Churn Reduction

Akash Kumar Dubey

1Sept 2018

Contents

Chapter 1

Introduction

1.1 Problem Description

Churn prediction is knowing which users are going to stop using your platform or service in the future.These predictions are used by marketers to proactively take retention actions on Churning users. For example, If a company knew a segment of users who were at risk of Churning on the platform or service then that company can proactively engage these users with Offers etc. and help them make a purchase. In order to achieve the aggressive goals companies set, growth cannot based only on acquiring new clients. A very important and crucial part of the growth is coming from the existing clients. Retaining and expanding revenues from existing clients is a must to achieve hyper positive growth.Acquiring new clients is 5-10 times (and sometimes even 20 times) more expensive than retaining and expanding existing clients. Companies have understood this, and therefore are trying to predict who are those customers at risk and what can be done to retain them.

1.2 Problem Statement

The objective of this Case study is to predict customer behaviour. We are provided with a public dataset that has customer usage pattern and if the customer has moved or not. We are expected to develop an algorithm to predict the churn score based on usage pattern.

1.3 Data

Our task is to build classification models which will classify the customers as ‘Churn’ and ‘Not Churn’ depending upon the different customers usage pattern.

Given below is a sample of the data set that we are using to predict the churn score based on usage pattern :

Table 1.1 : Churn Prediction Customer Data (Columns : 1-8)

state	account length	area code	internati onal plan	voice mail plan	number vmail message	total day minutes	total day calls
KS	128	415	no	yes	25	265.1	110
OH	107	415	no	yes	26	61.6	123
NJ	137	415	no	no	0	243.4	114
OH	84	408	yes	no	0	299.4	71
OK	75	415	yes	no	0	166.7	113

Table 1.2 : Churn Prediction Customer Data (Columns : 9-15)

total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge
45.07	197.4	99	16.78	244.7	91	11.01
27.47	195.5	103	16.62	254.4	103	11.45
41.38	121.2	110	10.30	162.6	104	7.32
50.90	61.9	88	5.26	196.9	89	8.86
28.34	148.3	122	12.61	186.9	121	8.41

Table 1.3 : Churn Prediction Customer Data (Columns : 16-21)

total intl minutes	total intl minutes	total intl calls	total intl charges	number customer service calls	Churn
10.0	10.0	3	2.70	1	False.
13.7	13.7	3	3.70	1	False.
12.2	12.2	5	3.29	0	False.
6.6	6.6	7	1.78	2	False.
10.1	10.1	3	2.73	3	False.

As we can see in the table above, we have we have the following 20 variables, using which we have to correctly predict whether the customer would Churn or not.

Table 1.4 : Predictor Variables

S.No	Predictor	S.No	Predictor
1	state	11	total eve minutes
2	account length	12	total eve calls
3	area code	13	total eve charges
4	phone number	14	total night minutes
5	international plan	15	total night calls
6	voice mail plan	16	total night charges
7	number vmail messages	17	total intl minutes
8	total day minutes	18	total intl calls
9	total day calls	19	total intl charges
10	total day charges	20	number customer service calls

1.4 Performance Metric

As the data is imbalance, accuracy cannot be our main measure of the performance as it will always favour the majority class and will give a good accuracy even if our model is dumb. Similarly ROC AUC score also cannot be our main measure on similar grounds. So, Precision and Recall seems to be good fit for the performance measure. Although the main measure of the performance of the model would be Recall but we will approach it as a trade off between Precision and Recall. We will also keep an eye on fscore.

Chapter 2

Methodology

2.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the first step in our data analysis process. We do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in our existing data, using visual and quantitative methods to get a sense of the story this tells. To start with this process, we will first have a look at univariate analysis like plotting Box plot and whiskers for individual features and Violin plots for the same. Then we will proceed to Multivariate analysis like Box-plot and whiskers and violin plots for the features with respect to the target variable. Also, we will have a look at the Principal Component Analysis (PCA) and t- distributed Stochastic Neighbor Embedding (t-SNE) visualisation of the data.

2.1.1 Data Visualisation

Data visualisation helps us to get better insights of the data. By visualising data, we can identify areas that need attention or improvement and also clarifies which factors influence customer behaviour and how the resources are used by the customers.

2.1.1.1 Univariate Analysis

Univariate analysis is the simplest form of data analysis where the data being analysed contains only one variable. Since it's a single variable it doesn't deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.

So, Lets have a look at the Box Plot and Whiskers for each numeric variables.

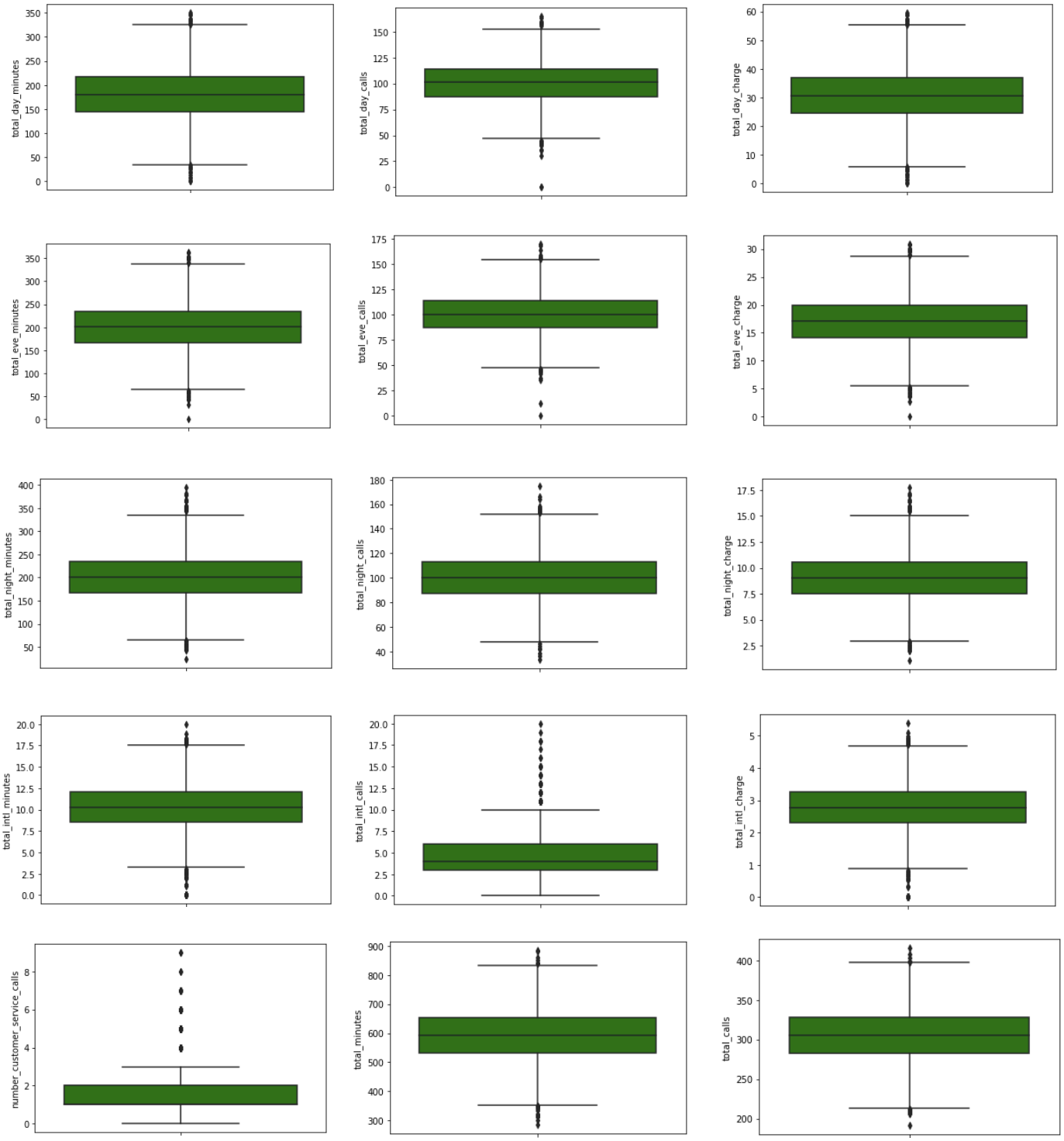


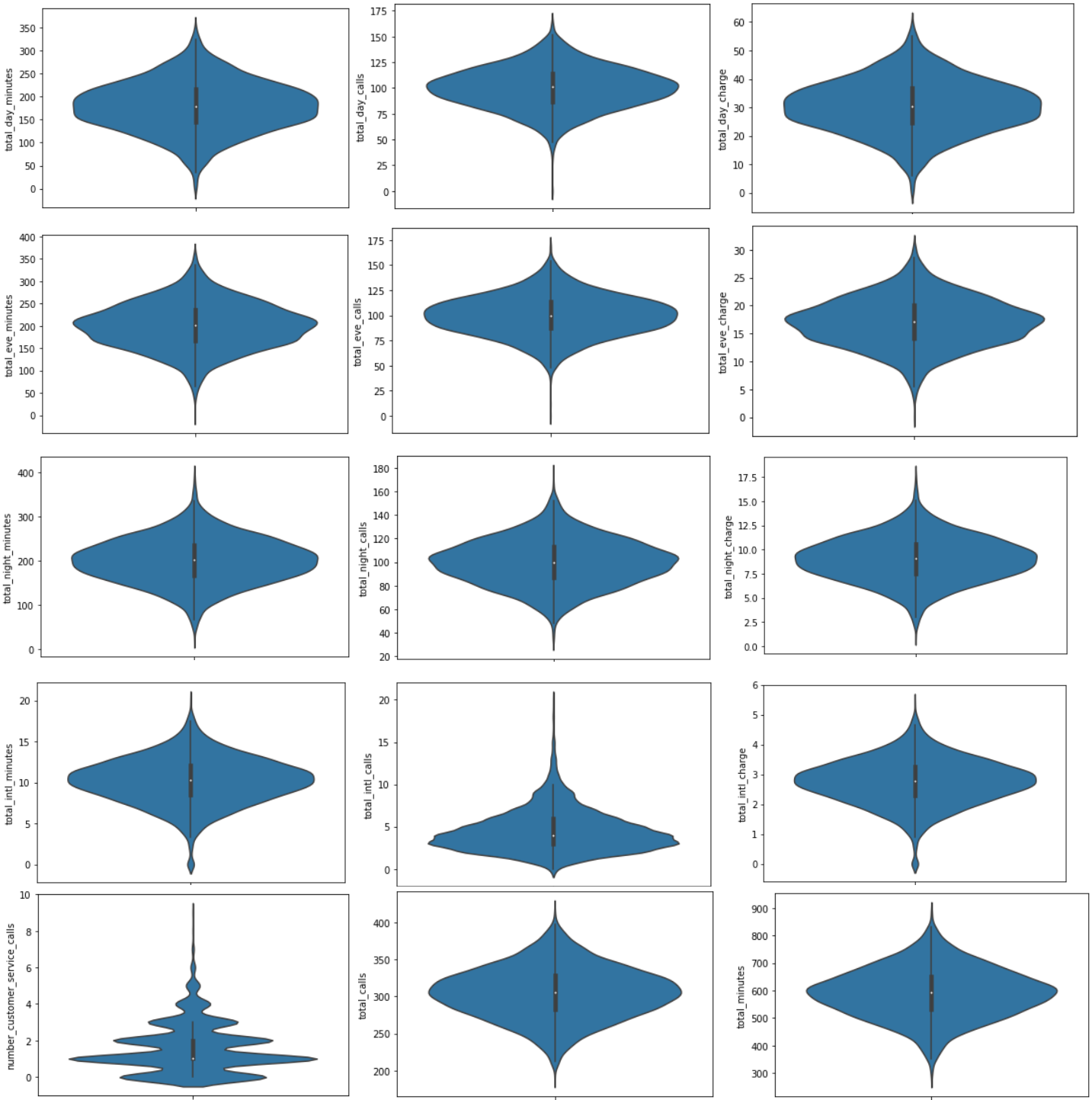
Figure 2.1 : Box plots of Customer Behaviour

So, we can clearly observe from these box-plots that, not every feature contains outliers or they either have very few outliers. It can also be observed that almost all the features are symmetrically distributed except for a few like *total intl calls* because the upper fence and the lower fence are equidistant from the 50th percentile mark.

Also, given the constraint that, we have only 3333 data-points and after removing the outliers, the data gets decreased by almost 15%. So, we will not drop the outliers. Instead we will choose algorithms that are not very much sensitive to the outliers for the modeling.

Now, Lets have a look at the Violin Plots for all the numerical features.*Violin Plots* is one of the best ways to visualise data as it combines the power of both histogram, Probability density function plots and Box plots.

Figure 2.2 Violin Plots of Customer Behaviour



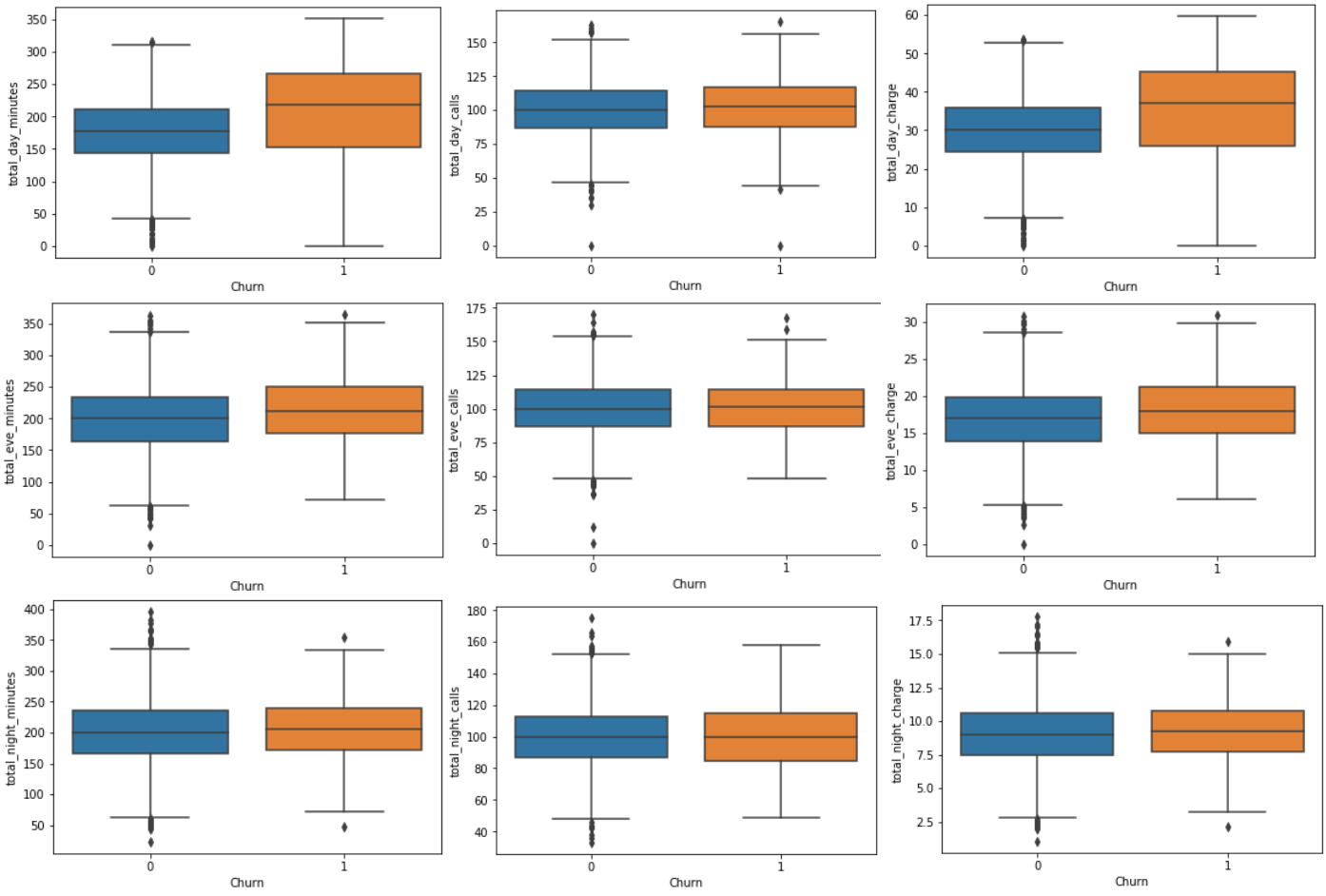
From the very first look at the violin plots , it seems like almost every feature follows gaussian distribution except a few like *number_customer _service_calls* and *total_intl_calls* that looks jaggered but after looking at the Violin Plots closely, we can infer that most features like *total_charge*, *total_calls*, *total_minutes*, t*otal_night_charge*, *total_night_minute, total_day_calls* and *total_day_minutes* seem to follow gaussian distribution. So, we can use all the properties of *central limit theorem* for the features that follow gaussian distribution.

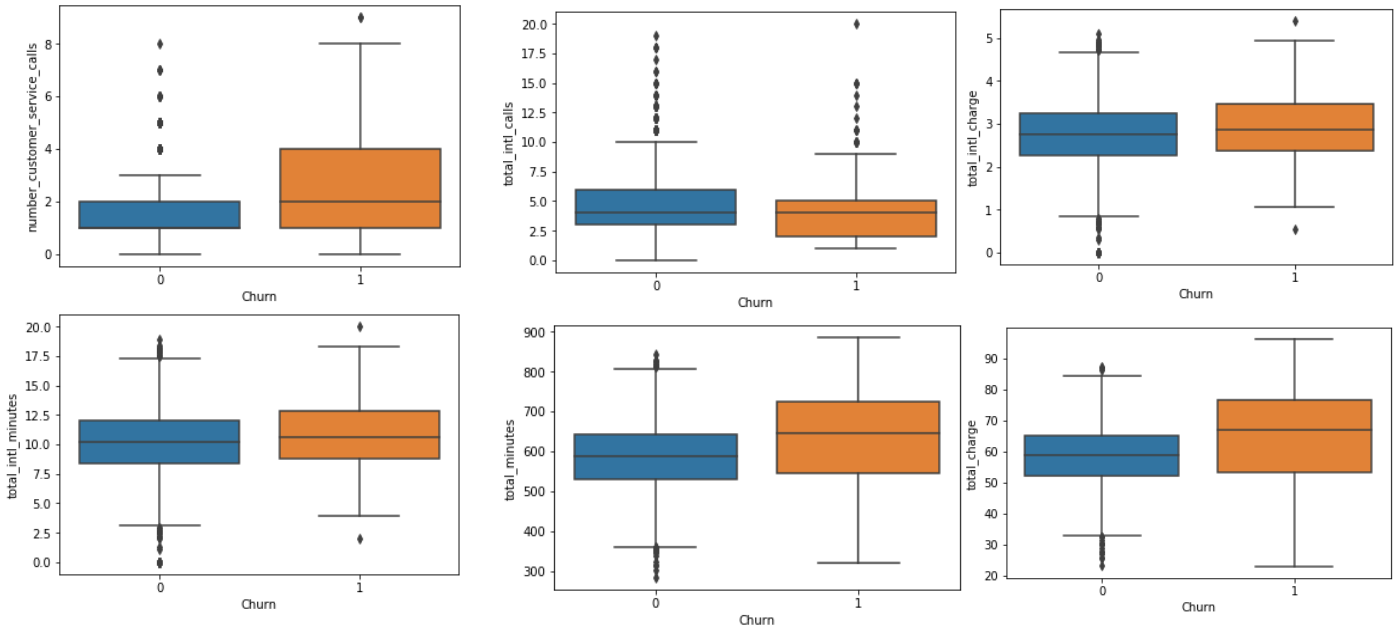
2.1.1.1 Bivariate Analysis

Bivariate analysis refers to the analysis of bivariate data. It is one of the simplest forms of statistical analysis, used to find out if there is a relationship between two sets of values. It usually involves one predictor variable and one target variable.

So, Lets have a look at box plot and whiskers, but this time with respect to the target variable.

Figure 2.3 Box Plots of Customer Behaviour against the target variable



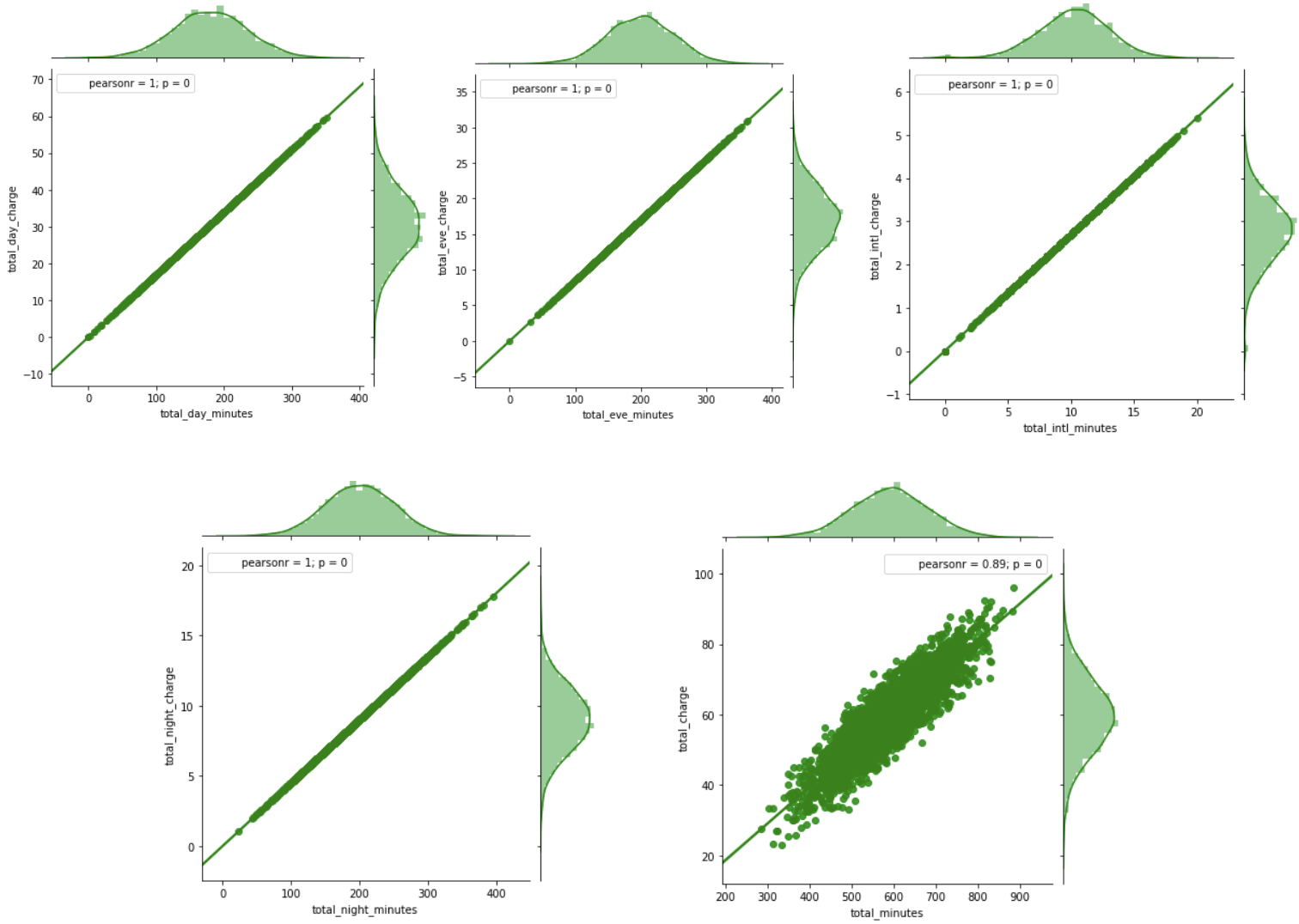


From the above plots, We can clearly observe that Box plot for almost every features based on the label '0' that indicates that the customer will not churn is symmetric and the features might follow symmetric distribution based on their label. Also features like *total_day_minutes, total_day_charge, number_customer_service_calls, total_minutes,* and *total_charge* do not look symmetric and might not follow symmetric distribution for positive label 1. Also, from the very first look at the box plot of *number_customer_service calls* , it can be interpreted that the customer with label '1' (i.e Churn)made more number of customer service calls then the customers with label '0' ('Not Churn') and follow entirely different distributions. So, this variable might be a very good predictor of the target variable.

Also, Violin plots for the variables plotted against the target variables gives a little information about the data and are plotted in appendix. From these plots, it can be inferred that features like *total_calls, total_intl_calls, total_eve_calls, total_eve_minutes, total_eve_charge ,total_day_calls* follow similar distribution based on their respective labels.

Now, lets have a look at the joint-plot for few of the numeric variables.

Figure 2.4 Joint Plots of Customer Behaviour



From the Joint-plot visualisation of features, we can easily observe that features like *total_day_charge : total_day_minutes, total_eve_charge : total_eve_minutes , total_night_charge : total_night_minutes* follow a linear relationship. This visualisation also gives us an idea that these features may be correlated as they follow a linear relationship. Features *total_minutes : total_charge* also follow linear relationship but not like other features mentioned above and the data points do not fall on the regression line and are linearly scattered along the regression line. So, *total_charge : total_minutes* are less correlated then the above predictors.

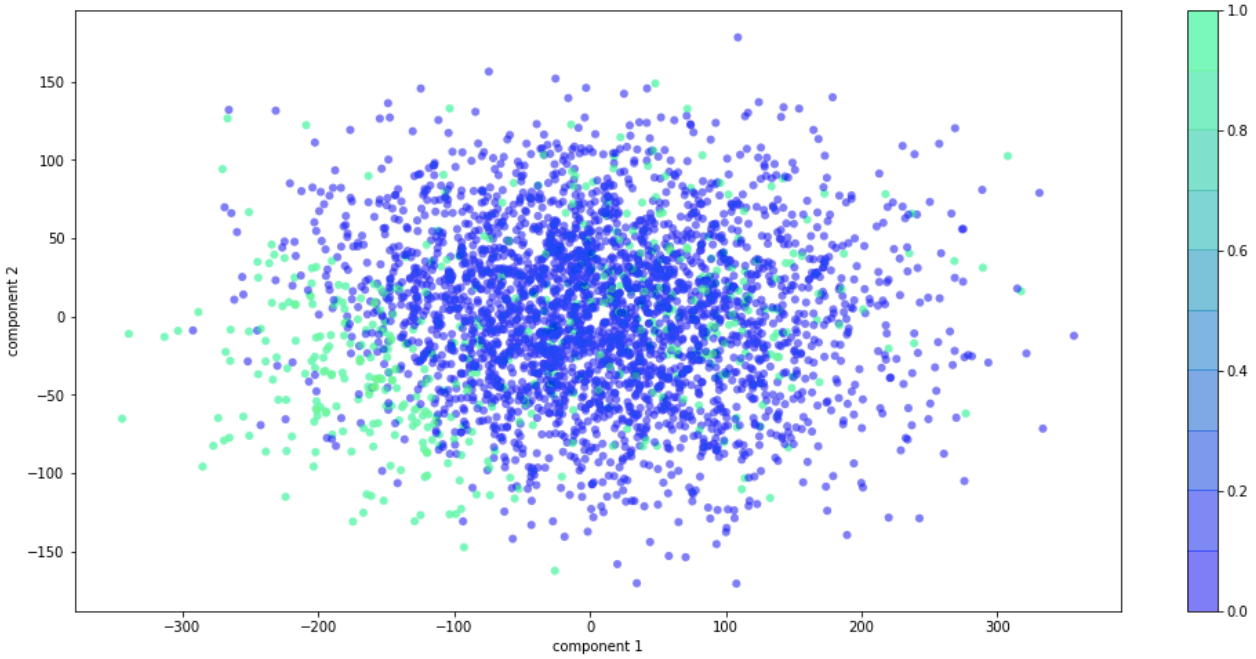
2.1.1.1 Multivariate Analysis

Multivariate analysis is the analysis of more then one variable in a dataset. Multivariate analysis becomes important when we have large dimensional data

to visualise and it becomes very difficult to visualise every predictor variable individually. It also helps us to identify the dominant patterns and clusters in the data.

So, we will first look at one of the most widely used algorithm to visualise high dimensional data based on Eigen Value and Eigen vectors which is called as *Principal Component Analysis*. In PCA visualisation, we have plotted the scatter plot along two dimensions.

Figure 2.5 PCA Visualisation of the data



In the above plot , 52.67% of the variance in the data is explained by the principal component 1 and 13.73% of the variance in the data is explained by the principal component 2. Also,It is hard to interpret anything from the above plot except the fact that the positive labels are spread more widely then the negative labels and the negative labels are more dense then the positive label but it is obvious given the fact that our data is imbalanced and it contains more negative labels then the positive ones.

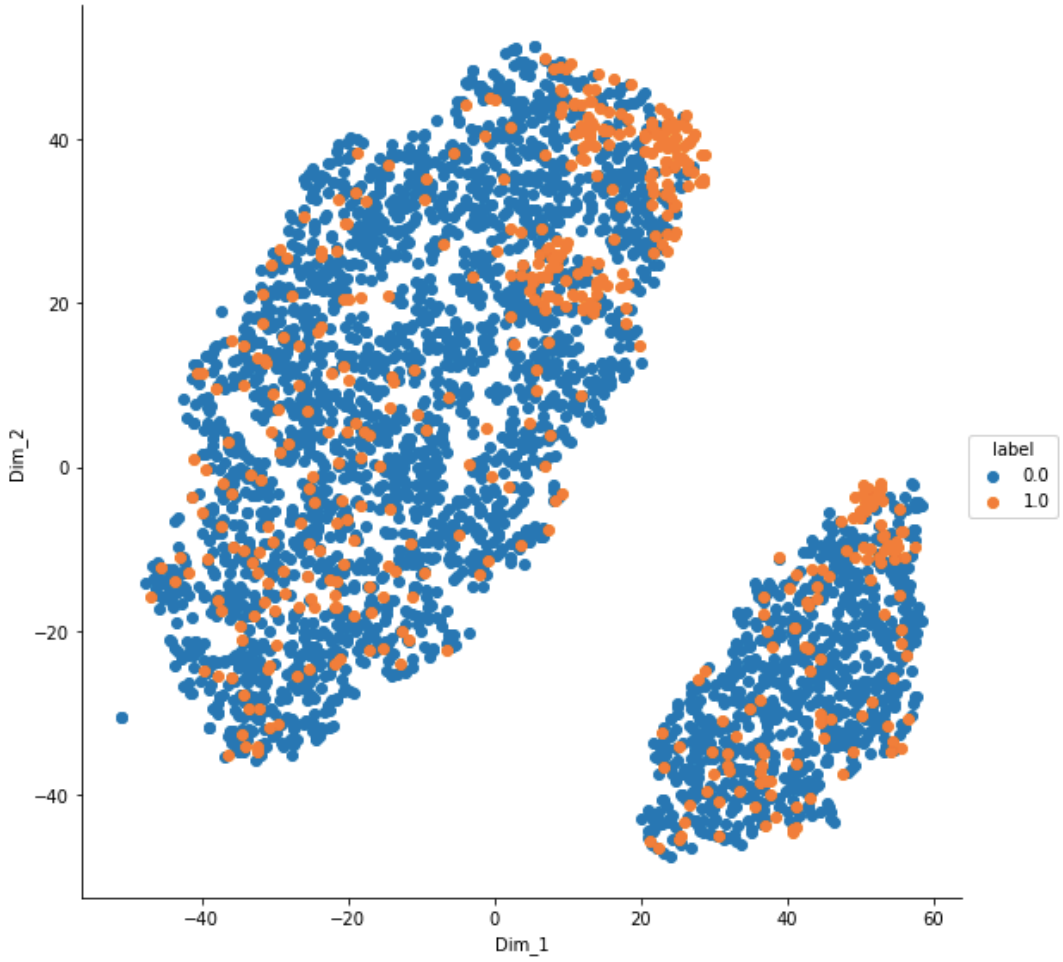


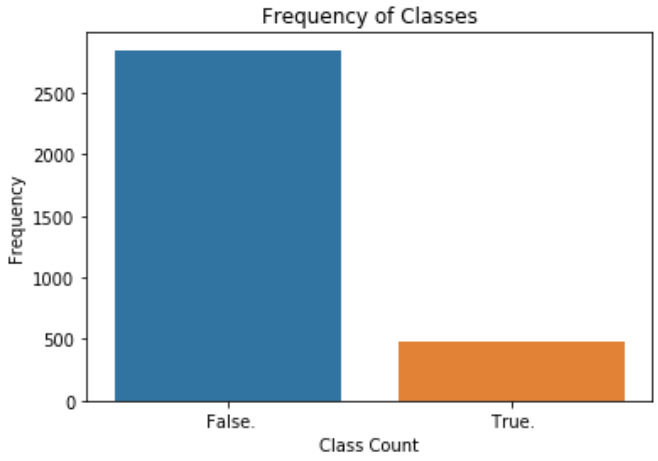
Figure 2.6 t-SNE Visualisation of the data

Now, lets have a look at even better algorithm then PCA which is called t-distributed Stochastic Neighbor Embedding , abbreviated as (t-SNE). It is a dimensionality reduction technique and is well-suited for embedding high-dimensional data for visualisation in a low-dimensional space of two or three dimensions.Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.

The above t-sne plot form two clusters of evenly scattered points along two dimensions for both positive and negative labels among the two clusters. *But again, it is hard to interpret anything from the t-sne plot as both the positive and negative label points are evenly distributed among the clusters.*

2.1.2 Data Preparation and Cleaning

2.1.2.1 Handling Imbalance Data



Looking at the above class distribution plot for the training data.We can clearly observe that the training data is imbalanced. It has 2850 data-points labelled as False and only 483 data-points labelled as True. So, almost 85% of the data is labelled as False and only 15% of the data labelled as True.

This imbalance data can make a dumb model look good. Imbalance data always favours the majority class if the right algorithms are not chosen. Let suppose , if we train a model on the train data which has 85% of data labelled as False will give an accuracy of 85% even if it incorrectly predicts all data-points that are labelled as 'True', which looks descent but this model doesn't help us in anyway and therefore this model can be termed as a dumb model. So, choice of a sensible performance metric becomes important here.

So, we can either upsample the data and make it balanced or we can use algorithms that are robust to imbalanced data like tree and ensemble models.I tried upsampling the data and then training the data on top of it but it doesn't help the model in anyway. So, we will train algorithms that can train imbalanced data well.

2.1.2.2 Outliers Analysis

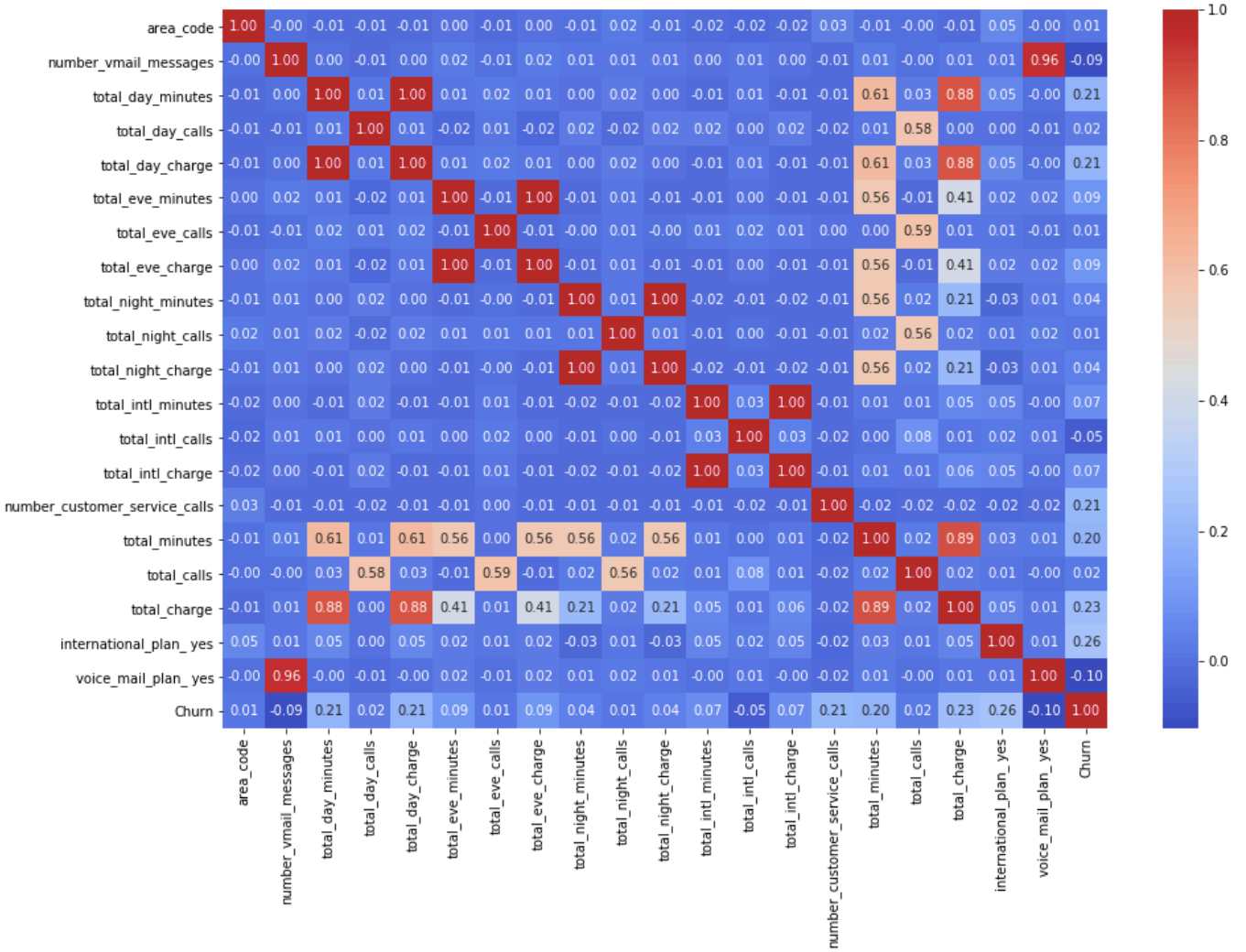
In Box plots analysis of individual features, we can clearly observe from these box-plots that, not every feature contains outliers or they either have very few outliers. Also, given the constraint that, we have only 3333 data-points and after

removing the outliers, the data gets decreased by almost 15%. So, we will not drop the outliers. Instead we will choose algorithms like Random Forest that are not very much sensitive to the outliers for the modeling .

2.1.2.3 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. We do this by plotting heat-map of the correlation plot for our data.

Figure 2.7 Correlation Plot of the data



The above plot is a correlation plot which gives correlation between the features and helps us to reduce the feature space. From the above plots, we can observe the features that are correlated to one other. According to above plot, we can observe that*, total_day_charge : total_day_minutes, total_eve_charge : total_eve_minutes, total_night_charge : total_night_minutes, total_int_charge : total_int_minutes, total_charge : total_minutes, voice_mail_plan_yes and number_vmail_messages*. So, we can remove either of the features among the above six pairs and reduce the feature space by six.

2.2 Modeling

We always start our model building from the most simplest to more complex. Therefore we start with Naive Bayes.

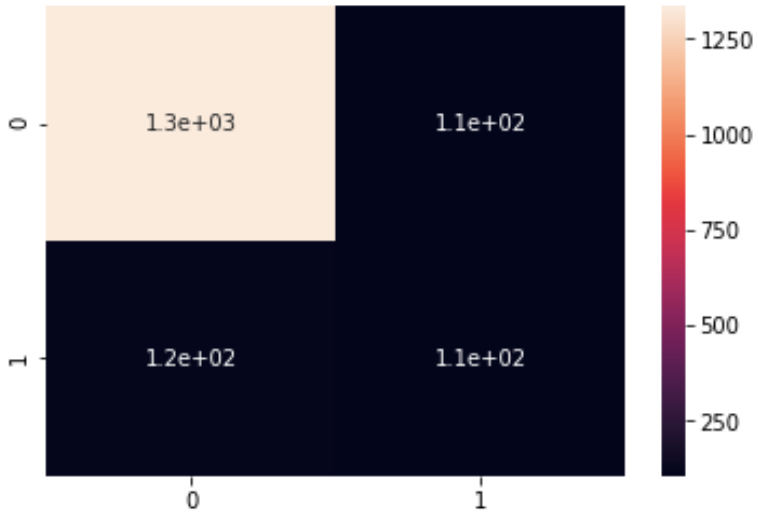
2.2.1 Naive Bayes

We do not expect a very good performance from Naive Bayes, as our data is imbalanced and also has a few outliers. We are training this model, so that we can stack up all the models at last and hope to improve our final model's performance. So, after training Naive Bayes on our data, we get the following result :

```
Train Data
Precision: 0.503 / Recall: 0.476 /fscore : 0.489 /Accuracy: 0.856
Test Data
Precision: 0.493 / Recall: 0.473 /fscore : 0.483 /Accuracy: 0.864
```

Although, the performance of the model is not impressive enough. We can conclude that our model is not overfitting as the train and test errors are almost similar. But lets have a look at the confusion matrix.

Figure 2.8 Confusion matrix for Naive Bayes



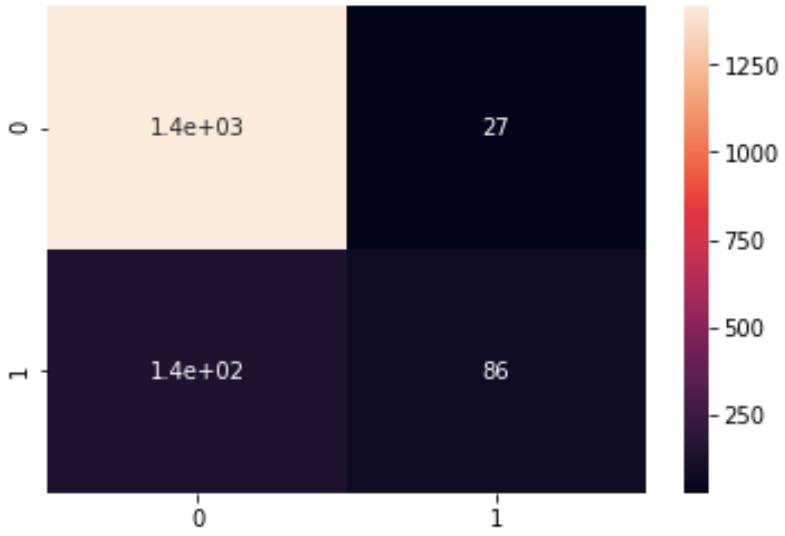
2.2.2 KNN

After hyper parameter tuning of the KNN algorithm, we train our final KNN model with 3-neighbors and get the following result on train and test data.

```
Train Data
Precision: 0.93 / Recall: 0.656 /fscore : 0.769 /Accuracy: 0.943
Test Data
Precision: 0.703 / Recall: 0.433 /fscore : 0.536 /Accuracy: 0.899
```

Although, KNN performs better then Naive Bayes in almost every performance measure, It seems that our model is overfitting as the difference between train and test error is significant but this is due to the fact that, we do not have enough data to perform k-fold cross-validation. So, lets have a look at the confusion matrix for knn.

Figure 2.9 Confusion matrix for KNN



As, It is obvious from the Recall value of 43.3%, our model predicts only 86 positive labels correctly, given that the test data has 224 positive labelled points.

2.2.3 Logistic Regression

Logistic regression provides class balancing as one of its feature. So, if we have an imbalanced dataset, Logistic regression will first upsample it and then it will

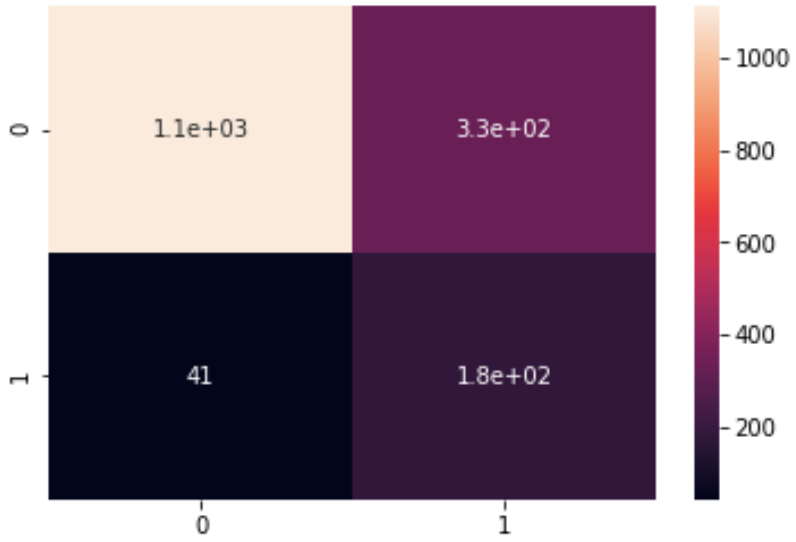
train the model. So, we will train our model, both on balanced and imbalanced class.

After training the model with *class balancing* and hyper parameter tuning we get the following performance :

```
Train Data
Precision: 0.361 / Recall: 0.758 /fscore : 0.489 /Accuracy: 0.771
Test Data
Precision: 0.356 / Recall: 0.817 /fscore : 0.496 /Accuracy: 0.777
```

So, after looking at the performance measure for the logistic regression with class balancing, we can say that it did best in terms of Recall, that means it was 81.7% successful in predicting the Positive class label which is much better then Naive Bayes and KNN. Although it incorrectly predicts a lot of negative labelled points as positive. So, this model too wont be of very much use to us. Also, we can see that the model doesn't overfits at all as the train and the test error significantly similar.

Figure 2.10 Confusion matrix for Logistic Regression(Class Balancing)



The confusion matrix supports the fact that, this model was successful in predicting the customer who would 'Churn' and it only misidentified 41 positive labelled data-points.

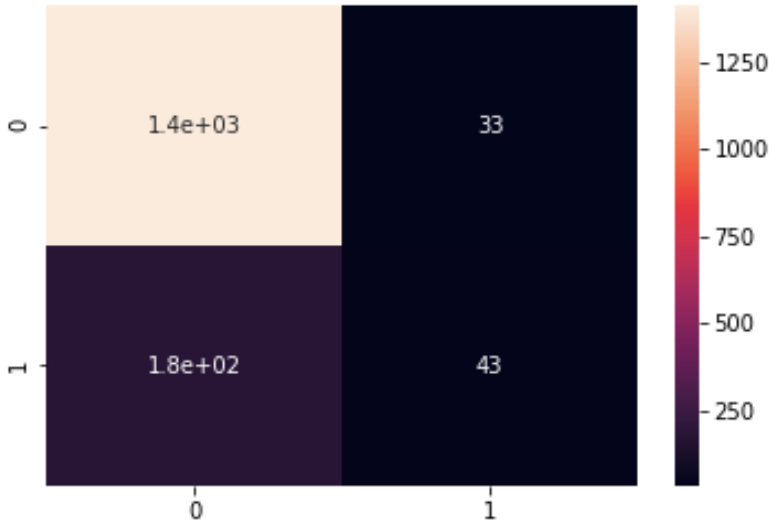
```
Train Data
Precision: 0.574 / Recall: 0.217 /fscore : 0.315 /Accuracy: 0.863
Test Data
Precision: 0.566 / Recall: 0.192 /fscore : 0.287 /Accuracy: 0.872
```

So, now lets have a look at the performance of ***logistic regression without class balancing***, which doesn't balance the data and trains the model on the data as it is given.

The performance measure of logistic regression without class balancing is a true example of why accuracy cannot be a main measure of performance of a model when the dataset is imbalanced. With just a just a recall of 19.2% and a precision of 56.6%, this model has got an accuracy of 87.2% which is impressive, even though our model is dumb.Although this model doesn't overfits , it doesn't do any good either.

Lets have a look at the confusion matrix of this model :

Figure 2.11 Confusion matrix for Logistic Regression(Without Class Balancing)



The confusion matrix plotted above supports the fact that this model did not perform well. This model only predicted 43 positive labelled points as positive out of 224 positive labelled points.

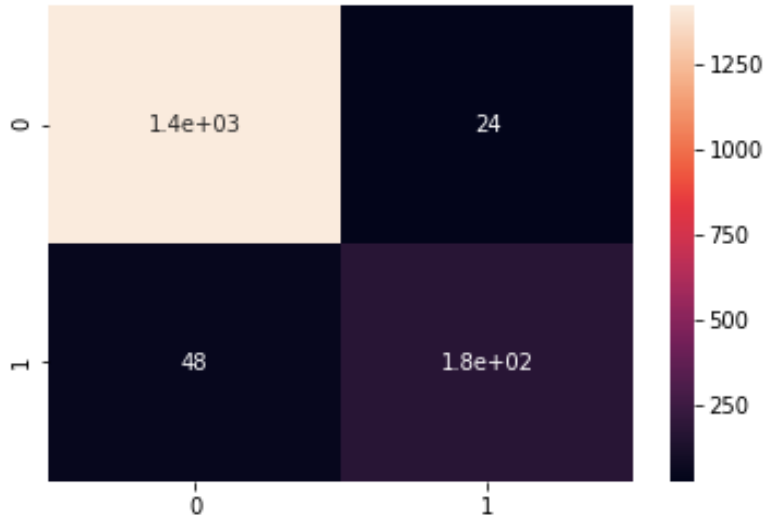
2.2.4 Decision Trees

We expect decision trees to perform better then any of the above algorithms as Decision trees can handle imbalanced data better then Naive Bayes, KNN and Logistic Regression. So, lets have a look at the performance of Decision Trees after hyper-parameter tuning.

```
Train Data
Precision: 0.998 / Recall: 0.88 /fscore : 0.935 /Accuracy: 0.982
Test Data
Precision: 0.88 / Recall: 0.786 /fscore : 0.83 /Accuracy: 0.957
```

Looking at the above performance measure of Decision Trees, we can argue that this model did better then all the models that we trained above because, although this model seems to overfit but gives pretty impressive results on test data. This model can correctly predicts 78.6% of positive labelled data points with a good precision score. Lets have a look at the confusion matrix of the above model to make things clear.

Figure 2.12 Confusion matrix for Decision Trees



In the above confusion matrix, we can observe that only 48 positive labelled data-points are incorrectly classified as negative labelled and only 24 negatively labelled data-points are incorrectly classified as positive labelled. Decision Trees turned out to be a pretty good model.

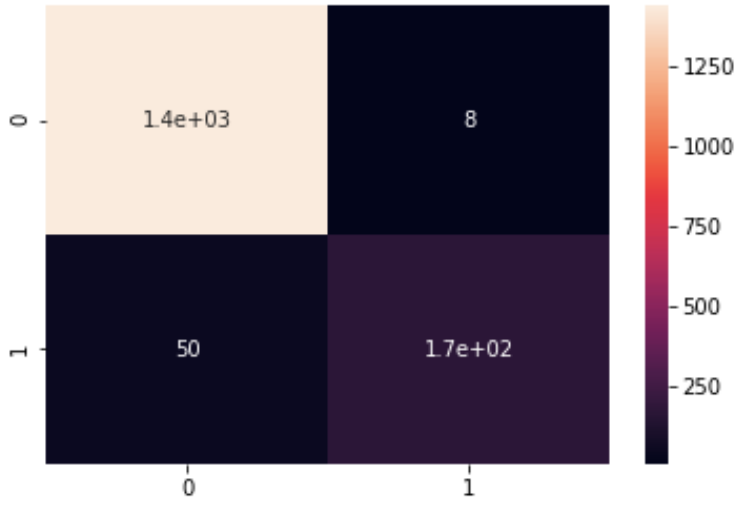
2.2.4 Random Forest

We expect even better results from Random Forest then Decision Trees, as Random forest is better then Decision trees in handling imbalance data. So, lets have a look at the performance of the Random Forest algorithm on our dataset after hyper parameter tuning.

```
Train Data
Precision: 1.0 / Recall: 1.0 /fscore : 1.0 /Accuracy: 1.0
Test Data
Precision: 0.956 / Recall: 0.777 /fscore : 0.857 /Accuracy: 0.965
```

Looking at the performance measure of Random Forest, we can argue that its performance doesn't vary much in terms of Precision-Recall trade off. Also, it overfits as the difference in the train and test error of the model is significant. Lets have a look at the confusion matrix for the same for much better understanding.

Figure 2.13 Confusion matrix for Random Forest



In the above confusion matrix, we can observe that 50 positive labelled data-points are incorrectly classified as negative labelled and only 8 negatively labelled data-points are incorrectly classified as positive labelled.

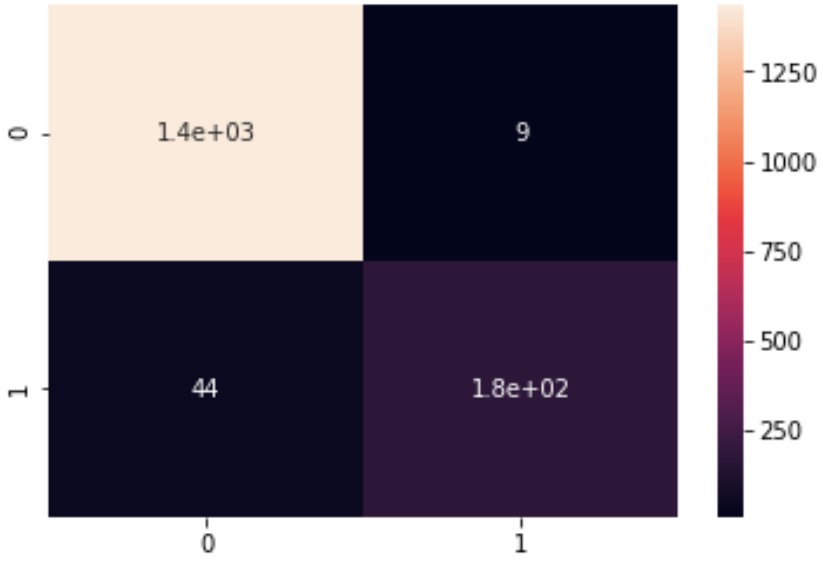
2.2.4 XGBoost

XGBoost is one of the strong algorithm based on ensembles. We can expect to see impressive performance results from the same. So, lets have a look at the performance measure of xgboost after hyper parameter tuning.

```
Train Data
Precision: 1.0 / Recall: 0.874 /fscore : 0.933 /Accuracy: 0.982
Test Data
Precision: 0.952 / Recall: 0.804 /fscore : 0.872 /Accuracy: 0.968
```

We can clearly observe that, XGBoost out performs all the above trained models. XGBoost gives the best performance with the best Precision-Recall trade off of 0.952 to 0.804 respectively. So, lets look at the confusion matrix to get a much better understanding.

Figure 2.14 Confusion matrix for XGBoost



Here, we can observe that only 44 positive labelled data-points are incorrectly classified as negative labelled and only 9 negatively labelled data-points are incorrectly classified as positive labelled. So, XGBoost turned out to be the best model of all.

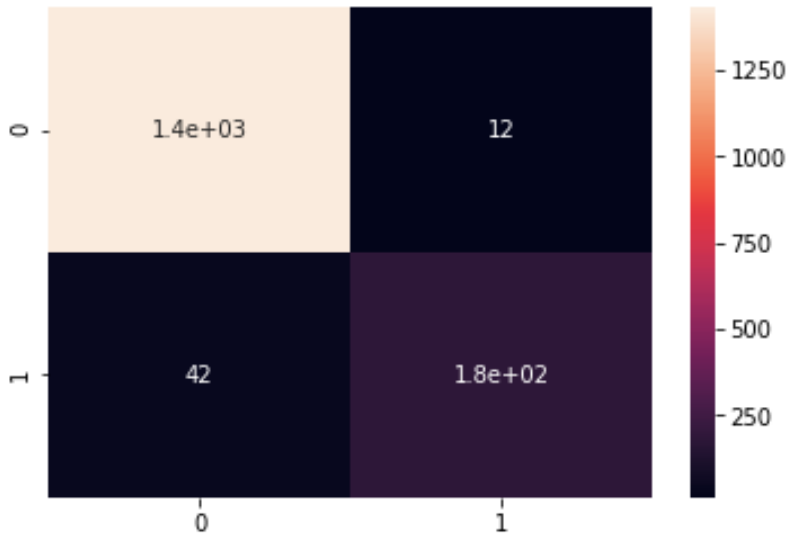
2.2.5 Stacking the models

So, by now we have trained all the models and have looked at their performances. So, lets take up every model that we have trained and stack them up and see if it improves the performance.So, lets look at the performance measure after stacking up all the models together.

```
Train Data
Precision: 1.0 / Recall: 1.0 /fscore : 1.0 /Accuracy: 1.0
Test Data
Precision: 0.938 / Recall: 0.812 /fscore : 0.871 /Accuracy: 0.968
```

This model also seems to overfit but gives almost similar results to XGBoost based on Precision-Recall tradeoff. Although it gives the best Recall of 81.2% of all all the models. Lets have a look at the confusion matrix of the same.

Figure 2.14 Confusion matrix for Stacking Classifier



In the above confusion matrix, we can observe that only 42 positive labelled data-points are incorrectly classified as negative labelled and only 12 negatively labelled data-points are incorrectly classified as positive labelled.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1.Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Customer Churn, the latter two, *Interpretability* and *Computation Efficiency*, do not hold much significance. Therefore we will use *Predictive performance* as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

3.1.1 Precision

Precision talks about how precise/accurate our model is out of those predicted positive, how many of them are actual positive. Precision is a good measure to determine the models performance, when the costs of False Positive is high.

3.1.2 Recall

Recall actually calculates how many of the Actual Positives our model capture through labelling it as Positive (True Positive). Recall is a good measure to determine the model performance, when the costs of False Negative is high.

3.1 Model Selection

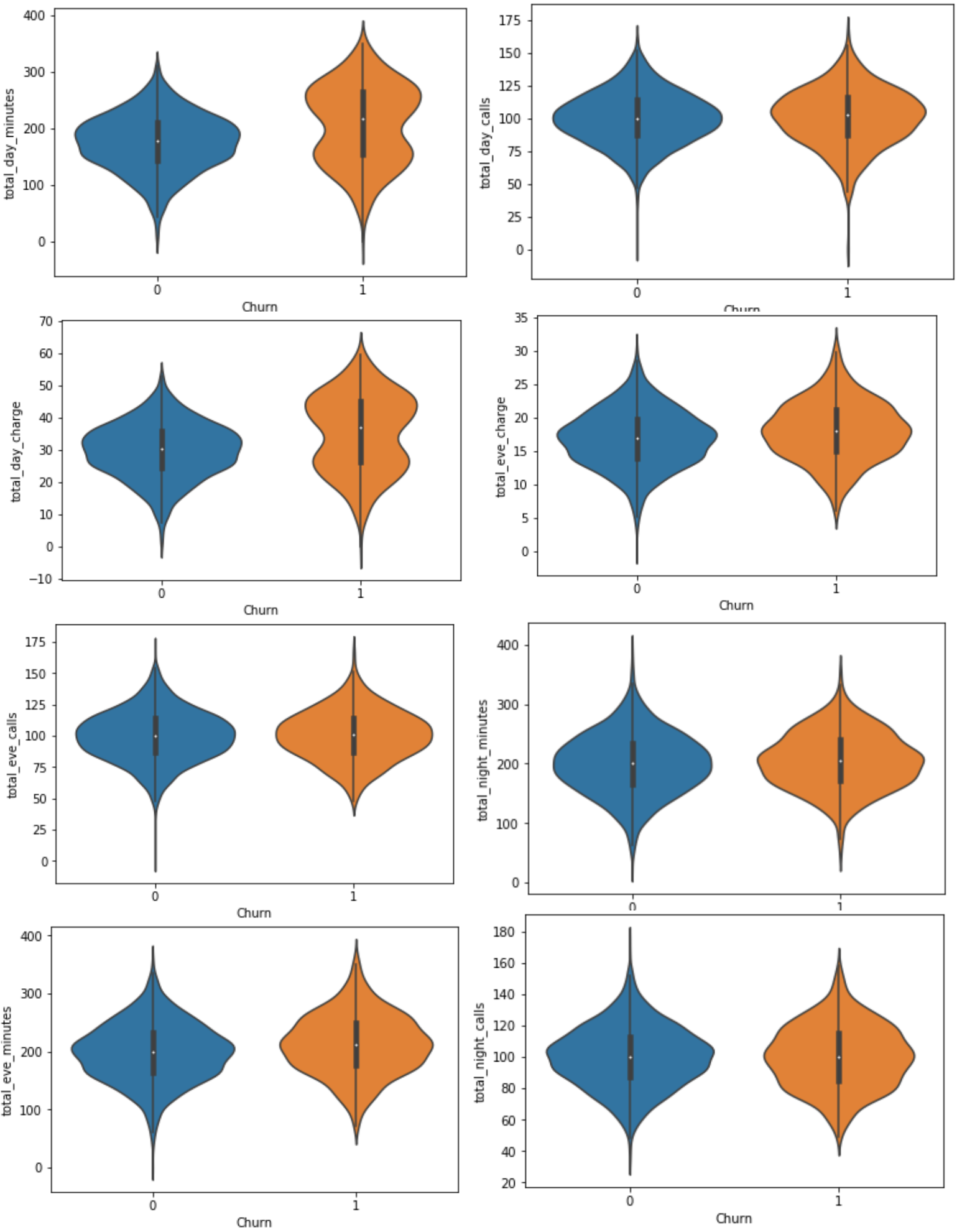
We saw that both models XGBoost and Stacking Classifier perform comparatively on average based on Precision-Recall trade off and therefore we can select either of the two models without any loss of information. Model comparison table is given below.

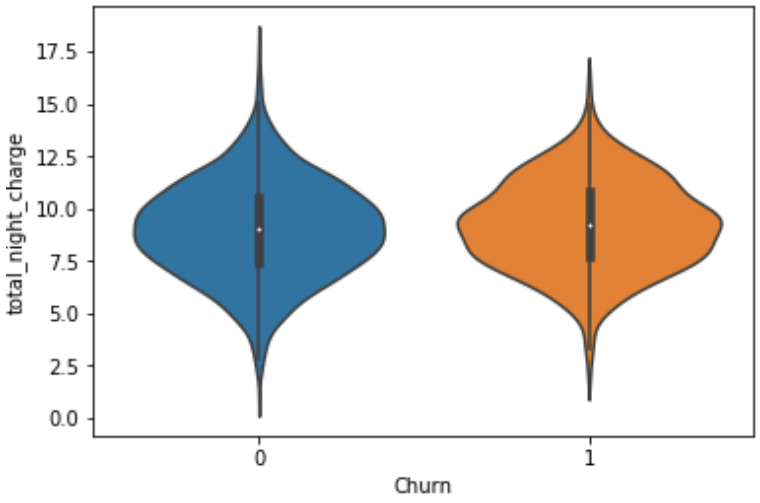
	Accuracy	Fscore	Model	Precision	Recall
0	0.864	0.483	Naive Bayes	0.493	0.473
1	0.899	0.536	KNN	0.703	0.433
2	0.777	0.496	Logis Regres(Class Balancing)	0.356	0.817
3	0.872	0.287	Logis Regres(without Class Balancing)	0.566	0.192
4	0.957	0.830	Decision Tree	0.880	0.786
5	0.965	0.857	Random Forest	0.956	0.777
6	0.968	0.872	XGBoost	0.952	0.804
7	0.968	0.871	Stacking Classifier	0.938	0.812

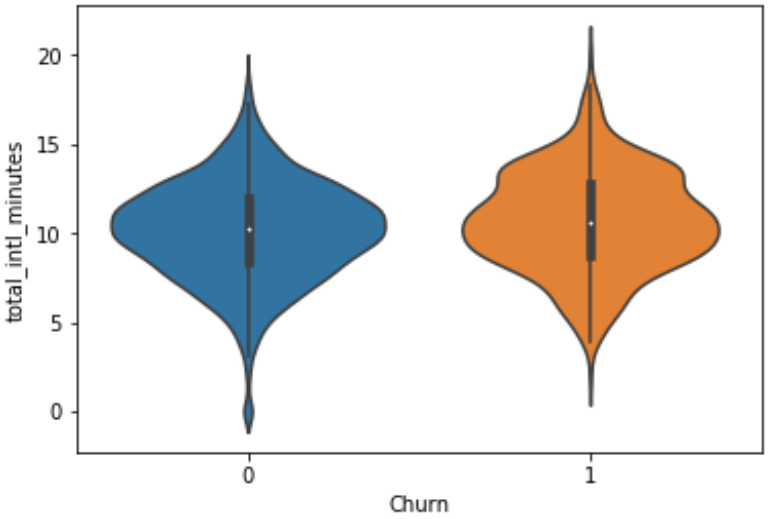
So, It is obvious from above model performance comparison table, both XGBoost are Stacking Classifier are apt for our task i.e to predict the customers who are about to Churn with very low false positive rates

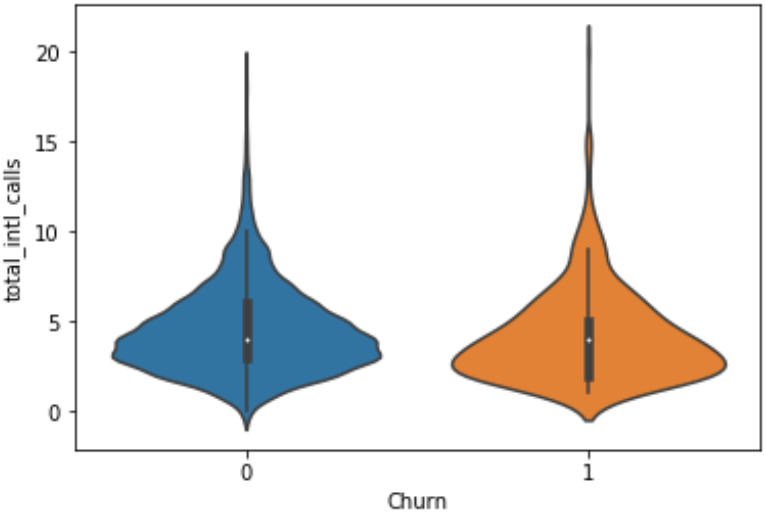
Appendix A - Extra Figures

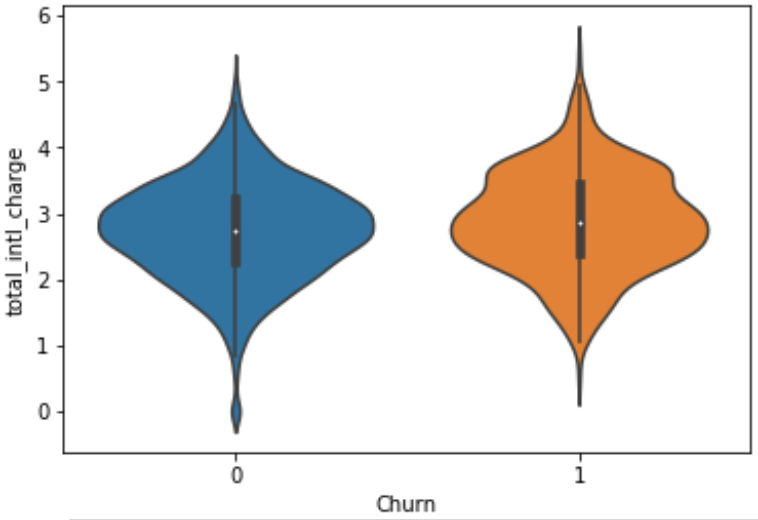
Fig A.1 Violin Plots of Customer behaviour based on labels

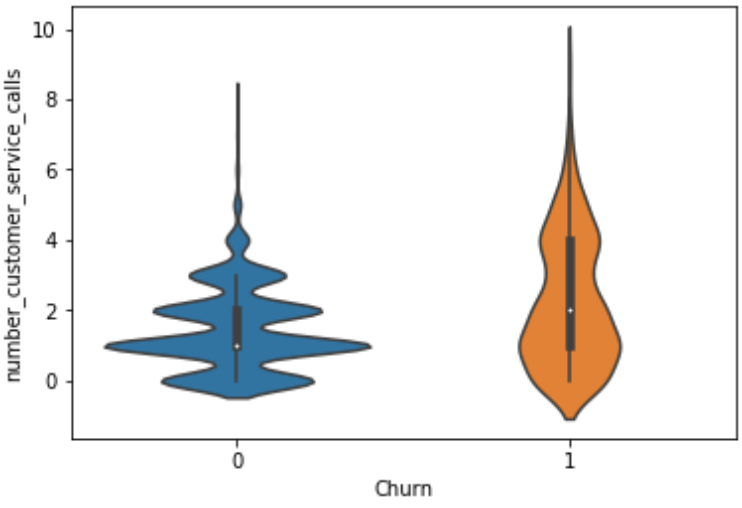



total_night_charge
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
1
Churn


total_intl_minutes
20
15
10
5
0
0
1
Churn


total_intl_calls
20
15
10
5
0
0
1
Churn


total_intl_charge
6
5
4
3
2
1
0
0
1
Churn


number_customer_service_calls
10
8
6
4
2
0
0
1
Churn

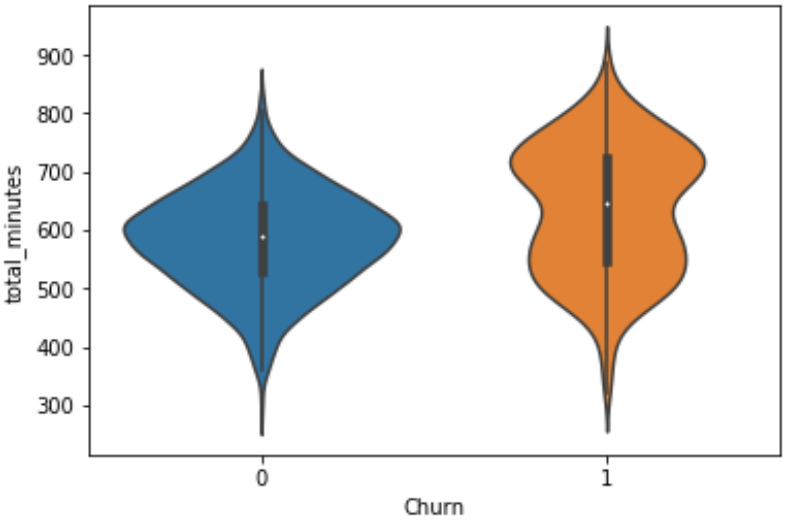
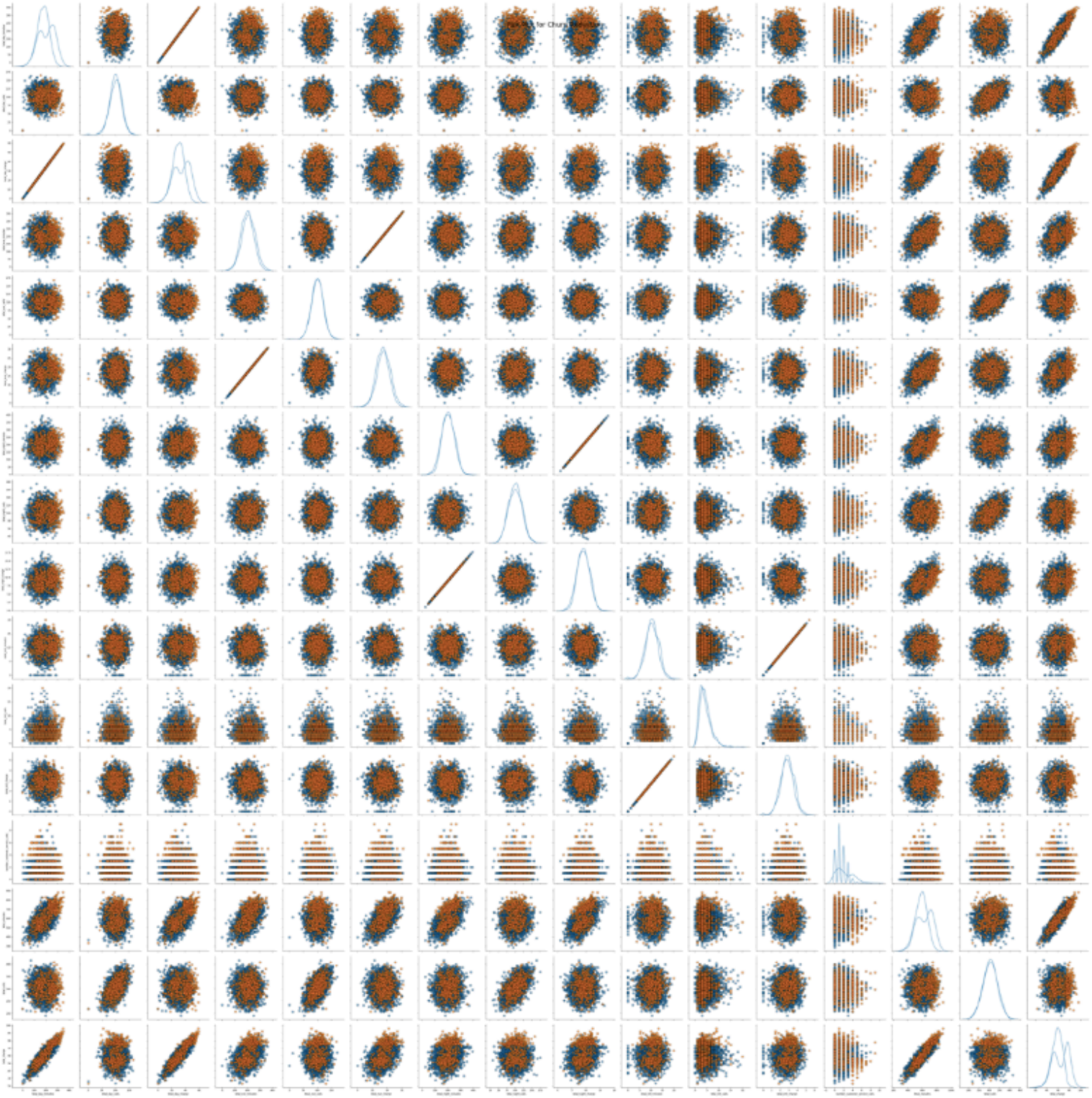

total_minutes
900
800
700
600
500
400
300
0
1
Churn

Fig A.2 pair Plots Of customer behaviour



References

Andrew Ng's Machine Learning course on Coursera (or, for more rigor, Stanford CS229).

An Introduction to Statistical Learning by Gareth James et al. Excellent reference for essential machine learning concepts, available free online.

All of Statistics: A Concise Course in Statistical Inference, by Larry Wasserman. Introductory text on statistics

MIT 18.05, Introduction to Probability and Statistics, taught by Jeremy Orloff and Jonathan Bloom. Provides intuition for probabilistic reasoning & statistical inference, which is invaluable for understanding how machines think, plan, and make decisions.