# Problem Statement

For a B2C company customer churn is a very serious problem, the business intelligence team have come up with some attributes that they think contributes to customer churn. We would like to get help with Machine Learning to put up a risk score against each customer.

# Data Set at hand

Data was provided in three different csv files

1. Train.csv
2. Train\_col\_name.csv
3. Train\_lable.csv

In total there are 36,000 observations with 37 features.

# Methodology & Tools

Just like any other data science project, CRISP-DM cycle was followed when doing this assignment by starting with business understanding, data understanding, data preparation, modelling, evaluation and finally going back again the cycle till the desired results were achieved.

Two different tools were used in this project:

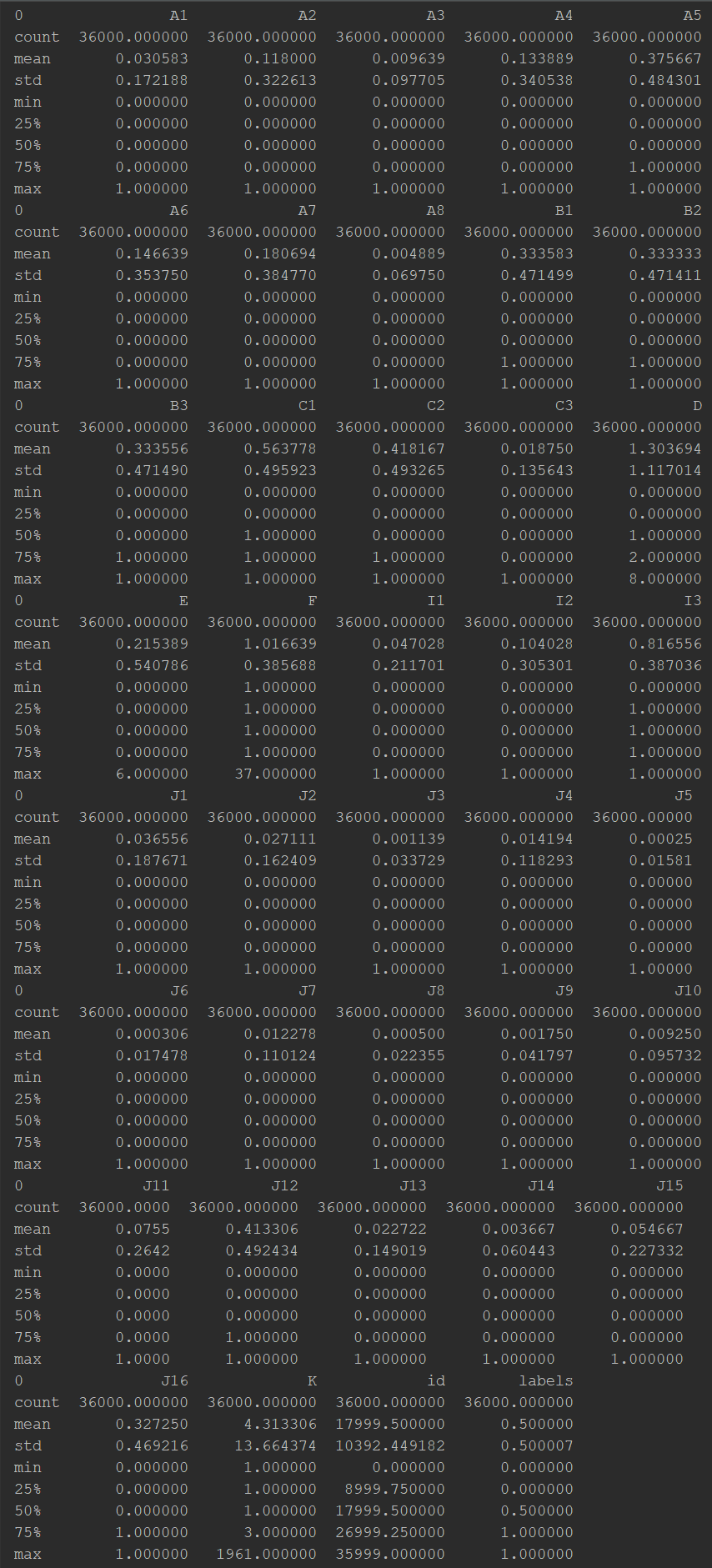
1. Azure ML Studio
   * Microsoft provides this machine learning cloud-based tool which was also used in this project for everything between data preparation to model deployment.
2. Python
   * Pandas
     + Used for mostly Data frames and series.
   * Numpy
     + Used for small functionalities like arrange and taking absolute values
   * Matplotlib
     + Used for distribution and count plots
   * Scipy
     + Used for getting z-scores
   * Seaborn
     + Used for plotting categorical plots
   * Sklean
     + Used for feature engineering and machine learning

# Data Preparation

First step was to consolidate all 3 csv files into one. Following steps were performed for preparation:

1. Fetch all csvs
2. Transpose headers then add.
3. Add ids col

# EDA

Several different techniques were used in order to explore the data, both on python and on Azure ML studio. Following observations were made about the data:

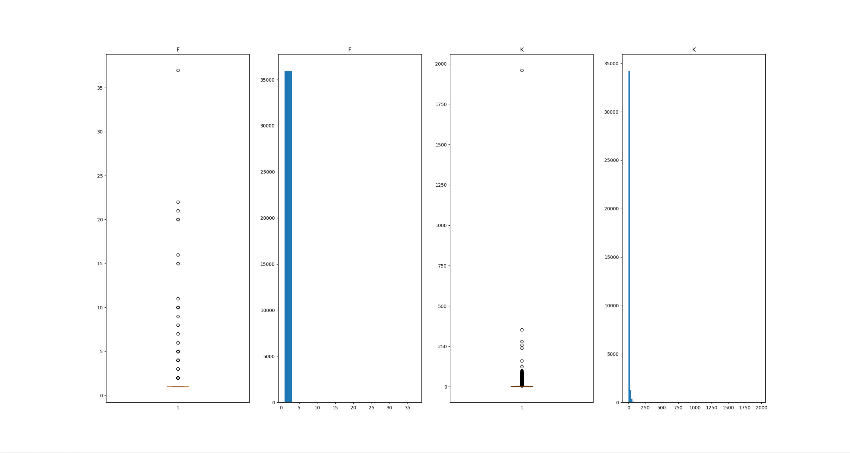
* There were no missing values
* All data types were consistent and no issues were found with the observations. Nonetheless, all data types were explicitly converted to integers to ensure consistency
* D and E columns had 9 and 7 unique values respectively
* F had values ranging from 1 to 37
* K had values ranging from 1 to 1961
* Rest columns were all binary
* Out of 37 features, 35 could be treated as categorical features, F and K were treated as continuous
* Outliers were spotted when visualizing the distribution of the features.
* Counts of each feature were plotted against the label. Some features were biased towards one category having frequency ratio below 5%.

Figure 1 - Data Description

Figure 2 - Categorical Features Distribution

Figure 3 - Continuous Features Distribution

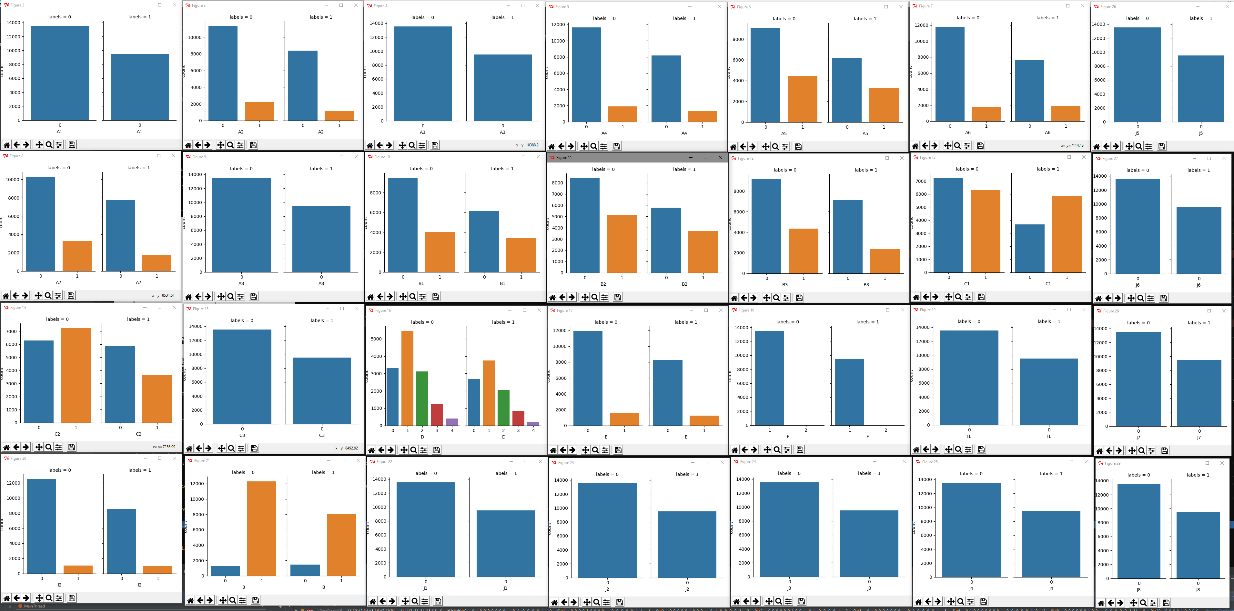


Figure 4 - Count of categories against labels - After removing outliers (1)

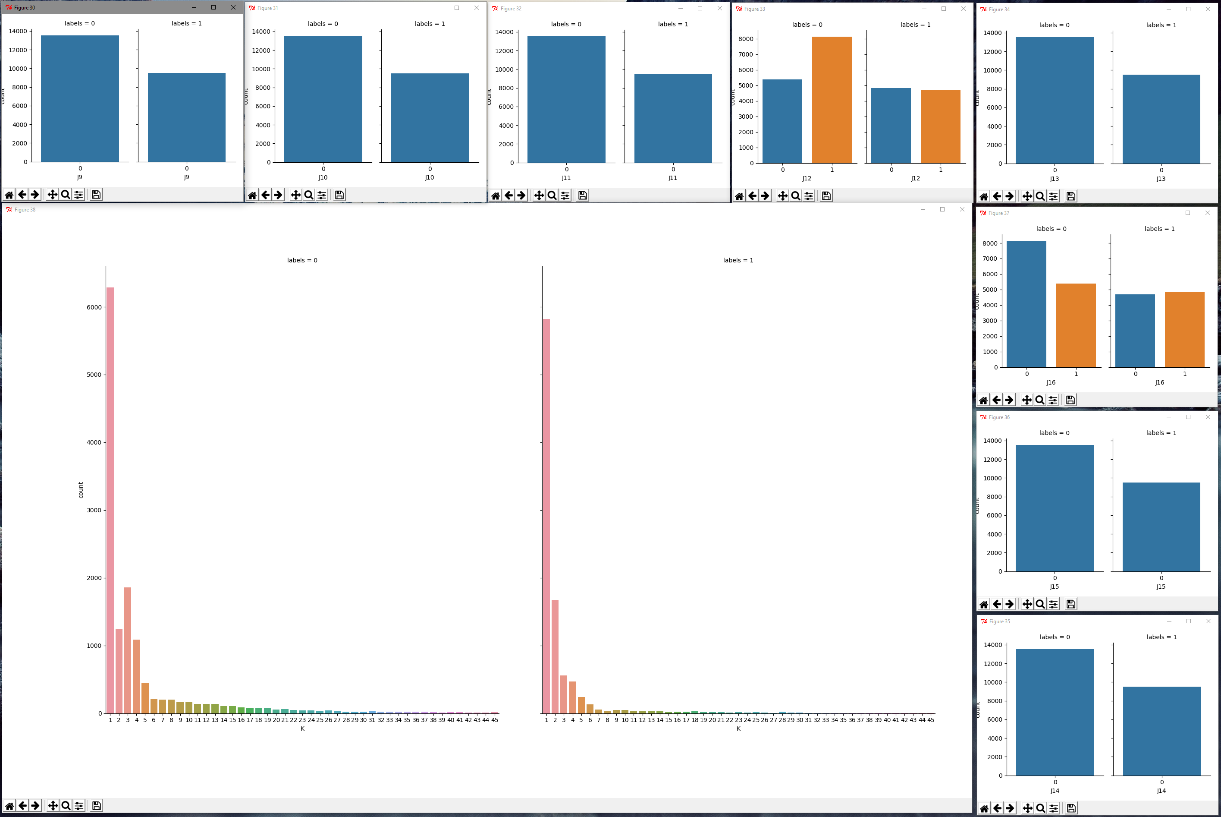


Figure 5 - Count of categories against labels - After removing outliers (2)

# Data Wrangling

After the observations made from the initial data analysis, following steps were performed in order to proceed to the next phase of feature engineering:

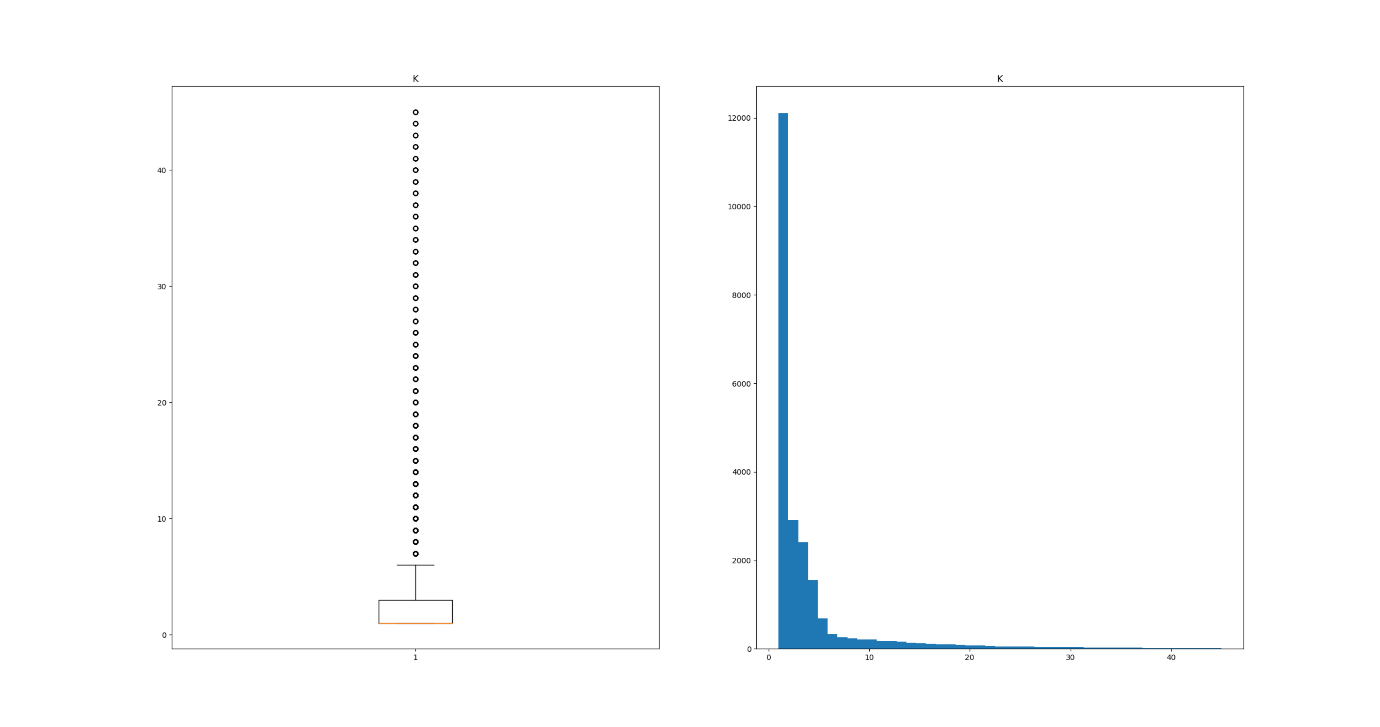
* Removed outliers
  + Two different approaches were made to tackle outliers for different models.
  + One technique was to take the z score of all columns and remove those whose absolute values were above 3.
  + Normally outliers are only removed from continuous features but when applying this technique, categorical features were also taken under consideration, the features which had very low category frequency division ratio were also affected and resulted features with just one category, which later on were excluded in feature engineering phase.
  + Other technique was to take the above and below 95 percentile of the numeric
  + features and replace them with the mean.
  + After removing outliers, feature F could now be treated as a categorical feature as 95 percent of the data was under value 1, the rest were outliers.

Figure 6 - Distribution of categorical features after removing outliers

Figure 7 - Distribution of K after removing outliers

* Changed the datatype first to all integers to consistency
* Changed datatypes to Boolean / categorical (Azure studio) for all categorical variables
* Took z-score for numerical features F and K. Although it was not necessary, but it made a slight impact on accuracy.

# Dimension Reduction

Several techniques were used in order to reduce the number of features used in multiple ml models. Reducing number of dimensions caused the accuracy to go higher. Following techniques were used:

* Permutational feature importance (linear regression, Decision Forest, Boosted Decision Tree, SVM)
* Removing categorical features with single category remaining.
* Principle component analysis
* Forward feature selection
* Backward feature reduction
* Chi-Square analysis
* Pearson Correlation

# Model Training

In this whole exercise, four different classification algorithms were used.

* Decision Forest
* Boosted Decision Tree
* SVM
* Logistic Regressions

After the data wrangling step, different dimension reduction techniques were used which each of the algorithm resulting in numerous numbers of trained model with different accuracies. Few of them are below:

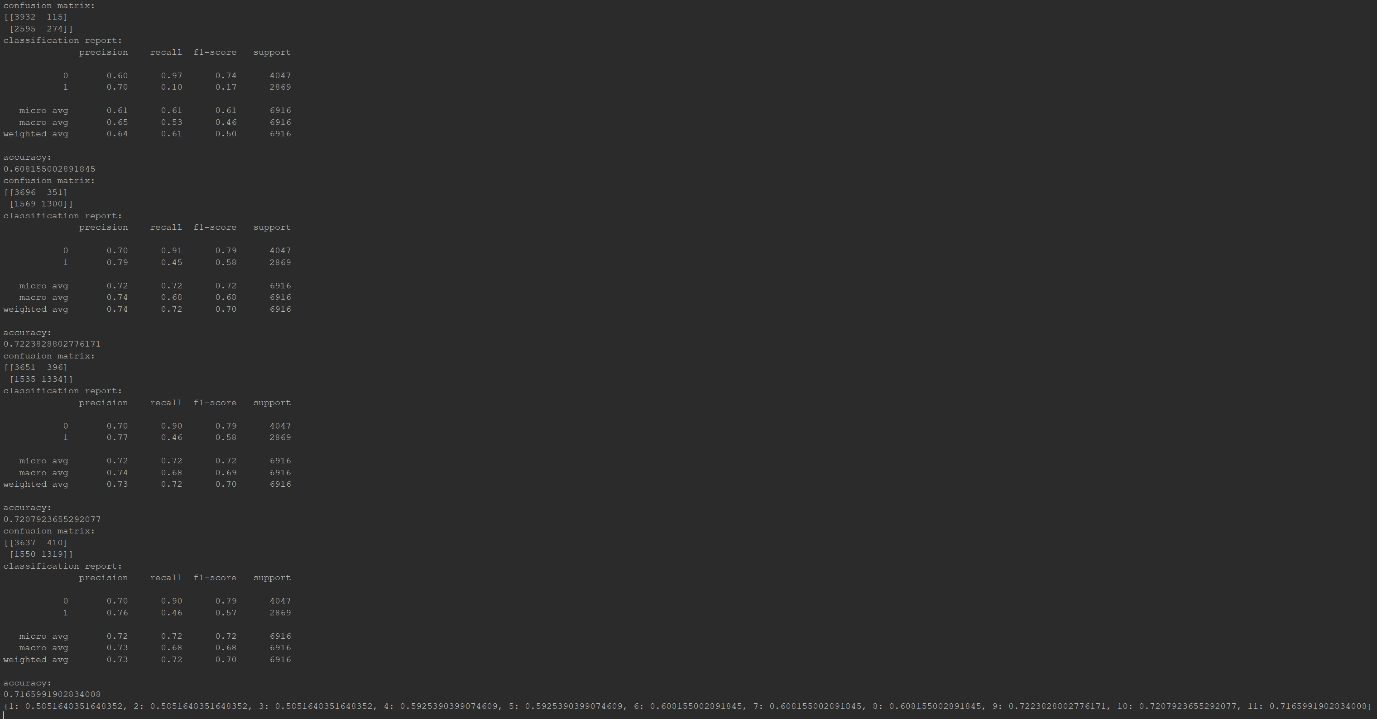
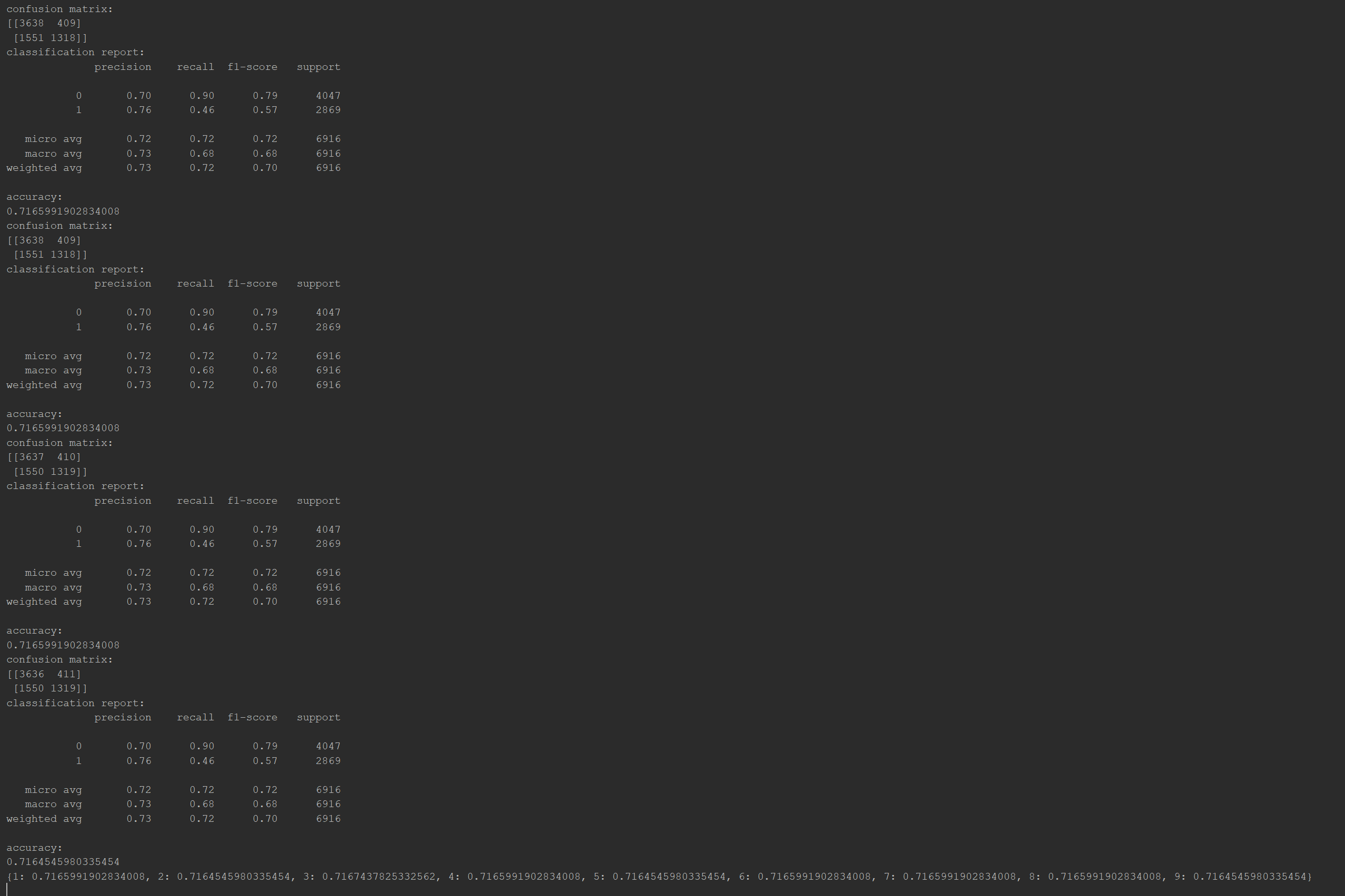
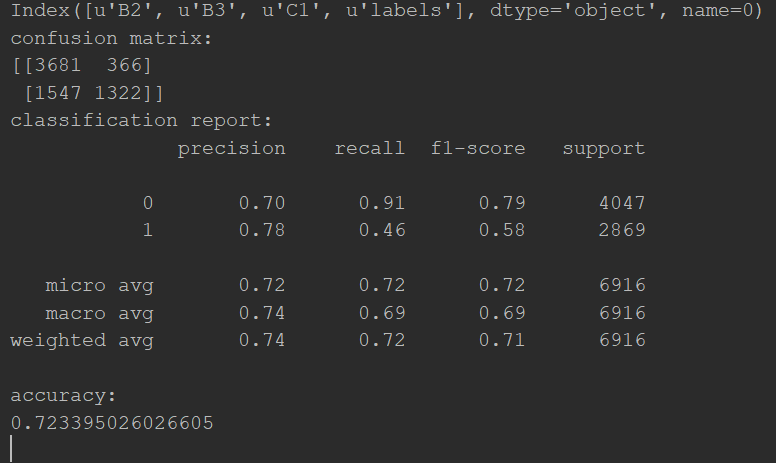
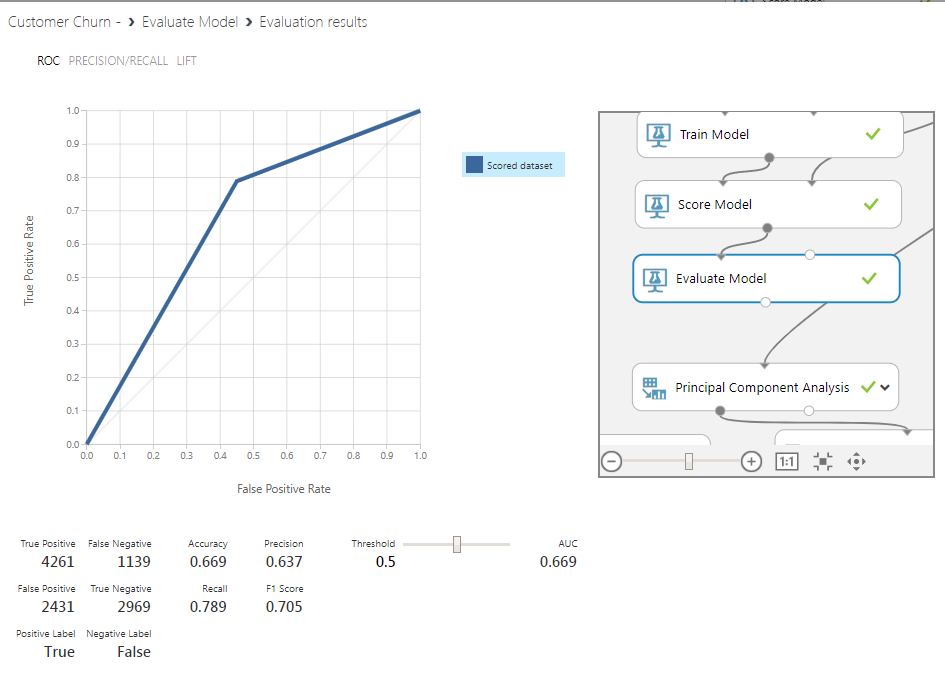
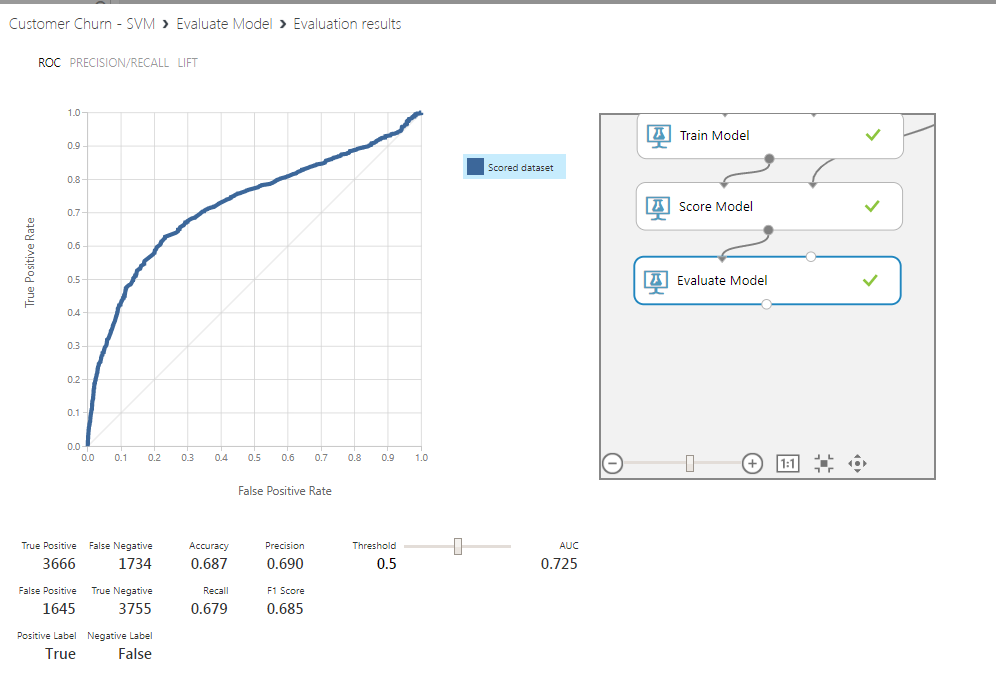
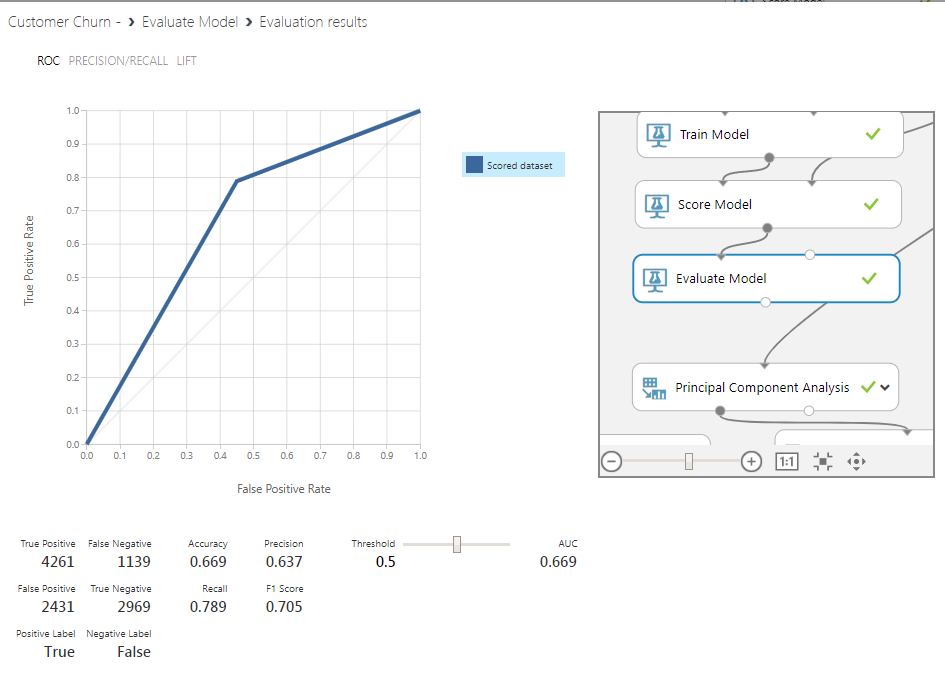
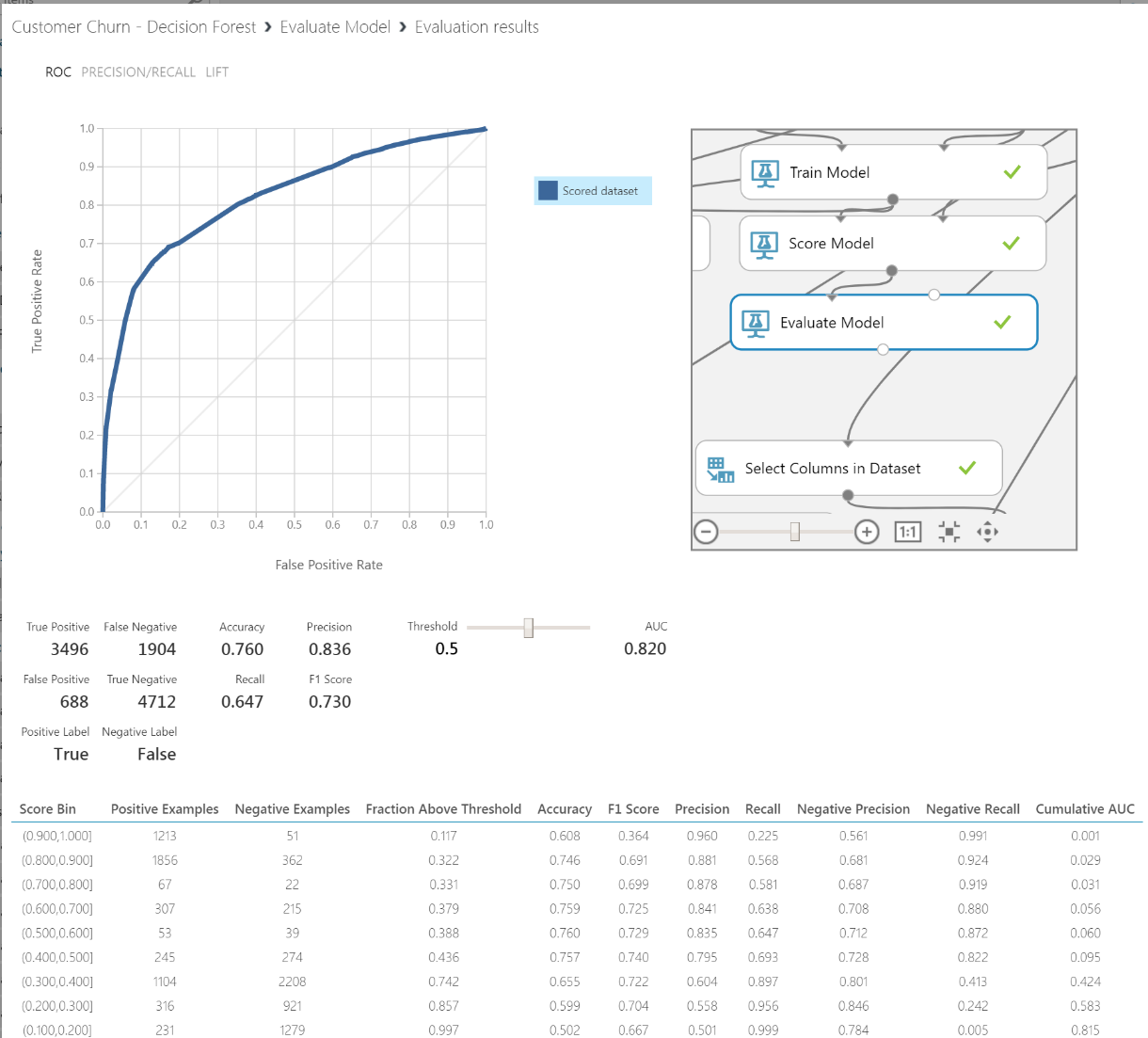
* Random Forest: In the below image, first the features were reduced by calculating permutational feature importance by running a linear regression against the label. Then the selected features were used to train a random forest classifier by starting with one feature and adding one more one by one. The last line in the console shoes the accuracies against each iteration.

Figure 8 - Random Forest - Forward Feature Selection

* Random Forest – After doing PCA. Different number of principle components were tried against the model. As the console shows, using only a single principle component and using 10 doesn’t make much of a difference.
* Random Forest – After doing permutational feature importance, forward feature selection and backward feature selection, the final number of features were reduced to three (B2, B3, C1). Among all the trained models in the python scripts, below one has the best evaluation of the random forest classifier:
* Boosted Decision Tree: Multiple feature selection methods were applied, and all had similar evaluation metrics. Below one is the best fit model in this algo series:
* SVM: The best model with this algorithm is below:
* Boosted Decision Tree: 
* Decision Forest: In all of this exercise, the best trained model was made using decision forest and feature selection by permutation feature importance which reduced the features used down to 7.

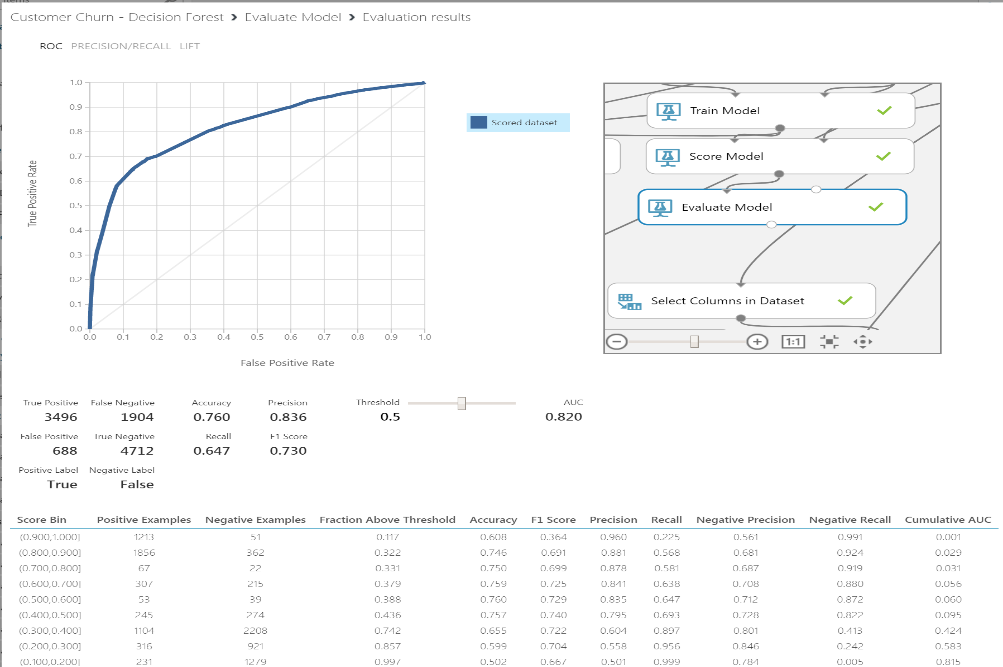
# Conclusion

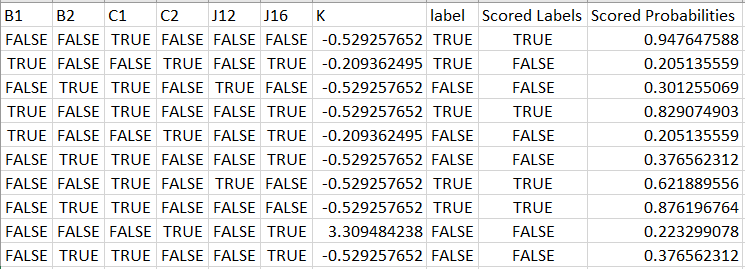
After all the data preparation steps and EDA, the final features which yielded the best accuracy were:

* B1
* B2
* C1
* C2
* J12
* J16
* K

The reasons why only these features were sufficient are that most of them have very similar distribution, highest density are against (True) values. So, if we use all of those similar ones, they will cause biasness. Same goes for the features which have only one category or have very low category frequency distribution. So, omitting all those features will generate a good machine learning model.

Among around ~ 30 different models which were trained using different techniques, the results of the best one is:





# Notes:

* All the work is on Github which can be accessed by: <https://github.com/abdulraheemabid/CustomerChurnModel.git>
* The github project includes
  + Python project for data prep, EDA and modelling.
  + All screenshots of models, most of them were not included in the report but can be found there.
  + Resultant dataset containing scored probabilities
  + Report itself
  + Csv files
* Models built on Azure ML Studio can be accessed via:
  + <https://gallery.cortanaintelligence.com/Experiment/Customer-Churn-Decision-Forest>
  + <https://gallery.cortanaintelligence.com/Experiment/Customer-Churn-SVM>