Customer Churn Prediction

Author: Shwet Prakash, Pranay Ankit, Putta Sachin, Lakshmi

Abstract

In this paper, we aim to document our methodologies in approaching the KKBox Churning Prediction Challenge. This challenge is essentially a classification problem, but the response variable is highly imbalanced. In the below sections, we will describe and visually explore the data sets. Then we will talk about several machine learning models we employed that are highly suitable for handling imbalanced data.

Introduction

In this paper, we aim to document our methodologies in approaching the KKBox Churning Prediction Challenge. This challenge is essentially a classification problem, but the response variable is highly imbalanced. In the below sections, we will describe and visually explore the data sets. Then we will talk about several machine learning models we employed that are highly suitable for handling imbalanced data.

Besides, handling and building effective machine learning models with imbalanced data is a common problem faced by data scientist in different industries such as credit risk analysis and churn prediction. Exploring and understanding the techniques and mechanisms behind the new machine learning models will also be beneficial to our future data science endeavors.

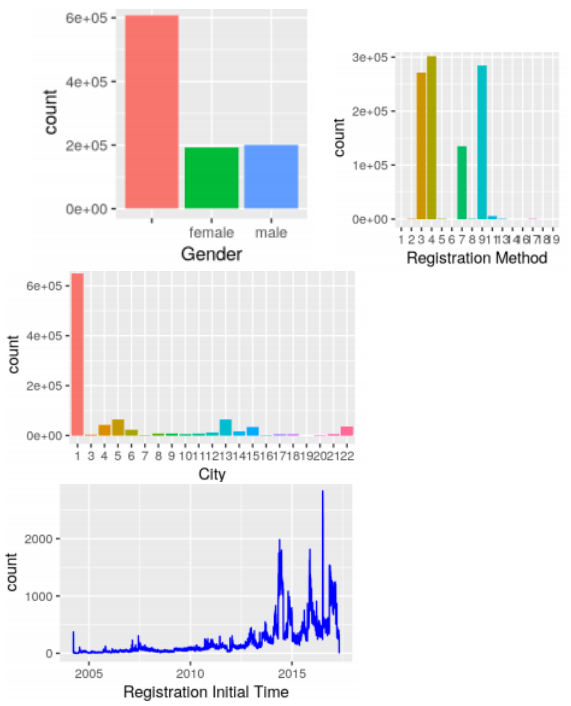
Data Description

Imbalanced Data

The most important problem as well as the focus of this project is the imbalanced nature of the data. In our training set, the response class has a ratio of 15:1 (no churn v.s churn). Therefore, conventional models (decision trees, random forests) and evaluation metric(prediction accuracy) will fail our purpose since it is the minority class (churn users) that we truly care about and with the majority of training data on majority class, the model will very much likely be unsuccessful.

Exploratory Analysis

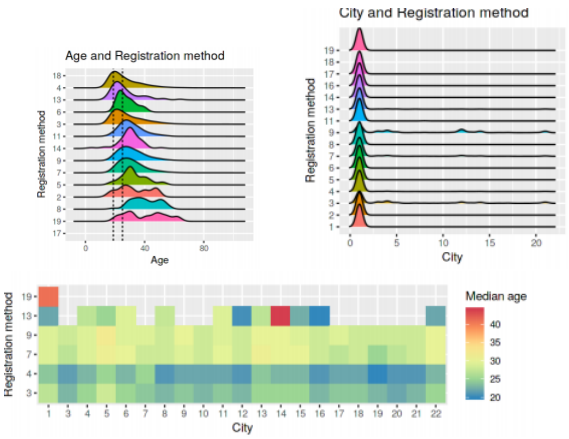
1. All the categorical variables are encoded numeric. They need to be transformed into factor variables for our visualization
2. There exist significant outliers. For instance, age range from as low as negative to over 1000.The min of total seconds played goes as low as to -9.223e+15.
3. Half of the gender information is missing



A quick glance at age, city, birthday distribution and time plot of user registration time. We observe that

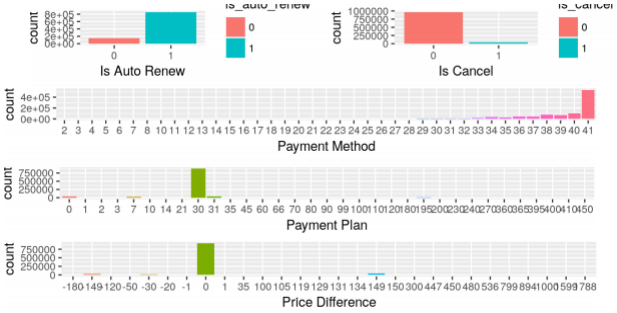
1. Gender distribution is pretty equal
2. The majority of users come from city 1.
3. The major registration method is 4,9,3.
4. a spike of user accumulation took place in 2014

Additionally, we want to explore the relationships among the features



Observe that

1. Young people mostly register through method 4 and 13. Older users mostly prefer method 7 and 9.
2. Registration distribution does not vary much among the cities (the reason might be city 1 users dominate the data)

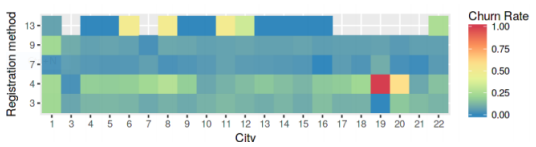


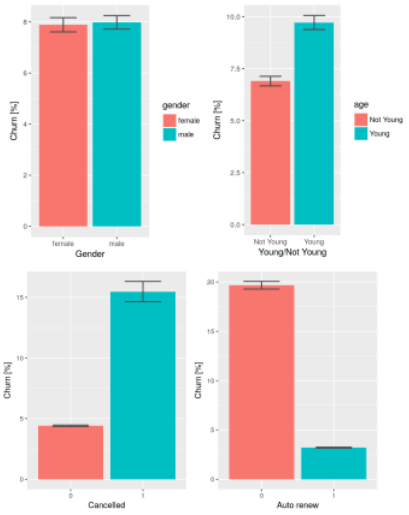
Observe that

1. Monthly subscription is the most popular plan
2. Payment method 41 is the most popular and dominant.
3. There are numerous cases when the actual amount paid and list price is different.
4. The pattern of transaction data fits the monthly subscription picture. Note that an increasing amount of transaction took.

Relationship Between Predictor Variables and Churn Rate

This analysis will provide important foundation for our model building as it gives us an idea of what features are closely related to our prediction.



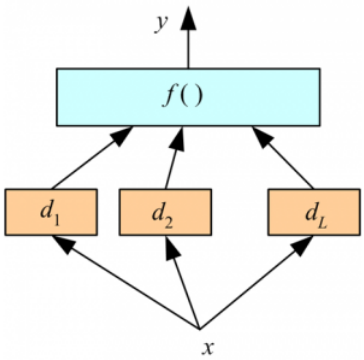


Building Machine Learning Models

Initially we used ROSE(Random Over-Sampling) to deal with binary classification problems in the presence of imbalanced classes.
We made use of stacking modeling techniques. In general, ensembling is a technique of combining two or more algorithms of similar or dissimilar types called base learners. This is done to make a more robust system which incorporates the predictions from all the base learners.

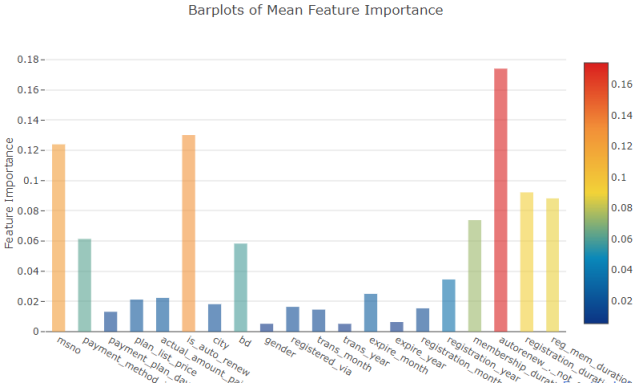
In stacking multiple layers of machine learning models are placed one over another where each of the models passes their predictions to the model in the layer above it and the top layer model takes decisions based on the outputs of the models in layers below it.

Here, we have two layers of machine learning models:



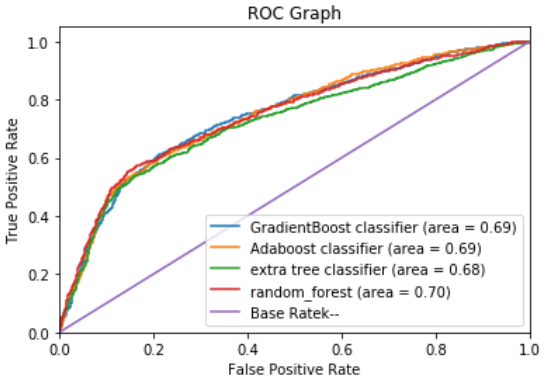
Bottom layer models (d1, d2, d3) which receive the original input features(x) from the dataset.
Top layer model, f() which takes the output of the bottom layer models (d1, d2, d3) as its input and predicts the final output. One key thing to

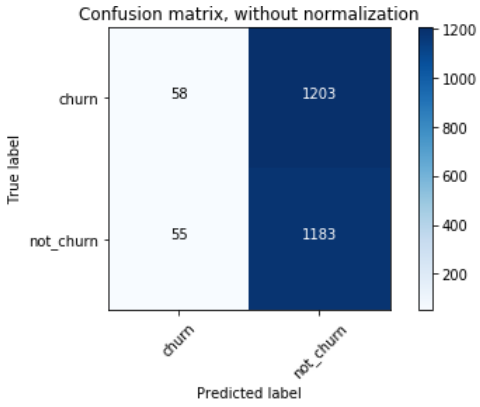
note here is that out of fold predictions are used while predicting for the training data.
Here, we have used only two layers but it can be any number of layers and any number of models in each layer. Two of the key principles for selecting the models:
The individual models fulfill particular accuracy criteria.
The model predictions of various individual models are not highly correlated with the predictions of other models.
One thing that you might have realized is that we have used the top layer model which takes as input the predictions of the bottom layer models. As our first layer models, we used Random Forest, Extratree, Adaboost, GradientBoost.
We have also used a technique called K-fold cross validation, a model-validation technique which is the best way to predict ML model's accuracy. In short, if we choose K = 10,then we split the entire data into 9 parts for training and 1 part for testing uniquely over each round up to 10 times for each of our first layers model.
So for here we took xgboost as a Top layer model which takes the output of the bottom layer models (randomforest, ExtraTree, Adaboost, GradientBoost) as its input and predicts the final output.
Also for the top layer model xgboost, we used hyperopt to hypertune the parameters to have the prediction accuracy from the model.
We found the mean feature importance from the base models.



Evaluation

We used the AUC Score, or Area Under the (Receiver Operating Characteristic) Curve. The Receiver Operating Characteristic (ROC) curve plots all possible combinations of True Positive rate and False Positive rate associated with all the possible classification thresholds. In our context, it visually displays the trade-off between proportion of correctly classified churn observations versus the proportion of misclassified churn observations. The large the area under ROC curve, the less trade-off and the better the model.





Spark for Big Data

Apache Spark is an open-source cluster-computing framework. It was originally developed at the University of California, Berkeley's AMP Lab, the Spark codebase later donated to Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Performance

Spark processes everything in memory. It can also use disk for data that doesn't fit into memory. Spark's in-memory processing delivers near real-time analytics for data from different data sources.
The primary difference between Hadoop MapReduce and Spark MapReduce is that Hadoop MapReduce uses persistent storage and Spark MapReduce uses Resilient Distributed Datasets(RDDs) which are fault-tolerant collections of elements that can be operated in parallel. RDDs can reference a dataset in an external storage system, such as shared file system, HDFS, HBase or any data source offering a Hadoop InputFormat. Spark can create RDDs from any storage source supported by Hadoop, including local file systems. RDDs can be persistent in order to cache a dataset in

memory across operations. This allows future actions to be much faster, by as much as ten times.

Ease of Use

Spark comes with user-friendly APIs for Scala(its native language), Java, Python and Spark SQL is very similar to SQL92, so there's almost no learning curve required in order to use it.
It also has an interactive mode so that developers and users alike can have immediate feedback for queries and other actions. On the other hand MapReduce has no interactive mode.

Our setup

1. We used an Amazon EC2 instance with 2GB of RAM, which acted as master and slave and Jupyter Notebook as our coding environment. The following actions were performed on our instance: SSH into EC2 instance, update all the system packages
2.Installed pip3 using python3-pip - which is Python's package manager.
3.Installed Jupyter Notebook using pip3 install jupyter. Jupyter Notebook is an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and narrative text.
4.Installed Scala because some of Spark's packages come from it
5.Installed py4j using pip3. py4j allows us to access dynamically access Java objects in a Java Virtual Machine.
6. Downloaded Spark from Apache archive and installed in EC2 instance.
7.Installed findspark which is a python module to access PySpark which by default isn't on sys.path.
8.We then defined a configuration file for Jupyter Notebook so that we can access it outside localhost. The configuration file configuration is given below:
c = get_config()
$c.NotebookApp.certfile = u`/home/ubuntu/certs/mycert.pem`$
$c.NotebookApp.ip = `*`$
c.NotebookApp.open_browser = False
$c.NotebookApp.port = 8888$

We then created a key-pair(in our case it is mycert.pem) to access our Jupyter Notebook outside localhost via HTTPS. This can be done by executing the following command on terminal:
sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem

Our results

We used Logistic Regression and Random Forest in spark to build our machine learning models.

We got 0.55 and 0.59 auc score from Logistic Regression and Random Forest models.

Though we have not used robust techniques as compared to sequential coding though there cannot be any comparision between both the approaches.

Conclusion

If we were allowed more time, we would spend more time on several things:
first, we should perform more careful missing value imputations not only using the sample mean. More accurately re-creating the original values will likely enhance our model performance;
second, in our analysis,we did not attempt to detect nor fix extreme values, which could potentially be helpful to model performance;
third, we would aim to gain a more extensive understanding of the hyperparameters the mechanisms behind XGBoost so that we can better utilize this powerful tool.

References

1. ROSE: A Package for Binary Imbalanced Learning by Nicola Lunardon, Giovanna Menardi, and Nicola Torelli
2. Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms James Bergstra, Dan Yamins, David D. Cox§
3. Stacked Ensemble Models for Improved Prediction Accuracy Funda Güneş, Russ Wolfinger, and Pei-Yi TanSAS Institute Inc.
4. Spark SQL: Relational Data Processing in Spark Michael Armbrust†, Reynold S. Xin†, Cheng Lian†, Yin Huai†, Davies Liu†, Joseph K. Bradley†, Xiangrui Meng†, Tomer Kaftan‡, Michael J. Franklin†‡, Ali Ghodsi†, Matei Zaharia† †Databricks Inc. MIT CSAIL ‡AMPLab, UC Berkeley
5. XGBoost: A Scalable Tree Boosting System Tianqi Chen, University of Washington and Carlos Guestrin, University of Washington
6. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems, DAVID J. HAND