classification model we will use this model to compare performance against other models.

Logistic regression is used mainly when the output variable has 2 classes, meaning it has 2 categories where the output can be under. For example if email is spam or not, in this case you have only two possible outcomes. Either email is spam or email is not spam. It predicts the probability of binary event happening using a logit function and determine the class. [18]

Logistic regression is very computationally efficient since its very straight forward model, it is also easy to implement and easy to interpret. It’s special kind of linear regression where the output variables is categorical. Due to its ease its one of widely used classification models.

One of the disadvantages of logistic regression is it cannot handle higher number of features or variables. Also the features have to be linear, if the features is non-linear then transformation is required. It will not perform well in conditions where the features are correlated to each other and correlated to target variable. [18]

**2. Support Vector Machines**

Support vector machine often called as SVM is one of more the most accurate and powerful models compared to Logistic regression and decision trees. It can handle non-linear inputs using a kernel trick. It is used in multiple applications such as classification of emails, intrusion detection, face detection, news articles and web pages, classification of genes and many other applications.

The whole goal SVM is to find an optimal hyperplane. Hyperplane is a plane which separates classes. The goal of SVM is to find the hyperplane with largest margin. Due to this reason it’s also called as discriminative classifier. [19]

In multidimensional space SVM creates a hyperplane to separate each class. This is done in an iterative way to minimize the error. The idea is to find a hyperplane which has the highest margin to divide the classes. Hyperplane is a decision boundary which separates the data points into different classes and margin is the distance between the two lines on the closest class data points. Higher the margin lower the error and hence good margin, lower the margin high error and hen bad margin. [19]

The data points closet to hyper plane are called support vectors, these support vectors will define separating line from margin calculations. The support vectors are main points to construct the classifier.

SVM models have high accuracy and use less memory as they use subset of data points which are the support vectors to construct the decision plane. [19]

Training time and prediction time for SVM are generally very high compared to other models, so SVM might not be a suitable model for large datasets.

**3. Random Forests:**

Random Forests is an ensemble method meaning it will use multiple learners to build a strong learner. Decision tress is the base model for random forests. It randomly selects data in each iteration builds a learner. These individual decision tress are called forest. [20]

Random forest works below

* Random samples are selected from a given dataset
* For each of these sample a decision tree is constructed and results are predicted from each of these decision tress.
* Each predicted results from above step get voted.
* Final model is chosen based on the prediction results with highest number of votes.

Random forests are highly accurate models and does not suffer from overfitting problem because of multiple decision tress used in the process. Random forest are used for both Regression and classification problems. They can handle missing values by imputing median values in the missing value points for a particular feature. They also have feature importance method which can tell you the most important features in the decision making process. Random forest are slow because of multiple decision trees and also hard to interpret.

**4. Ada Boost:**

Ada-boost or Adaptive Boosting is one of ensemble boosting technique proposed by Yoav Freund and Robert Schapiro in 1996. It combine several weak classifier to build a strong classifier. AdaBoost is an iterative ensemble method. AdaBoost sets weights in each classifier, providing higher weights to wrong classification points so in the next iteration the wrong classification are given higher weight for classification

Two conditions should be met for AdaBoost classifier:

* The AdaBoost classifier is trained on multiple weighted training data points.
* In every iteration, it fits these data points by minimizing the error.

It works in the following steps:

* The training subset is selected randomly for the first time.
* Based on previous accurate predictions in last iteration it iteratively trains the subsequent iteration
* In each iteration it assigns higher weights to wrongly classified points, so in the next iteration these points with high weights will get higher priority for classification.
* Based on the accuracy of the classifier, weight is assigned to each classifier, higher the accuracy higher weight for the classifier.
* The process is repeated until there are no more miss classification point or until it reaches the threshold number of estimators. [21]

AdaBoost is easy to implement, not prone to overfitting, sensitive to noisy data as it tries to fit each data point.

**5. XG Boost:**

Extreme Gradient Boosting mostly called XGBoost which is one of the boosting algorithms which uses gradient boosting at its core. It is an optimized distributed gradient boosting library. Boosting build on several weak classifiers and builds a stronger classifier. Its builds the weak learner assign higher weights to miss-classified points then in the next iteration it gives more priority to miss-classified points and fixes those issues. It iteratively it fixes all the mis-classified points. Based learner for Xgboost is tree ensembles. Iteratively tress are grown one after the other and in each iteration fix the past mistakes. [22]

The next step is to rain these model and check the accuracy, F-score and training time. We will then choose the best 2 models and tune the parameters further the algorithm.

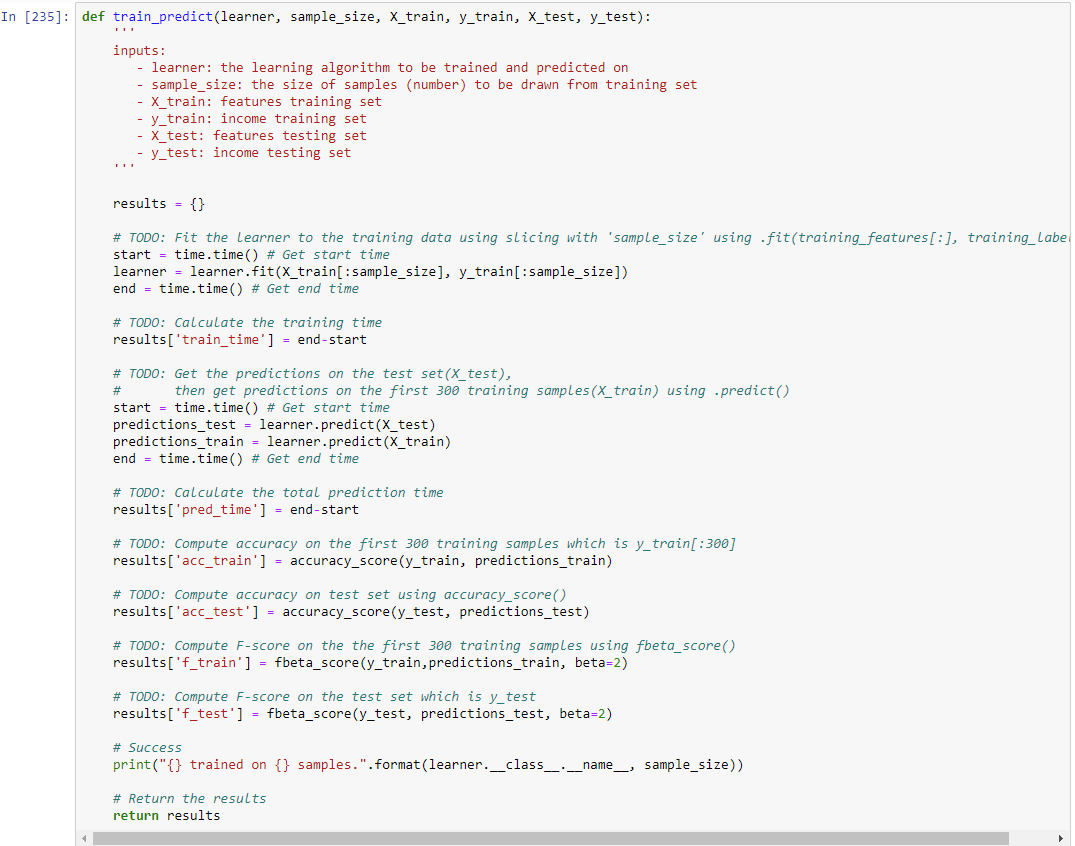


Figure 4.7 2 Training and prediction pipeline

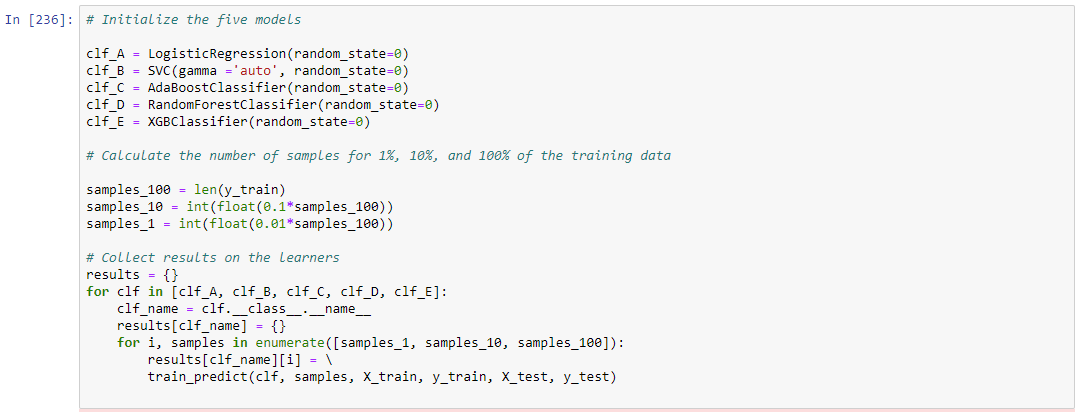


Figure 4.7 3 Execute the models

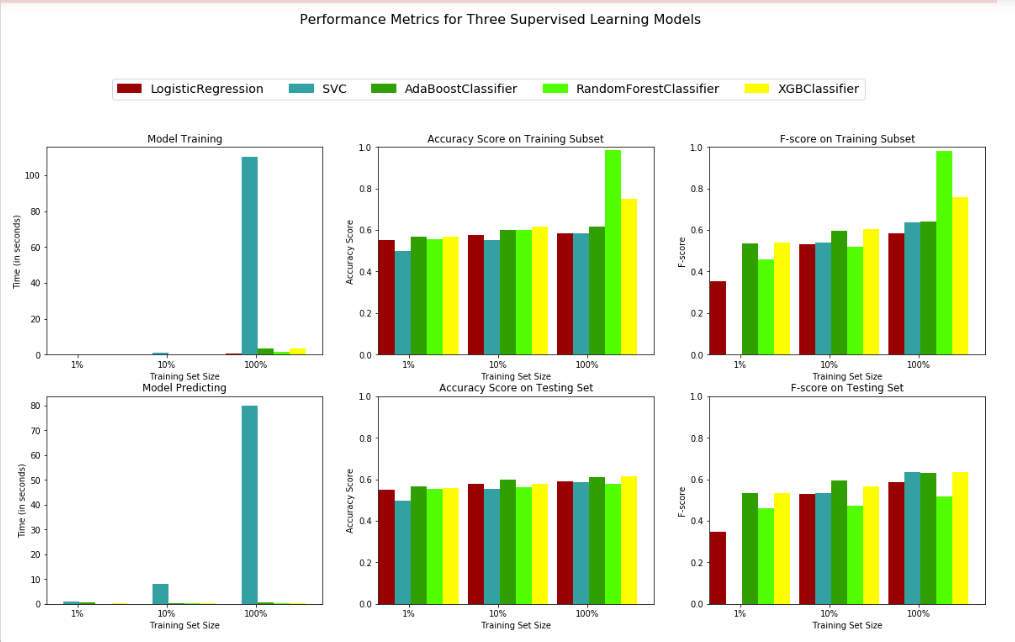


Figure 4.7 4 Models Performance Metrics

Based on the above chart as expected Boosting algorithms performed best and were able to generalize well on test set meaning they did not Ove fit and also SVM though the performance was good has the highest training and testing time. So we will use move forward with AdaBoost and XG Boost classifiers and tune the parameters to improve performance.

### **4.7.2 Model Tuning:**

The reason for model tuning is to select optimal Hyper-parameters for the algorithms which provides the best results. The hyper parameter values will vary from data set to dataset so it is always best practice to tune the hyper-parameters for optimal performance.

We can use 2 methods for hyper-parameter tuning. GridsearchCV or RandomizedsearchCV.

**1. GridsearchCV:**

Grid Search is an extensive search for selecting the hyper parameters of the model. We setup up a grid with multiple values for each hyper-parameter and the model is trained on each combination of these parameters and finally model with hyper parameters with best accuracy is selected. In this process every combination of the hyper-parameter is trained. For example if we have 5 parameters with 5 different values for each parameter then 3125 different models have trained and the hyper parameters of the model with best results out of these 3125 will be picked as optimum hyper-parameters. Sklearn offers gridsearchCV but computationally very expensive takes lot time and not efficient when the dataset is large with multiple features. [13]

**2. RandomizedsearchCV:**

Random search is similar to gridsearchCV except for the fact that it won’t test each and every combination rather it will pick random combinations from the parameter grid and test and find the best solutions. This process is highly efficient when the the dataset and feature set is large and it’s much faster than gridsearchCV. [13]

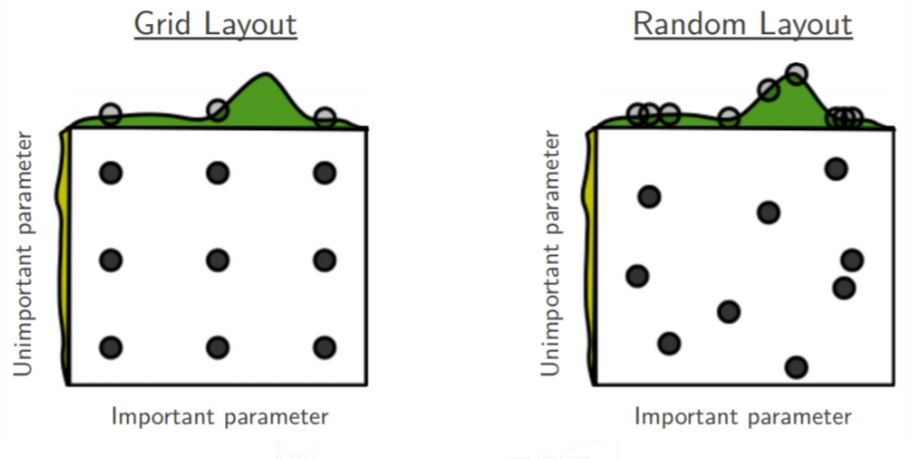


Figure 4.7 5 Hyper-parameter tuning Grid vs Random search

We will use GridsearchCV and RandomizedsearchCV from sklearn.

Based on the trial run did on both these techniques, i found gridsearchCV takes lot of time to complete. Running gridsearchCV on xgboost took 4 days to run. Whereas RandomizedsearchCV took around 20mins. So for this project I will be using RandomizedsearchCV.

GridsearchCV is very computationally expensive but very through. For this project we will use RandomizedsearchCV due to time taken to perform gridsearchCV.

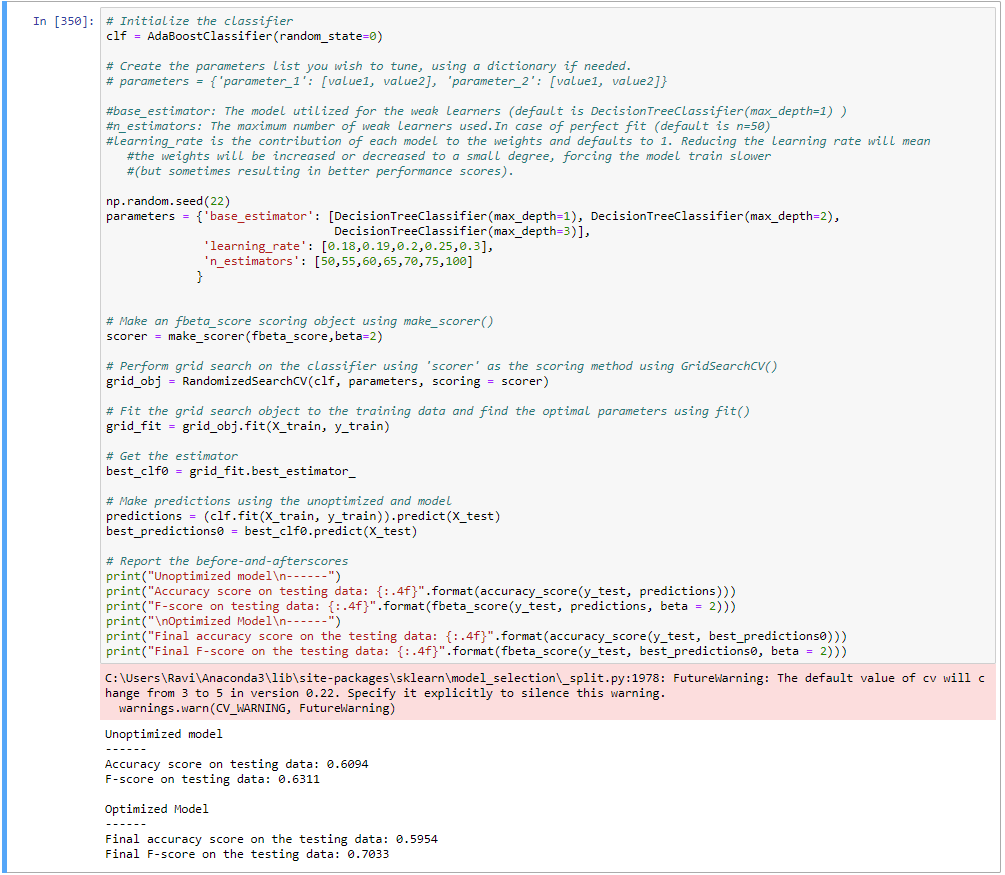


Figure 4.7 6 AdaBoost Tuning

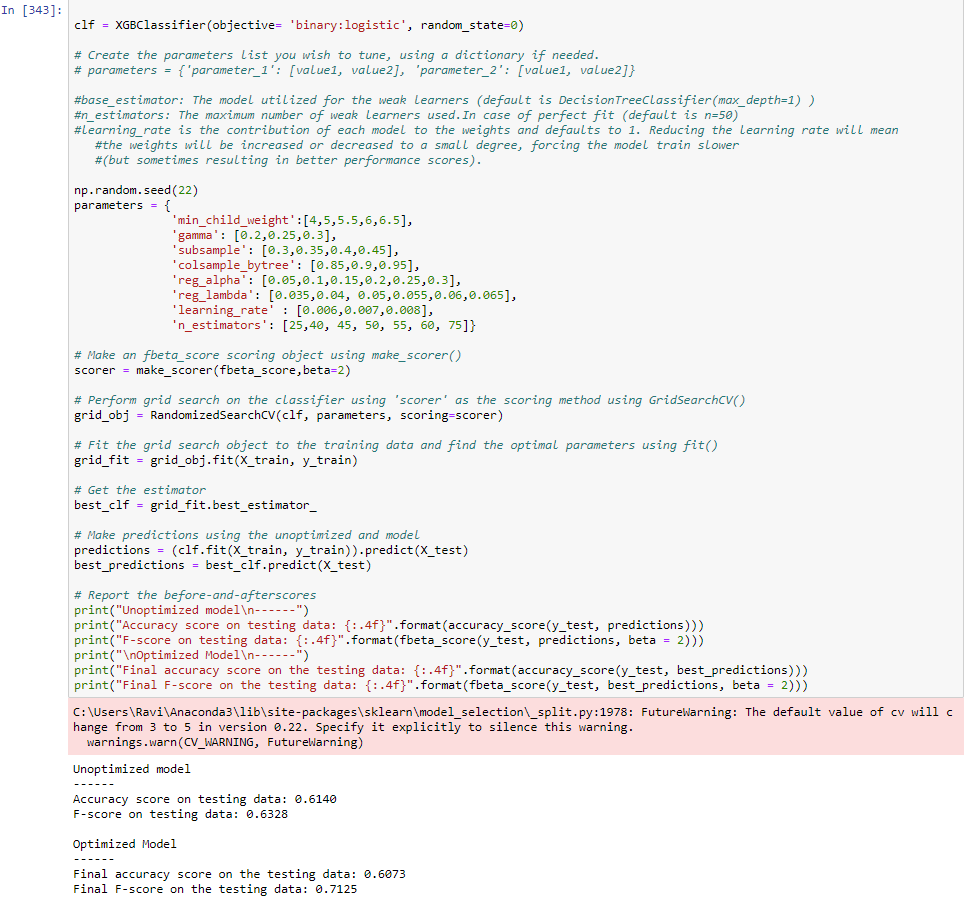


Figure 4.7 7 XGBoost Tuning

Xgboost provided slightly better results than AdaBoost with accuracy at 61% and F-score of 71%.

**XGBOOST FINAL ALGORITHM:**

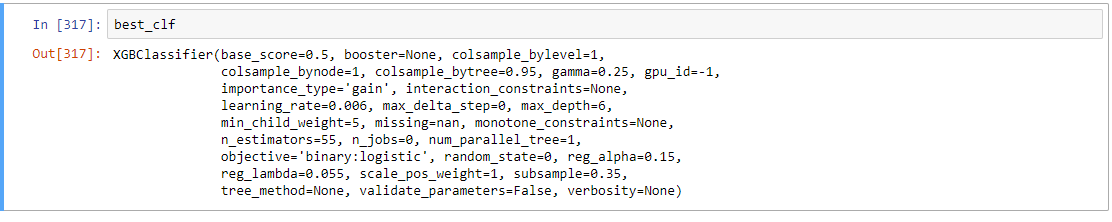


Figure 4.7 8 Final Model

## **4.8 MODEL EVALUATION**

Accuracy is good measure on how well the models is classifying the point. It’s the ratio of all the correctly classified point to all the points.

**Accuracy** can be used when the target label classes are well balanced. Accuracy won’t be a good measure for some problems like email is spam or not or if a patient has cancer or not.

**Precision** is the ratio of TP to all the positives. It tells how precise your predictions, this is useful when there is no room for error such as email is spam or not. When the email is spam it better be spam and it has to be precise, we don’t want emails which is labelled as spam not be spam.

**True Positives/(True Positives + False Positives)**

**Recall (sensitivity)** is the ratio of TP and TP+FN. For ex it’s the ratio of true spam messages to message which re classified as spam.

**True Positives/ (True Positives + False Negatives)**

For classification problems where classes are imbalanced where for ex there are 100 transactions and 2 were fraudulent and the rest 98 are not accuracy won’t be a good metric, since you classify all transactions as not fraudulent and you will be right 98% of the time.

For case like these precision and recall will be useful. We can use the combination of these 2 metrics to get F1 score which si weighted average of precision and recall. The F1 score can be between 0 and 1, 0 being the least F1 score and 1 being the highest.

**F-beta score** is combination of both precision and recall is give by

**F (beta) = (1+ (beta)^2) \*(precision recall)/((beta)^2\*precision) + recall)**

If beta=0.5 , more emphasis is placed on precision. when beta=2, more emphasis on recall, here we are choosing 2 because going higher will allow more FP, we dont want too many customers who are not churning showing as churn, since it will take too many customer care calls to people who are not churning and we don’t want to disturb these customers. [23]

In the case of this data set, the target variable is nearly balanced, so accuracy can be a good metric to evaluate model performance.

Also this is a high recall model, meaning we are OK with some FP (meaning customers who are not churn but predicted as churn) but NOK with any FN (meaning customer who are churn but model classifies as not churn). If we miss customers who are churning then we can’t reach out to customers to make them happy and probably they will churn, but its ok when customer is not churning but still labelled as churn since no harm happens due that.

Since ours is high recall model meaning more importance to recall the beta in Fbeta lets assign as 2. Beta of 1 mean equal importance is given to both precision and recall. If beta is less than 1 lest say 0.5 then more importance is given to precision and beta is >1 let’s say 2 then more importance is given to recall.

Based on the results from final model below are the results:

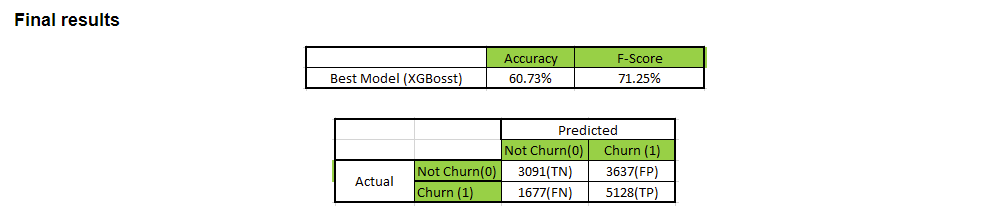


Figure 4.8 1 Final results

The final 2 steps of machine learning workflow Business evaluation and Prediction are not part of this project as this is just a case study project. In practical world these 2 steps are very important. If the Business goals are not met then further optimization is required like adding additional features or data and further feature engineering. Once business goals are met then the model will be deployed into production. After deploying the model into production the model has to be tuned periodically with new data feeding into the model.

# **5. CONCLUSION**

Based on results and several trials we can conclude that **F-score 72%** is a decent score for predicting churn. Based on these results customer care can reach out to the potential customer who will churn and try to resolve their issue so customer will not churn.

To improve results further will require additional features or improve data quality. Since this a case study project, further data inclusion is not in the scope of this project. Data science team can work with business teams and see if any additional features to further improve the model performance.

# **6. REFERENCES**

1. https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

2. https://www.bogotobogo.com/python/scikit-learn/scikit\_machine\_learning\_Data\_Preprocessing-Missing-Data-Categorical-Data.php

3. https://jakevdp.github.io/PythonDataScienceHandbook/03.04-missing-values.html

4. https://www.bogotobogo.com/python/scikit-learn/scikit\_machine\_learning\_Data\_Preprocessing-Missing-Data-Categorical-Data.php

5. https://www.pluralsight.com/guides/cleaning-up-data-from-outliers

6. https://scikit-learn.org/stable/modules/preprocessing.html#imputation-of-missing-values

7. https://blogs.sas.com/content/iml/2011/04/27/log-transformations-how-to-handle-negative-data-values.html

8. https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/

9. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

10. https://www.datacamp.com/community/tutorials/adaboost-classifier-python

11. https://www.datacamp.com/community/tutorials/feature-selection-python

12. https://stackabuse.com/applying-filter-methods-in-python-for-feature-selection/

13. https://blog.usejournal.com/a-comparison-of-grid-search-and-randomized-search-using-scikit-learn-29823179bc85

14. https://www.datacamp.com/community/tutorials/principal-component-analysis-in-python

15. https://www.statista.com/statistics/283511/average-monthly-churn-rate-top-wireless-carriers-us/

16. Understanding How Data Powers Big Business: Bill Schwarz

17. https://scikit-learn.org/stable/modules/impute.html#iterative-imputer

18. https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

19. https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

20. https://www.datacamp.com/community/tutorials/random-forests-classifier-python

21. https://www.datacamp.com/community/tutorials/adaboost-classifier-python

22. <https://www.datacamp.com/community/tutorials/xgboost-in-python>

23. www.psycoderblog.com

# **APPENDIX A**

Description of Files

1. Churn\_data.csv: Original churn data file

2. Predict\_churn.IPYNB: Jupyter Notebook file for the all the code.

3. Visuals.py: Functions which were used for visualizations

4. Project Report.doc: Final project report.