CIND 119 Class Project:

Predicting Customer Churn

Members

Ahmed, Shahzad, shahzads.ahmed@ryerson.ca
Chandra, Akash, akash.chandra@ryerson.ca
Nguyen, Thi Ngoc Thanh, thingocthanh.nguyen@ryerson.ca

I. Summary:

According to IBM (2020), "data science combines the scientific method, math and statistics, specialized programming, advanced analytics, AI, and even storytelling to uncover and explain the business insights buried in data." [1] With the emphasis on using math & statistics, programming, domain knowledge as well as story-telling skills, data scientists are experts who use the above tools and domain knowledge to make sense of patterns, giving answers to any question business stakeholders may have, and/or going further to predict possible actions and their outcomes of those actions.

In this project, as a team of data scientists, our consultants are working together to solve a business problem for our client - a telecommunications company with a predictive analytics business problem. Based on historical data on the customers' phone usage, the client would like to analyze and predict which customers will be likely to churn in the future. "Customer churn, also known as customer attrition, is when someone chooses to stop using your products or services" (Qualtrics, 2022).[2] Most companies would want to control the average churn rate at a feasible level, as a high churn rate would impact brands, costs, customer engagement metrics (lower customer lifetime value (CLV) and higher customer acquisition costs (CAC) etc.), as well as the company's long-term growth.

The ability to segment customers and predict the customer churn, through data analytics, machine learning & predictive modeling methods, would be a great business advantage for our client. Thus, this project aims to find a solution to our client's "customer churn" business problem by Exploratory Data Analysis (EDA) and Predictive Modeling/Classification techniques (Classification using Decision Tree, Naïve Bayes). The dataset we are using is "churn.arff". There are 3333 rows, 21 columns/attributes in this dataset.

---

[1] IBM (2020): https://www.ibm.com/cloud/learn/data-science-introduction

[2] Qualtrics (2022): https://www.qualtrics.com/experience-management/customer/customer-churn/. Date Retrieved : July 31, 2022.

In the first part of the project, we will carry out Data Preparation to understand the data and identify the research questions such as: "How is the data distributed?", "Which attributes seem to be correlated?", "Which attribute can be included/eliminated in the analysis based on statistics?" and so on. Within the second part, we will build the classification machine learning models based on the Decision Tree and Naïve Bayes algorithm.

The main tools used in the scope of this project are Python, its packages for Data Analysis, Visualization, Machine Learning (Pandas, Numpy, Matplotlib, Seaborn, Pandas Profiling, Scikit-learn, Imbalanced-learn) along with additional visualization tools (Graphviz).

II. Workload Distribution

Member Name	List of Tasks Performed
Ahmed, Shahzad	Data Prep & EDA, Classification Tree Models
Chandra, Akash	Data Prep
Nguyen, Thi Ngoc Thanh	Data Prep & EDA, Naïve Bayes Models, Organizing & Completing the Project Report

III. Data Preparation & Exploratory Data Analysis (EDA)

3a. Data Dictionary

Below is a brief description of all columns and their values in the dataset "churn.arff":

Column	Explanation	Variable Type	Data Type
State	Customer's state	Categorical (Nominal)	object
Account Length	Integer number showing the duration of activity for customer account	Quantitative (Continuous)	int64
Area Code	Area code of customer	Categorical (Nominal)	int64
Phone Number	Phone number of customer	Categorical (Nominal)	object
Inter Plan	Binary indicator showing whether the customer has international calling plan	Categorical/ Binary (yes, no)	object
VoiceMail Plan	Indicator of voice mail plan	Categorical/ Binary (yes, no)	object
No of Vmail Mesgs	The number of voicemail messages	Quantitative (Discrete)	int64
Total Day Min	The number of minutes the customer used the service during day time	Quantitative (Continuous)	float64

Total Day calls	Discrete attribute indicating the total number of calls during day time	Quantitative (Discrete)	int64
Total Day Charge	Charges for using the service during day time	Quantitative (Continuous)	float64
Total Evening Min	The number of minutes the customer used the service during evening time	Quantitative (Continuous)	float64
Total Evening Calls	The number of calls during evening time	Quantitative (Discrete)	int64
Total Evening Charge	Charges for using the service during evening time	Quantitative (Continuous)	float64
Total Night Minutes	Number of minutes the customer used the service during night time	Quantitative (Continuous)	float64
Total Night Calls	The number of calls during night time	Quantitative (Discrete)	int64
Total Night Charge	Charges for using the service during night time	Quantitative (Continuous)	float64
Total Int Min	Number of minutes the customer used the service to make international calls	Quantitative (Continuous)	float64
Total Int Calls	The number of international calls	Quantitative (Discrete)	int64
Total Int Charge	Charges for international calls	Quantitative (Continuous)	float64
No of Calls Customer Service	The number of calls to customer support service	Quantitative (Discrete)	int64
Churn	Class attribute with binary values (True for churn and False for not churn)	Categorical/ Binary (TRUE, FALSE)	object

From the table above, most of the columns are quantitative/numerical; though there are some categorical/text/object columns exist. Those categorical columns are - State, Area Code, Phone Number, Inter Plan, VoiceMail Plan, and Churn. We will have to use encoding technique to transform those categorical columns into numerical values, before building predictive models by machine learning.

The Churn column is our target class column here.

3b. Missing Values & Duplications

After reading the “churn.arff” dataset into a Pandas dataframe (“df”), we’ve captured a snapshot of the data with 21 columns, 3333 rows:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 21 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   State                         3333 non-null   object
 1   Account Length                3333 non-null   float64
 2   Area Code                     3333 non-null   object
 3   Phone Number                  3333 non-null   object
 4   Inter Plan                    3333 non-null   object
 5   VoiceMail Plan                3333 non-null   object
 6   No of Vmail Mesgs             3333 non-null   float64
```

```
 7   Total Day Min                   3333 non-null   float64
 8   Total Day calls                 3333 non-null   float64
 9   Total Day Charge                3333 non-null   float64
 10  Total Evening Min               3333 non-null   float64
 11  Total Evening Calls             3333 non-null   float64
 12  Total Evening Charge            3333 non-null   float64
 13  Total Night Minutes             3333 non-null   float64
 14  Total Night Calls               3333 non-null   float64
 15  Total Night Charge              3333 non-null   float64
 16  Total Int Min                   3333 non-null   float64
 17  Total Int Calls                 3333 non-null   float64
 18  Total Int Charge                3333 non-null   float64
 19  No of Calls Customer Service    3333 non-null   float64
 20  Churn                           3333 non-null   object
dtypes: float64(15), object(6)

memory usage: 546.9+ KB
```

Counting the number of missing values for each column by `df.isna().sum()`

```
State                           0
Account Length                  0
Area Code                       0
Phone Number                    0
Inter Plan                      0
VoiceMail Plan                  0
No of Vmail Mesgs               0
Total Day Min                   0
Total Day calls                 0
Total Day Charge                0
Total Evening Min               0
Total Evening Calls             0
Total Evening Charge            0
Total Night Minutes             0
Total Night Calls               0
Total Night Charge              0
Total Int Min                   0
Total Int Calls                 0
Total Int Charge                0
No of Calls Customer Service    0
Churn                           0
dtype: int64
```

Counting the number of duplicated rows for the dataset by `df.duplicated().value_counts()`

```
False    3333
dtype: int64
```

We found no missing values and duplicates in the dataset.

3c. Statistical Summary for numerical columns (max, min, mean, standard deviation)

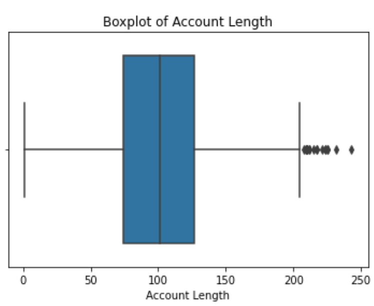
Let's look into the statistical table of our customer churn dataset:

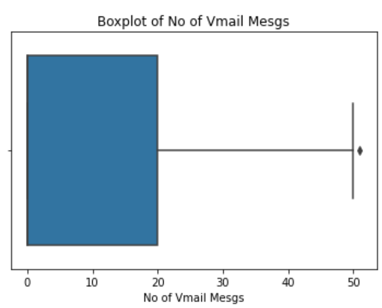
index	count	mean	std	min	25%	50%	75%	max
Account Length	3333.0	101.0648065	39.82210593	1.00	74.00	101.00	127.00	243.00
No of Vmail Mesgs	3333.0	8.099009901	13.68836537	0.00	0.00	0.00	20.00	51.00
Total Day Min	3333.0	179.7750975	54.4673892	0.00	143.70	179.40	216.40	350.80
Total Day calls	3333.0	100.4356436	20.06908421	0.00	87.00	101.00	114.00	165.00
Total Day Charge	3333.0	30.56230723	9.259434554	0.00	24.43	30.50	36.79	59.64
Total Evening Min	3333.0	200.980348	50.71384443	0.00	166.60	201.40	235.30	363.70
Total Evening Calls	3333.0	100.1143114	19.92262529	0.00	87.00	100.00	114.00	170.00
Total Evening Charge	3333.0	17.08354035	4.310667643	0.00	14.16	17.12	20.00	30.91
Total Night Minutes	3333.0	200.8720372	50.57384701	23.20	167.00	201.20	235.30	395.00
Total Night Calls	3333.0	100.1077108	19.56860935	33.00	87.00	100.00	113.00	175.00
Total Night Charge	3333.0	9.039324932	2.275872838	1.04	7.52	9.05	10.59	17.77
Total Int Min	3333.0	10.23729373	2.791839548	0.00	8.50	10.30	12.10	20.00
Total Int Calls	3333.0	4.479447945	2.461214271	0.00	3.00	4.00	6.00	20.00
Total Int Charge	3333.0	2.764581458	0.753772613	0.00	2.30	2.78	3.27	5.40
No of Calls Customer Service	3333.0	1.562856286	1.315491045	0.00	1.00	1.00	2.00	9.00

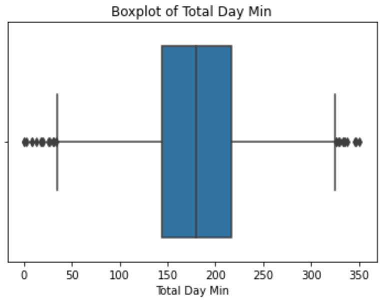
From the table above, 11 out of 15 of the numeric attributes have a minimum value of 0. Those might be customers, or missing values recorded as 0 – which might be another good business problem about data quality to explore further. However, as long as the overall data seem not to contain missing values & duplication, we will accept the dataset as is within the scope of this project.

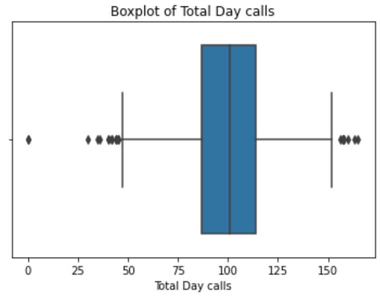
3d. Outliers Detection for numerical columns

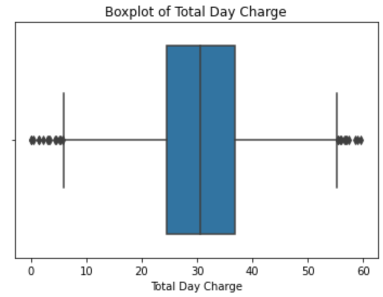
We can detect & show the outliers of the attributes using boxplot:

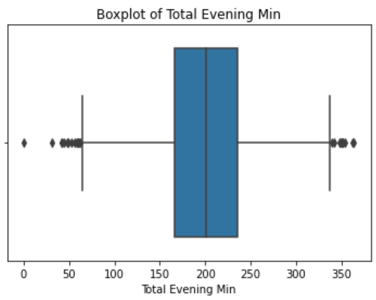


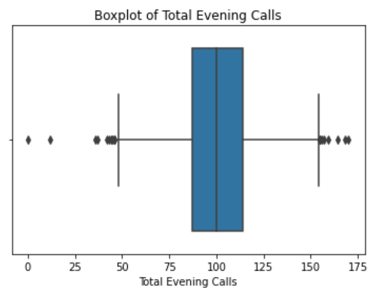


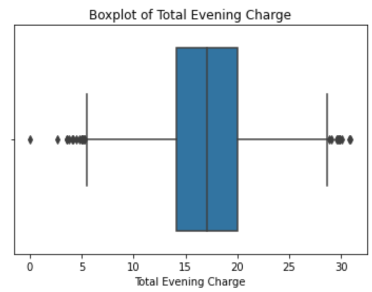

Boxplot of Total Day Min
0 50 100 150 200 250 300 350
Total Day Min

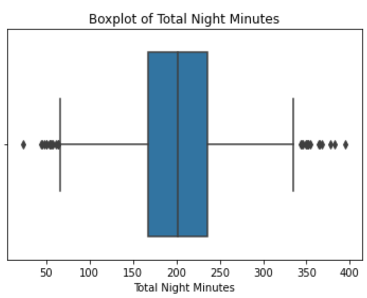

Boxplot of Total Day calls
0 25 50 75 100 125 150
Total Day calls

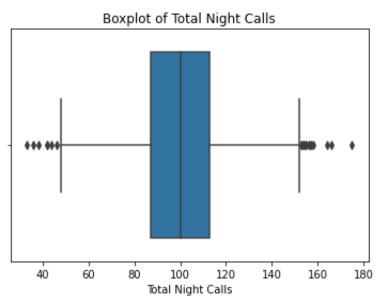

Boxplot of Total Day Charge
0 10 20 30 40 50 60
Total Day Charge

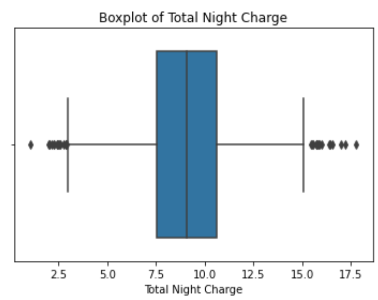

Boxplot of Total Evening Min
0 50 100 150 200 250 300 350
Total Evening Min

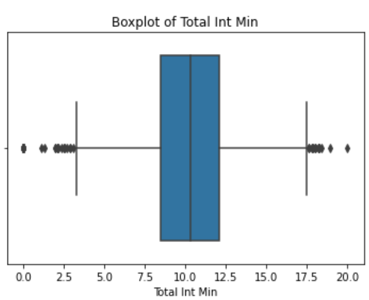

Boxplot of Total Evening Calls
0 25 50 75 100 125 150 175
Total Evening Calls

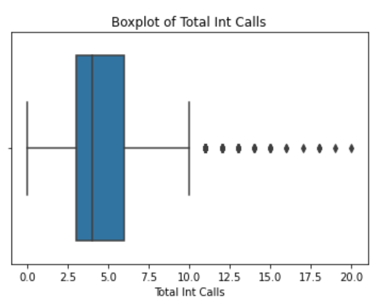

Boxplot of Total Evening Charge
0 5 10 15 20 25 30
Total Evening Charge

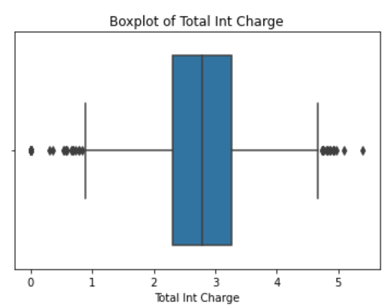

Boxplot of Total Night Minutes
50 100 150 200 250 300 350 400
Total Night Minutes

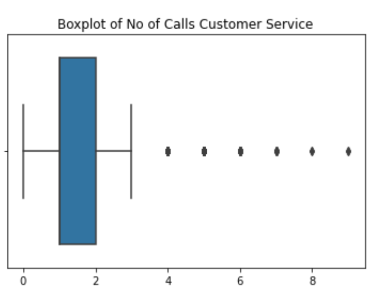

Boxplot of Total Night Calls
40 60 80 100 120 140 160 180
Total Night Calls


Boxplot of Total Night Charge
2.5 5.0 7.5 10.0 12.5 15.0 17.5
Total Night Charge


Boxplot of Total Int Min
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Total Int Min


Boxplot of Total Int Calls
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0
Total Int Calls

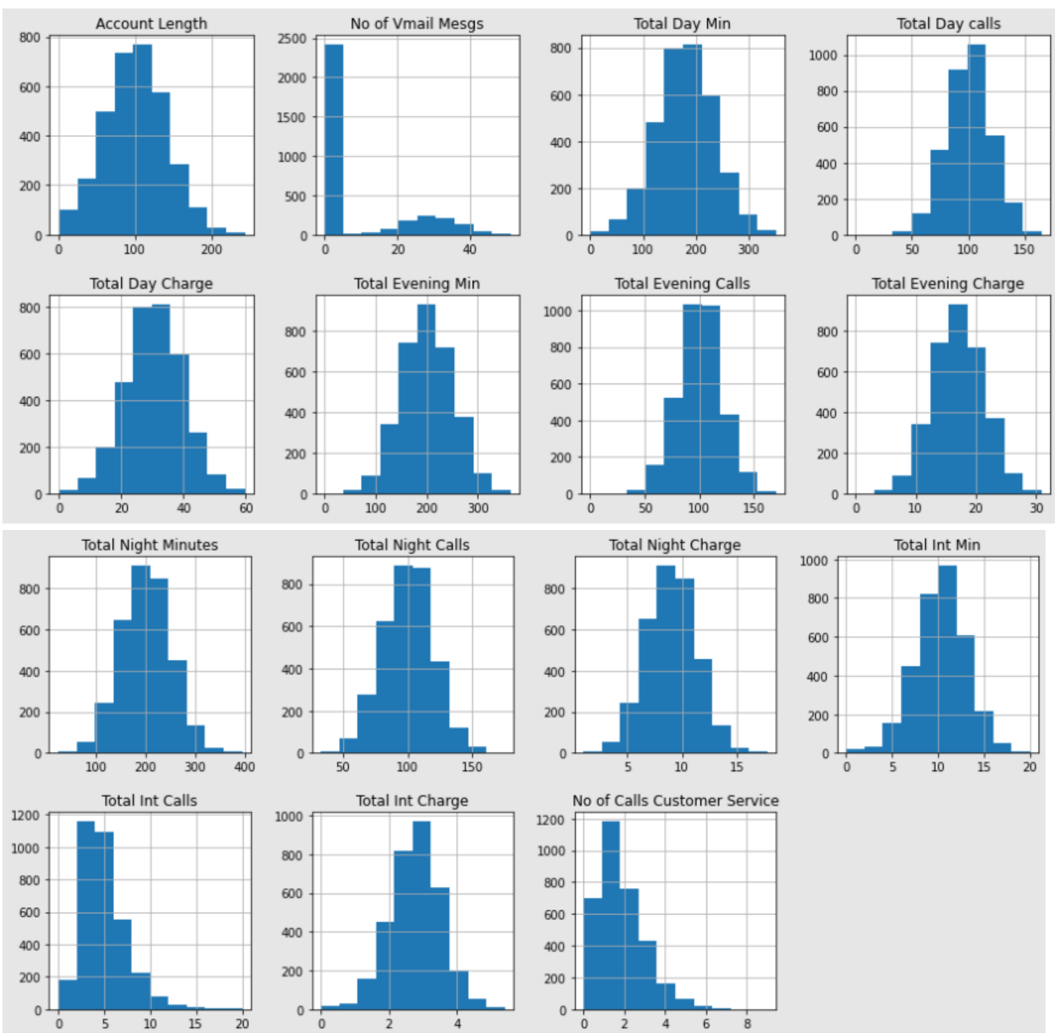

Boxplot of Total Int Charge
0 1 2 3 4 5
Total Int Charge


Boxplot of No of Calls Customer Service
0 2 4 6 8
No of Calls Customer Service

By using a boxplot for every univariate variable, we have spotted some extreme outliers in every numerical column. Outliers can indicate of mere variances in our customer churn data, or it can be a mistake during data collection. Each outlier is an individual point distant from the box and its whiskers (outside of the IQR range). The decision to remove outliers out of the dataset or not, depends on whether their presences are important or not. We can observe from the boxplots that the number of outliers across columns is not significant, so we decide to not drop the outliers from our dataset at this point.

3e. Distribution Visualization of numerical columns & Impacts on Class Attribute (Bonus)

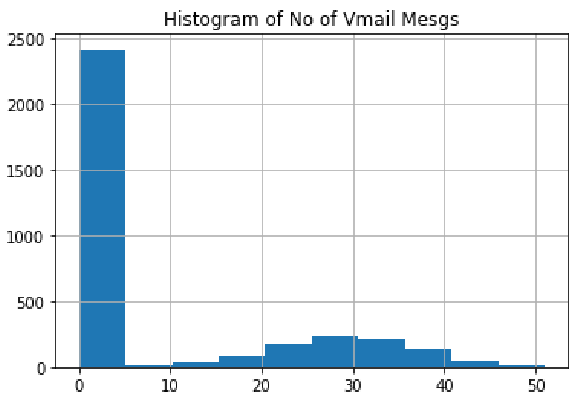
We use histograms to display the distribution of our quantitative/numerical data. Histogram is a powerful tool to "reveal the shape of the distribution of the distribution, its central tendency, and the spread of values" in our dataset (Statistics By Jim, 2022)[3]. Below is an overview of the distribution of all numeric attributes:

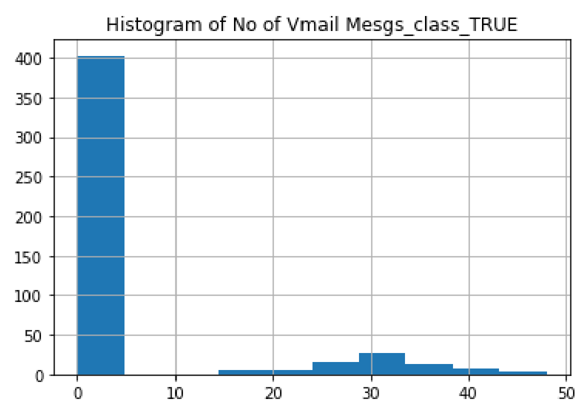


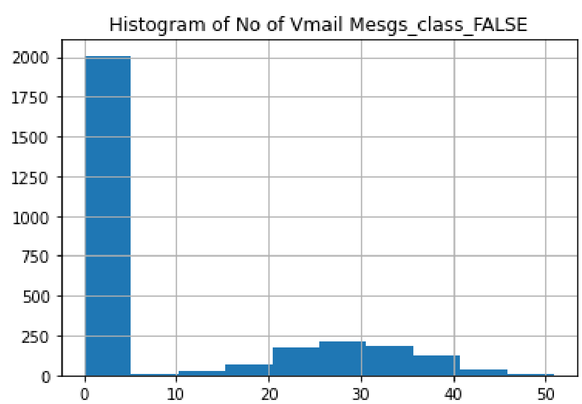
[3]Statistics By Jim, 2022: https://statisticsbyjim.com/basics/histograms/. Date Retrieved : July 31, 2022.

We can see from the illustration above, most of the numeric variables have a pretty much symmetrical, unimodal distribution as a bell curve. The exceptions are notoriously found in: No of Vmail Mesgs, Total Int Calls, No of Calls Customer Service. This finding is in line with the boxplots shown in the previous Outlier Detection section.

The "No of Vmail Mesgs" attribute: A special case where a large percentage of values centers around the approximate [0, 5] range (possibly outliers at the left end) – which makes its boxplot heavily skewed to the right. Let's create histograms for the "No of Vmail Mesgs" attribute, then create the same histograms for this attribute, for the instances of class "TRUE" and for the instances of class "FALSE" to investigate. We use the query() method in Python to slice the data and create the histograms.

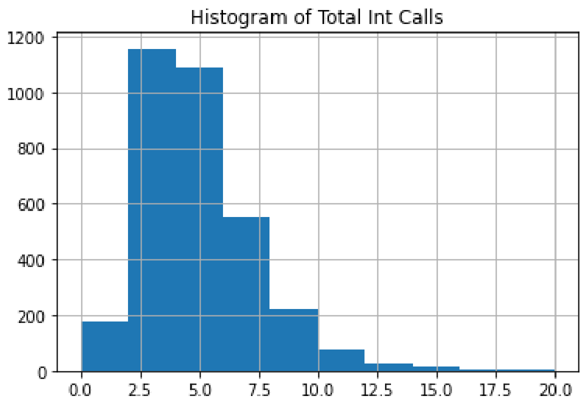


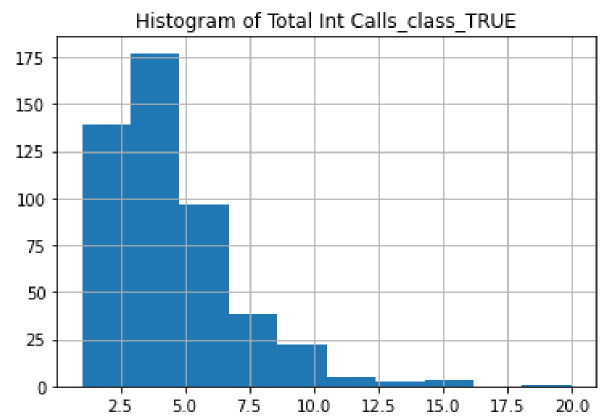


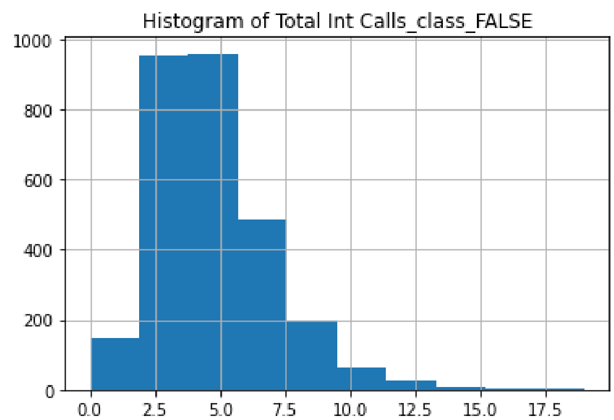


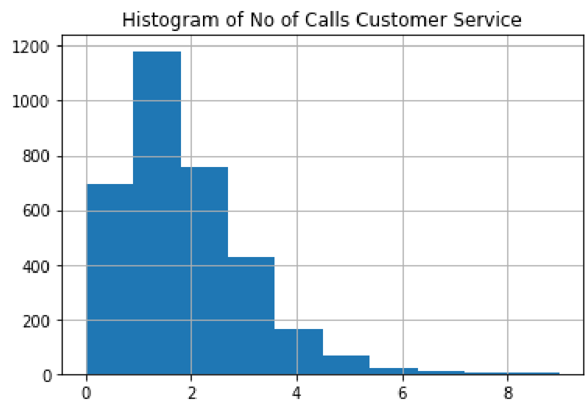
The histogram displays we came up with indicates that "No of Vmail Mesgs" does not impact much on the class attribute. However, we strongly propose further data processing steps for this attribute to find the root cause of the weighty left-end outliers from the range of 0 to 5 (which would not be covered in this analysis due to the fact that it's not feasible removing roughly 2400/3300 rows = 72% of the data we have in the client's dataset). If we can find a way to correct those data, the rest of the distribution seems to follow a normal distribution.

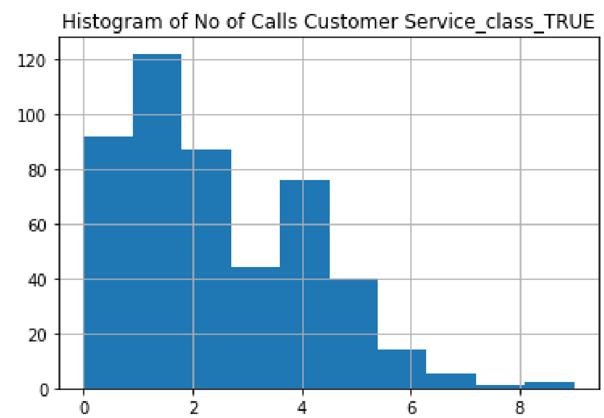
The "Total Int Calls" & "No of Calls Customer Service" attributes: These two attributes also have data values skewed to the right with the long tails displaying the outliers, from our boxplots and histograms. As said, the anomalies size in these cases are not big enough for us to drop them. Let's also study whether these two attributes have any notable influence on the class attribute.

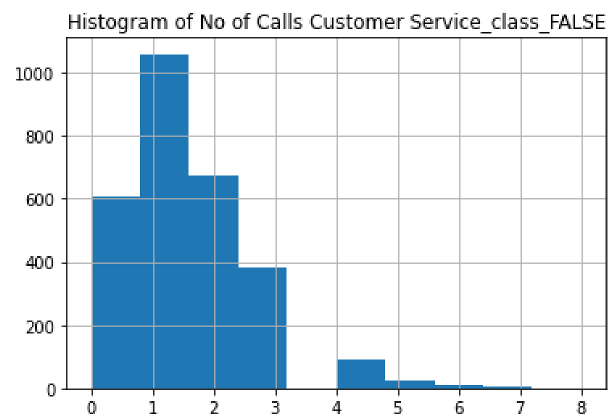








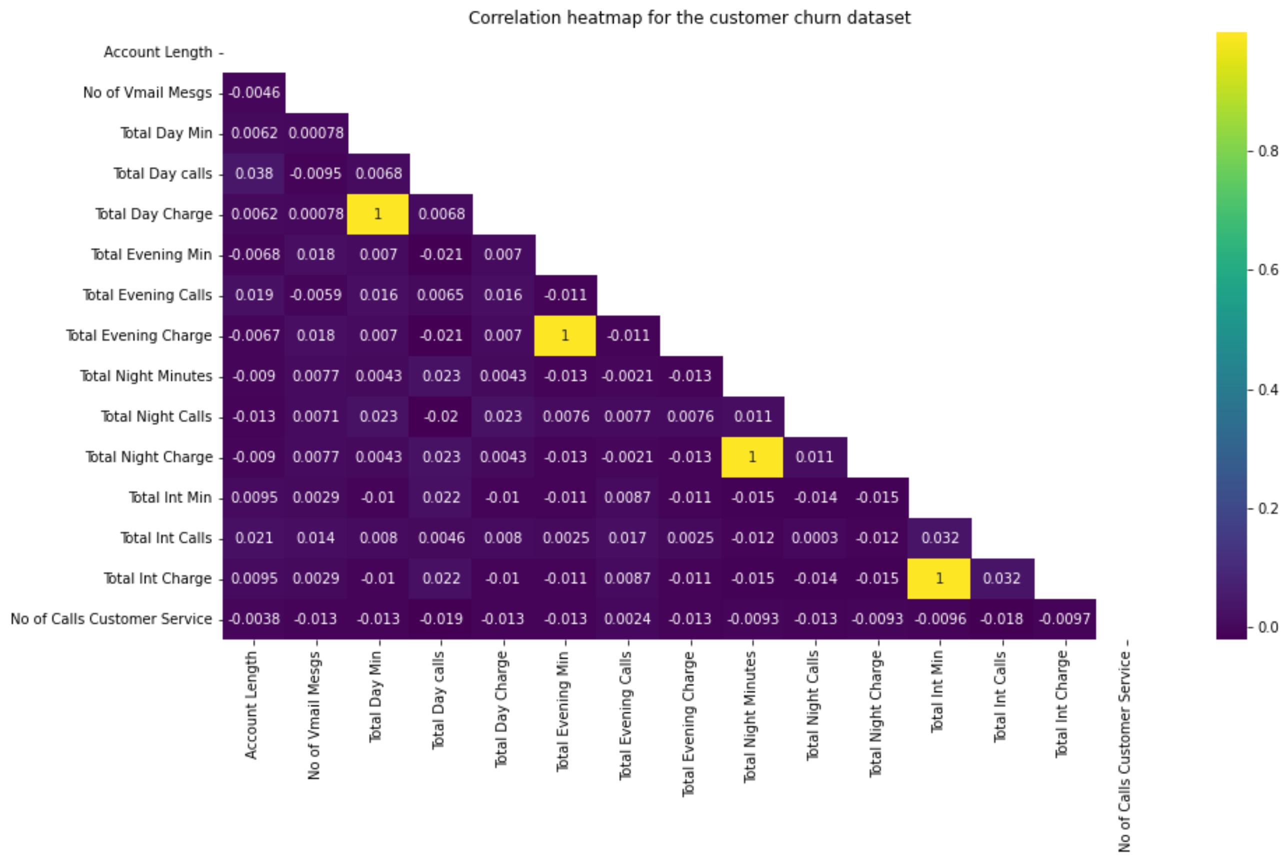




In general, there are proportionate differences among each of the “Total Int Calls” & “No of Calls Customer Service” attributes, plotted with different instances of the “Churn” class attribute (TRUE, FALSE). We found more of a normal distribution for class_FALSE in the “Total Int Calls” attribute, and bimodal distributed, separated samples in the No of Calls Customer Service” attribute; when dividing class attribute. There are much more samples in the class_FALSE dataset for both attributes that can indicate class imbalance (we’ll analyze this point later). We can also see a common pattern that for class_TRUE, more data gather around the left side although the weights of such data are relatively small. Hence, our team has decided to only observe and keep these findings as a reference point. Further data preprocessing iterations are still advised after this initial model building.

3f. Correlation Heatmap (Bonus)

Using the Python’s Seaborn package, we create the correlation heatmap for our customer churn dataset. Only numerical attributes are included in the heatmap, with categorical columns excluded because we haven’t transformed them into numerical values yet. This step will be done in the latter part.



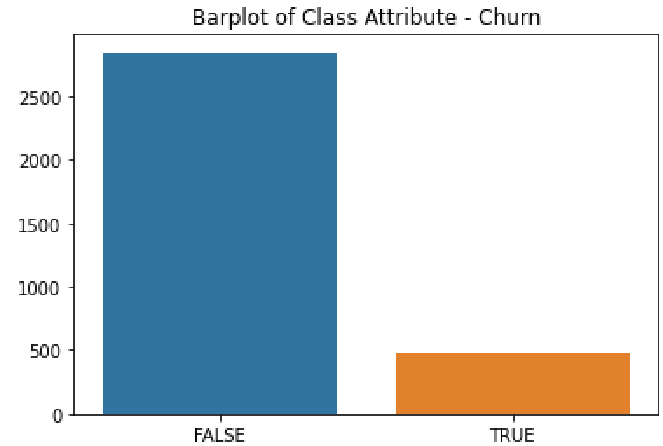
Interestingly, the correlation heatmap shows perfect correlation coefficients of 1 between those attribute pairs: ***Total Day Charge – Total Day Min, Total Evening Charge – Total Evening Min, Total Night Charge – Total Night Minutes, Total Intl Charge - Total Int Min***.

It is usually recommended to avoid having correlated features in our dataset, because we'll gain little information yet increase the complexity and the risk of errors in our algorithm. Indeed, a group of highly correlated features will not bring additional information (or just very few), but will increase the complexity of the algorithm, thus increasing the risk of errors (stackoverflow, 2022)[4]. Thus, we'll exclude those pairs of features from our selected features while re-train the baseline model (with all features/attributes) classification algorithm on our selected features on the validation set (or test set).

3g. Class Imbalance in "Churn" – churned (TRUE) or not churned (FALSE) (Bonus)

Our goal is to predict the churned customers properly, we can test the class imbalance in our target attribute "Churn" by examining how many rows are available for each class in the data.

[4] stackoverflow, 2022: https://stackoverflow.com/questions/65302136/what-we-should-do-with-highly-correlated-features#:~:text=In%20general%2C%20it%20is%20recommended,increasing%20the%20risk%20of%20errors. Date Retrieved : July 31, 2022.




```
Percentage of churned customer: 0.14491449144914492
Percentage of not-churned customer: 0.8550855085508551
```

Calculating the current churn-rate:

```
Churn Rate: 0.14491449144914492
```

The output shows only 15% of data are related to the churned customers and 85% of data are related to non-churned customers. With such a big difference, we will need to oversample the minority class of “TRUE” by using SMOTE (aka “Synthetic Minority Over-sampling Technique”) later on. We would want to create synthetic data using the characteristics of the nearest neighbours using SMOTE (via the imblearn python library). However, we would only use this technique in the training data, using the “train_test_split” splitting strategy from scikit-learn.

3h. Categorical to Numeric Encoding for categorical columns (Bonus)

First, we will drop the “Phone Number” column before numerical encoding, since it’s a text column with many unique values that will get complicated in encoding into numerical data. We will not gain much information from the phone numbers as well with all unique numbers.

After that, we will use the One hot Encoding technique, converting only categorical features to dummy/one-hot features to treat the following categorical attributes**: "State", "Area Code", "Inter Plan", "VoiceMail Plan".** One hot encoding is meant to create additional attributes based on the unique values of the existing categorical variables – features with the “object” data type in our dataset. The Pandas’ get_dummies() method is used in this case to generate binary values (0,1) for each unique values of our one-hot categorical features.

By the end of this step, our new encoded dataframe will look like this before running the classification algorithms.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
```

```
Data columns (total 74 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Account Length                 3333 non-null   float64
 1   No of Vmail Mesgs              3333 non-null   float64
 2   Total Day Min                  3333 non-null   float64
 3   Total Day calls                3333 non-null   float64
 4   Total Day Charge               3333 non-null   float64
 5   Total Evening Min              3333 non-null   float64
 6   Total Evening Calls            3333 non-null   float64
 7   Total Evening Charge           3333 non-null   float64
 8   Total Night Minutes            3333 non-null   float64
 9   Total Night Calls              3333 non-null   float64
 10  Total Night Charge             3333 non-null   float64
 11  Total Int Min                  3333 non-null   float64
 12  Total Int Calls                3333 non-null   float64
 13  Total Int Charge               3333 non-null   float64
 14  No of Calls Customer Service   3333 non-null   float64
 15  Churn                          3333 non-null   object 
 16  State_AK                       3333 non-null   uint8  
 17  State_AL                       3333 non-null   uint8  
 18  State_AR                       3333 non-null   uint8  
 19  State_AZ                       3333 non-null   uint8  
 20  State_CA                       3333 non-null   uint8  
 21  State_CO                       3333 non-null   uint8  
 22  State_CT                       3333 non-null   uint8  
 23  State_DC                       3333 non-null   uint8  
 24  State_DE                       3333 non-null   uint8  
 25  State_FL                       3333 non-null   uint8  
 26  State_GA                       3333 non-null   uint8  
 27  State_HI                       3333 non-null   uint8  
 28  State_IA                       3333 non-null   uint8  
 29  State_ID                       3333 non-null   uint8  
 30  State_IL                       3333 non-null   uint8  
 31  State_IN                       3333 non-null   uint8  
 32  State_KS                       3333 non-null   uint8  
 33  State_KY                       3333 non-null   uint8  
 34  State_LA                       3333 non-null   uint8  
 35  State_MA                       3333 non-null   uint8  
 36  State_MD                       3333 non-null   uint8  
 37  State_ME                       3333 non-null   uint8  
 38  State_MI                       3333 non-null   uint8  
 39  State_MN                       3333 non-null   uint8  
 40  State_MO                       3333 non-null   uint8  
 41  State_MS                       3333 non-null   uint8  
 42  State_MT                       3333 non-null   uint8  
 43  State_NC                       3333 non-null   uint8  
 44  State_ND                       3333 non-null   uint8  
 45  State_NE                       3333 non-null   uint8  
 46  State_NH                       3333 non-null   uint8  
 47  State_NJ                       3333 non-null   uint8  
 48  State_NM                       3333 non-null   uint8  
 49  State_NV                       3333 non-null   uint8  
 50  State_NY                       3333 non-null   uint8  
 51  State_OH                       3333 non-null   uint8  
 52  State_OK                       3333 non-null   uint8  
```

```
 53   State_OR                        3333 non-null   uint8
 54   State_PA                        3333 non-null   uint8
 55   State_RI                        3333 non-null   uint8
 56   State_SC                        3333 non-null   uint8
 57   State_SD                        3333 non-null   uint8
 58   State_TN                        3333 non-null   uint8
 59   State_TX                        3333 non-null   uint8
 60   State_UT                        3333 non-null   uint8
 61   State_VA                        3333 non-null   uint8
 62   State_VT                        3333 non-null   uint8
 63   State_WA                        3333 non-null   uint8
 64   State_WI                        3333 non-null   uint8
 65   State_WV                        3333 non-null   uint8
 66   State_WY                        3333 non-null   uint8
 67   Area Code_A408                  3333 non-null   uint8
 68   Area Code_A415                  3333 non-null   uint8
 69   Area Code_A510                  3333 non-null   uint8
 70   Inter Plan_no                   3333 non-null   uint8
 71   Inter Plan_yes                  3333 non-null   uint8
 72   VoiceMail Plan_no               3333 non-null   uint8
 73   VoiceMail Plan_yes              3333 non-null   uint8
dtypes: float64(15), int64(1), uint8(58)
memory usage: 605.5 KB
```

As mentioned in the Class Imbalance section, we would have to apply SMOTE in the imblearn python library, on the training dataset, to balance our class attribute.

The result of resampling the data and verifying the data:

```
Resampled dataset shape Counter({'FALSE': 2000, 'TRUE': 2000})
```

IV. Predictive Modeling (Classification)

Machine learning is the core of data science to help business stakeholders make predictions about the future. This applies to predicting the customer churn in our client's dataset. "Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data." (Machine Learning Mastery, 2022).[5]

One of the most popular types of classification is the "binary classification" – where the classification tasks have two class labels. Customer churn is a typical example of a binary classification with two states of the predicted results: TRUE and FALSE. The class for the normal state will be assigned the class label "TRUE", whereas the class with the abnormal state will be assigned the class label "FALSE".

[5] Machine Learning Mastery, 2022: https://machinelearningmastery.com/types-of-classification-in-machine-learning/#:~:text=In%20machine%20learning%2C%20classification%20refers,one%20of%20the%20known%20characters. Retrieved : July 31, 2022.

4a. Data Splitting Strategy

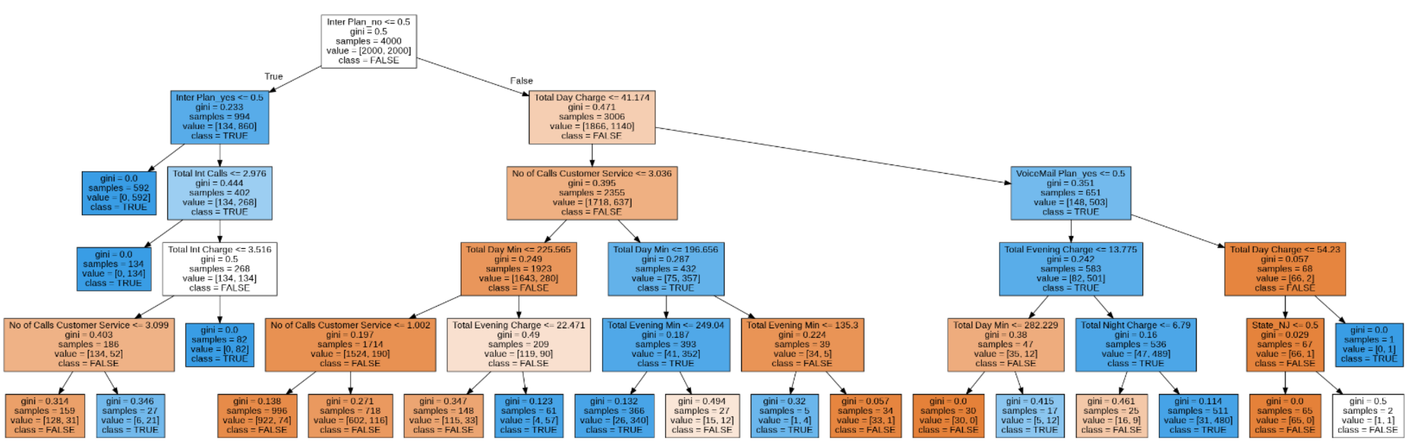
How to split the existing data is one of the important decisions our data scientists' team have to make prior to data modeling. We've decided to use a common technique, that is to split the data into two groups of the training and testing (validation) sets to align with the classification modeling. "Classification requires a training dataset with many examples of inputs and outputs from which to learn" (Machine Learning Mastery, 2022).[6] The chose data split strategy is the 70% training and 30% testing spit method.

4b. Classification using Decision Tree (Supervised Learning)

We're building a classification decision tree model to predict whether telecom customers would churn (binary outcomes: TRUE – for yes and FALSE – for no) based on the dataset. The output diagram of the decision tree will provide more detail about the nodes and splits in a capped five-level tree. The default grow algorithm here is gini.

The Decision Tree baseline model:

When our team run the decision tree with all features, balanced class labels on "Churn" using the training set (max depth of 5) in Python; below is a snapshot of the baseline decision tree model.



The output baseline decision tree utilizes attributes such as***: Inter Plan, Total Int Calls, Total Int Charge, No of Calls Customer Service, Total Day Charge, Total Day Min, Total Evening Charge, Total Evening Min, VoiceMail Plan, Total Night Charge, State_NJ*** in making the prediction of which customers' characteristics group would churn. We also observe that some highly correlated feature pairs discussed in the Correlation Heatmap, shows up in the baseline decision tree. For instance: ***Total Day Charge, Total Day Min.***

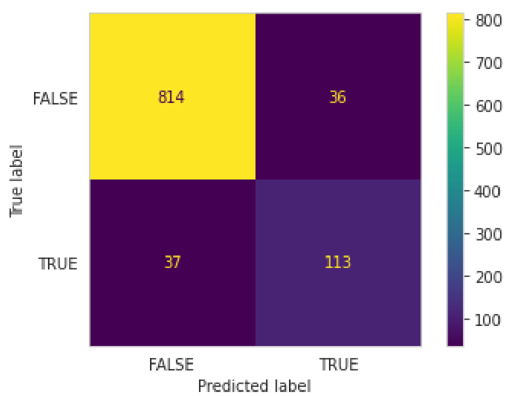
[6] Machine Learning Mastery, 2022: https://machinelearningmastery.com/types-of-classification-in-machine-learning/#:~:text=In%20machine%20learning%2C%20classification%20refers,one%20of%20the%20known%20characters. Retrieved : July 31, 2022.

In the resulted baseline decision tree, we have samples=4000 observations at the root node (as a result of class balancing in the training dataset), with each decision node is labeled with the corresponding independent variable name and split value. At the leaf nodes, the diagram shows the classification class decision after five splits.

From the test dataset, we also have to check if the baseline decision tree model works well in predicting the churned customer, using accuracy, precision and recall as evaluation metrics. The accuracy score, the classification report in the , along with the can show us whether the model did a great job at predicting the customer as churned.

```
The Baseline Classification Tree Model accuracy score is: 0.9270

              precision    recall  f1-score   support

       FALSE       0.96      0.96      0.96       850
        TRUE       0.76      0.75      0.76       150

    accuracy                           0.93      1000
   macro avg       0.86      0.86      0.86      1000
weighted avg       0.93      0.93      0.93      1000
```

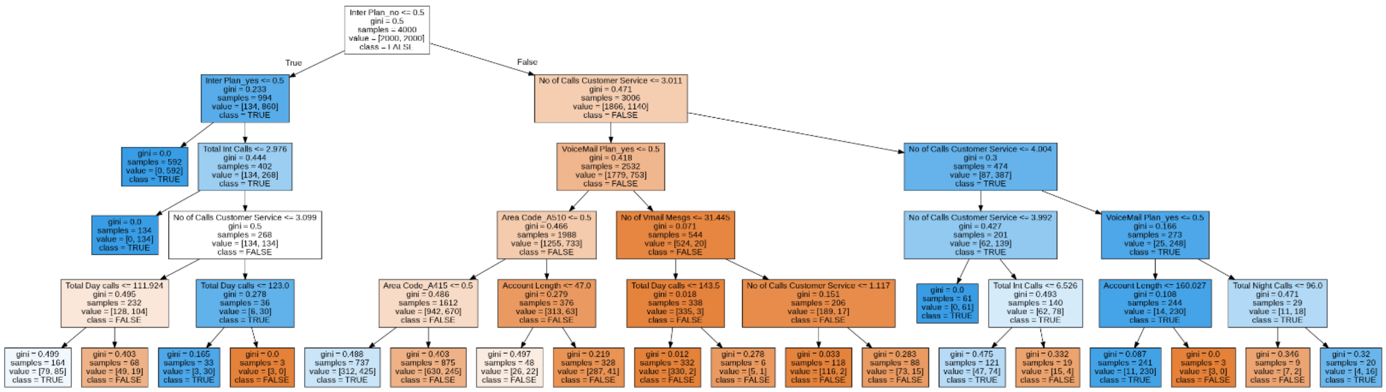
The confusion matrix for the baseline decision tree model:



The overall accuracy of this baseline decision tree is 92.7% ~ 93% which is great. Precision and recall are not much different, respectively at 76% and 75%. From 1,000 churned customer test samples (balanced class), we are detecting 113 samples correctly and 36 are misclassified.

The Decision Tree model with selected features:

We'll train the same training set with only selected features (removing highly correlated features) using the training set (max depth of 5) in Python; below is a snapshot of the baseline decision tree model.

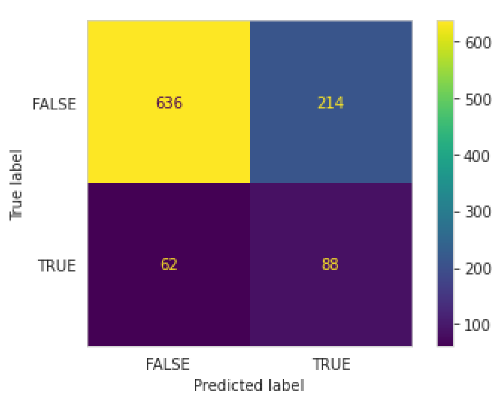


Attributes such as***: Inter Plan, Total Int Calls, No of Calls Customer Service, Total Day calls, No of Calls Customer Service, VoiceMail Plan, Area Code_A510, Area Code_A415, Account Length, No of VMail Mesgs, Total Night Calls*** are used in the predictive classification tree model with only the selected features. The highly correlated feature pairs discussed in the Correlation Heatmap section, for example: ***Total Day Charge - Total Day Min*** are completely removed from the model. The leaf node at ***Area Code_A510 <= 0.5*** then splits into ***Area Code_A415 <= 0.5*** and ***Account Length <= 47.0*** does not really make sense. This model seems not be very accurate.

Let’s check the evaluation metrics for this decision tree classification model:

```
The Classification Tree Model with selected features accuracy score is:
0.7240
              precision    recall  f1-score   support

       FALSE       0.91      0.75      0.82       850
        TRUE       0.29      0.59      0.39       150

    accuracy                           0.72      1000
   macro avg       0.60      0.67      0.61      1000
weighted avg       0.82      0.72      0.76      1000
```

The confusion matrix plot for the decision tree model with selected features:



Accuracy of this decision tree with selected features is 72.4%; precision and recall are really low at 29% and 59%. From 1,000 churned customer test samples (balanced class), we are detecting only 88 samples correctly and 214 are misclassified.

Compare the two Decision Tree prediction models – baseline vs selected features:

To compare the performances of the two decision-tree classifiers, we measure by the accuracy, precision, and recall metrics to determine which decision tree model is more accurate:

	Baseline Decision Tree Model	Selected Features Decision Tree Model
	Using all features	*Using only selected features*
Accuracy	92.70%	72.40%
Precision	0.76	0.29
Recall	0.75	0.59

From the performance table above compared between the decision tree algorithm using all attributes and the decision tree algorithm using selected attributes, we consider the decision classifier with all attributes more accurate with better performance metrics.

4c. Classification using Naïve Bayes (Unsupervised Learning)

The Naive Bayes classification algorithm is based upon Bayes' Theorem. Basically, Naïve Bayes classifiers measures the conditional probabilities of each class by using their counts/frequencies in each record, and predict the class with the highest probability.

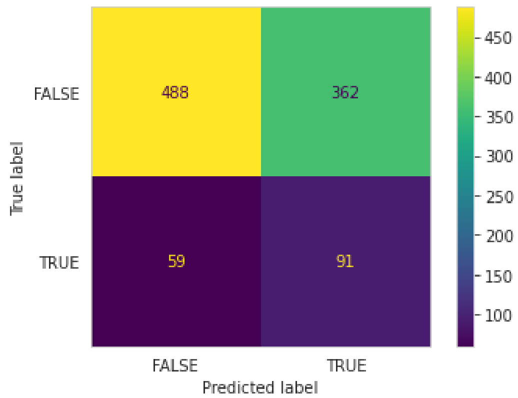
We are using a multinomial naive Bayes – MultinomialNB in Python - "where the features are assumed to be generated from a simple multinomial distribution. The multinomial distribution describes the probability of observing counts among a number of categories" (Python Data Science Handbook, 2022)[7]. Each row represents one record; each column represents one attribute.

The Naïve Bayes baseline model with all features evaluation metrics:

```
The Baseline Naive Bayes Model accuracy score is: 0.5790

              precision    recall  f1-score   support

       FALSE       0.89      0.57      0.70       850
        TRUE       0.20      0.61      0.30       150

    accuracy                           0.58      1000
   macro avg       0.55      0.59      0.50      1000
weighted avg       0.79      0.58      0.64      1000
```

[7] Python Data Science Handbook, 2022: https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bayes.html. Retrieved : July 31, 2022.

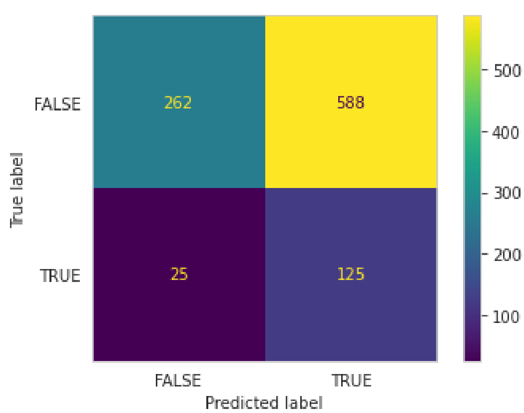
The confusion matrix plot:



The Naïve Bayes baseline model with selected features evaluation metrics:

```
The Naive Bayes Model with selected features accuracy score is: 0.3870

              precision    recall  f1-score   support

       FALSE       0.91      0.31      0.46       850
        TRUE       0.18      0.83      0.29       150

    accuracy                           0.39      1000
   macro avg       0.54      0.57      0.38      1000
weighted avg       0.80      0.39      0.44      1000
```

The confusion matrix plot :



Compare the two Naïve Bayes prediction models – baseline vs selected features:

We create a performance comparison table to compare the two Naïve Bayes classifiers, via the evaluation measures of precision, and recall metrics to determine which decision tree model is more accurate:

	Baseline Naïve Bayes Model	Selected Features Naïve Bayes Model
	Using all features	*Using only selected features*
Accuracy	57.90%	38.70%
Precision	0.20	0.18
Recall	0.61	0.83

From the performance table above, we can conclude that the Naïve Bayes predictive model with all attributes is more accurate with better performance metrics than the model with only selected features.

4c. Compare the two classification techniques - Decision Tree vs Naïve Bayes

	Baseline Decision Tree Model	Selected Features Decision Tree Model	Baseline Naïve Bayes Model	Selected Features Naïve Bayes Model
	Using all features	*Using only selected features*	*Using all features*	*Using only selected features*
Accuracy	92.70%	72.40%	57.90%	38.70%
Precision	0.76	0.29	0.20	0.18
Recall	0.75	0.59	0.61	0.83

Overall, decision tree models predict the class attribute in our customer churn dataset better than Naïve Bayes models. The two techniques have significantly different evaluation metrics. The decision tree classification algorithms produce better performing classifiers with greater accuracies, greater precisions. But when it comes down to recalls, it's really a hit or miss in our data models; as Naïve Bayes classifiers can either produce a lower or higher recall rate – depending on the feature selection chosen.

V. Conclusions and Recommendations

By far, the best predictive result for customer churn characterization and prediction comes from utilizing all the attributes in the dataset to build a decision tree classification model with an accuracy score of 92.7%. a precision rate of 0.76.

The decreasing performances going from using all features down to several selected features reflect a need to further investigate and select a better combination of selected features. This implies that choosing the suitable features is very important if we would like to enhance the outcome of this predictive analytics project. Carrying out iterative cycles of data-preprocessing and feature selection process may indeed improve our models' performances. Another separate step of "Feature Evaluation" can be considered in which we rank which features play the most important role in the identification of customer churn. Probably, we can try applying RandomForestClassifier's "feature_importance" to rank the most important features for a given classification.

Moreover, we also recommend exploring other predictive techniques such as K-Means in case they may yield more promising results for customer churn prediction.