Udacity Data Scientist Nanodegree Capstone Project: Sparkify

Author: Benjamin Sassonko

Project Overview

The goal of this project was to predict churn based on user log data of a fictional music streaming service. Depending on how users interact with the service, the resulting model predicts whether that user is likely to churn in the future. This information can then be used for targeted measures (e.g. personalized offers) to keep the users from churning. Since acquiring new users is more expensive than keeping existing users engaged[1], churn prediction and subsequently prevention can significantly improve profitability.

Instead of using libraries like Pandas and scikit-learn, the challenge was to code the whole analysis, ETL pipeline and model pipeline with PySpark. While only a small subset of the data was used for the current state of the project, the code can be used for a larger-scale analysis with the full dataset on AWS, though some optimizations may be required.

Problem Statement & Metrics

The problem at hand is as follows: We have user data from several weeks at our disposal where each user interaction with the service is included. Based on this information, we want to create a model that allows us to predict which users will likely churn in the future when applied to new data. Therefore, we want a model that is good at distinguishing between users that are likely to churn and those that are not. At the same time, the model specificity should be high since we want to avoid too many false positives.[2] A large number of false positives, combined with costly user retention measures would harm profitability.

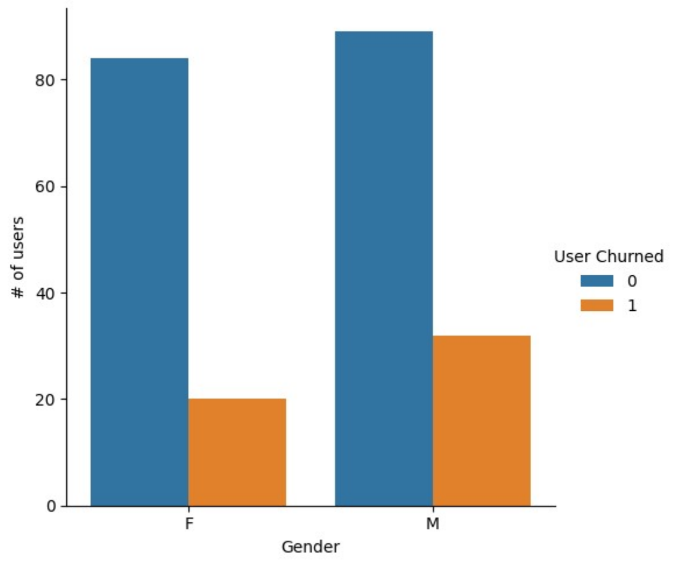
Generally, a good measure for binary classification problems is the Area Under the Curve (AUC). It represents the probability that the model can correctly separate the two classes.[3]

1 https://mode.com/content/the-essential-guide-to-retention/
2 https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
3 https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

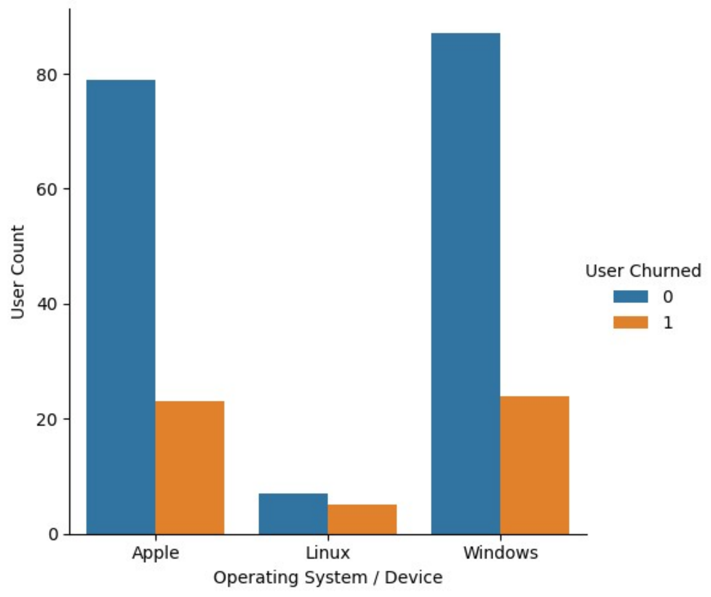
Data Exploration

The small subset of data used for this project contains log data for a total of 225 users. Each row represents a timestamped user interaction with the service (e.g. user registered, user clicked on “next song”). The first interaction in the dataset took place on 01.10.2018 and the last one on 03.12.2018. During this period, a total of 52 users have churned while 173 have not, which results in a churn rate of ~23%. However, this does not mean that some of the users who did not churn in the observed period will not churn in the future. Ideally, we would build the model on the whole dataset since it covers more users and likely a larger time period.

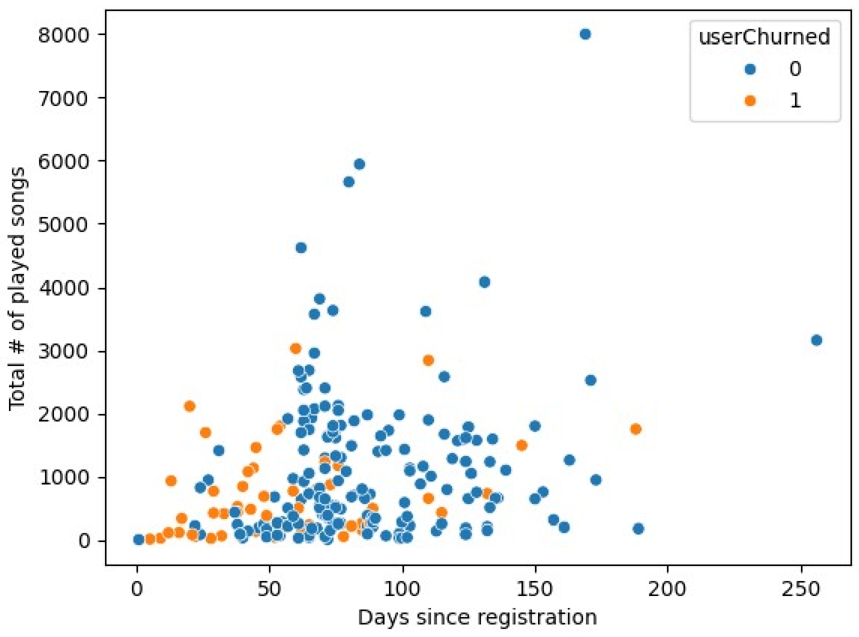
When we explore the data in more detail, there are some anomalies, especially when comparing churned users with those that are still active. For example, the churn rate among male users in our subset is substantially higher than for their female counterparts. With the relatively low number of overall users this may be a sampling issue, nonetheless including gender in the modeling process is reasonable.



Similarly, the churn rate seems to be higher for those people who use Linux as compared to Windows and Apple users. However, due to the low number of Linux users, this may again be purely due to chance.



When we look at the time since the users have registered and their interactions with the service, most users seem to churn within 100 days after registration. Also, users who are exceptionally active and listen to many songs seem to be less likely to churn, especially if they are still very active after more than 50 days.



Data Preprocessing & Feature Engineering

During the data preprocessing, several steps were taken. First, only data from logged-in users was kept since all other data cannot be analyzed with respect to user churn. This also took care of the relevant null values in the data set. Then, the location column was analyzed to subsequently extract the US state of residency for each user. Moreover, the devices used to access Sparkify were separated into three categories (Windows, Apple, Linux). Lastly, the timestamp was converted into the right format to enable further feature engineering and visualizations.

One of the most important preprocessing steps was to define churn. Since all churned users visited the page “Cancellation confirmation”, this interaction was used to engineer the churn flag on the user level. All interactions before the actual churn were flagged.

Apart from features such as gender, location and device, two feature groups were engineered as a proxy for the way users interacted with the service. First, for each user, their personal distribution of clicks on the different Sparkify pages (e.g. Next song, Error, Thumbs up) was added. With these features, it is possible to see things like error occurrence, positive reactions to songs, and interactions with friends. Moreover, the week-over-week changes in listening activity were engineered. For example, if the user listened to 100 songs in week 41 and 110 songs in week 42, this would be represented by 1.1 (a 10% increase). In theory, users who begin to use the service less frequently might be more likely to churn in the future.

Modeling, Metrics & Performance

After engineering all of the relevant features, the categorical ones were indexed and vectorized. Then, all vectors were assembled into a single features column. Since the data is imbalanced with 23% churned and 77% active users, a weight column was added with the inverse values.

For the model itself, two algorithms were selected: The binomial logistic regression and random forest classification. Both approaches are valid for a binary classification problem.[4] Nonetheless, each algorithm has its advantages and disadvantages. While the logistic regression is relatively simple in terms of implementation and training, it does not perform well when the underlying relationships between independent and dependent variables are complex. Moreover, it can be prone to overfitting with high-dimensional data sets.[5] On the other hand, random forest works well with many features and outliers are not as much of a problem. Nonetheless, the interpretation of the results is more difficult and on large data sets, efficient computation poses a problem.[6]

As our main metric for the cross-validation and final evaluation, we use the AUROC (Area Under the Receiver Operating Characteristic). It represents the probability that the model can separate the two classes successfully.[7] Moreover, the area under the precision-recall curve (AUPR) will be used as an additional metric for model quality since we have imbalanced data.[8]

Iterative Results & Hyperparameter Tuning

Due to the class imbalance, the class weight column was used for all models and during the evaluation. The final evaluation of all models was done on 20% of the data, while the remaining 80% were used for training and cross-validation. To ensure consistent results across many iterations, a seed has been used:

data.randomSplit([0.8, 0.2], seed=2)

While there were many smaller changes and experiments during the modeling process, the four key iterations will be discussed in more detail. The first model iteration was done with a logistic regression (LR) and fixed parameters to establish a baseline. This model's performance was unsatisfactory with an AUROC of 0.57. Given the imbalanced classes, we want a model with an AUROC above 0.77, otherwise there is no advantage in comparison to random chance. In the next step, the Random Forest classifier (RF) was tested with default parameters. This model already significantly outperformed the baseline LR model with an AUROC of 0.83. Lastly, cross-validation

4 https://www.kaggle.com/code/kkhandekar/top-10-algorithms-for-binary-classification
5 https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/
6 https://medium.datadriveninvestor.com/random-forest-pros-and-cons-c1c42fb64f04
7 https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
8 https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248

was performed with different iterations of parameter grids. For the LR model, the hyperparameter tuning did not seem to matter much, as the values barely changed. However, the cross-validation for the RF classifier yielded an AUROC value of 0.95. The best performing RF model is shallow (maxDepth = 3) and does not have too many trees (numTrees = 50). The results are summarized in the following table:

Model Iteration	**Parameters / Parameter Grid** **(best CV parameters bold)**	**AUROC**	**AUPR**	**Comments**
LR	All default	0.57	0.47	With values close to 0.5, there is no predictive value in this model.
RF	All default	0.83	0.82	Significantly higher evaluation metrics. RF seems to perform much better in this particular case.
LR CV	• regParam, [0.0, 0.1, **0.2**, 0.5] • maxIter, [**50**, 100, 200] • threshold, [**0.4**, 0.5, 0.6, 0.7, 0.8]	0.56	0.43	Basically the same performance as the default model. Hyperparameter tuning did not improve the model.
RF CV	• maxDepth, [**3**, 5, 10, 15] • numTrees, [10, **20**, 30, 40, 50]	0.95	0.88	Improvement of AUROC and AUPR compared to RF default model.

Discussion & Summary

During the model tuning, the random forest classifier consistently outperformed the logistic regression for both the AUROC and AUPR metrics. The final random forest classifier after cross-validation achieved an AUROC of 0.95. As described before, the AUROC represents the probability that the model can correctly distinguish between the classes.[9] With a value of 95% we can correctly separate users that will likely churn from those that will likely continue using the service. Despite the class imbalance (base chance of correctly selecting a non-churned user = 77%) this provides a significant improvement. The second metric, the area under the precision-recall curve is 0.88 in the final model. A large value attests “high recall and high precision”[10] and values above 0.5 are better than random chance[11]. Conclusively, the tuned RF model performed quite well in its binomial classification task.

Nonetheless, one important thing to note is that depending on the data selected for training, testing and validation with the function *df,randomSplit(seed=X)*, the results sometimes differed significantly (from 0.73 – 0.95 for the AUROC). With a larger dataset, this value would likely be more stable. Still, the achieved values show that the classification is better than the baseline.

During the project, one particularly challenging task was to find a viable proxy for user interaction with the service over time. After some testing, we settled for the week-over-week change in listening frequency. Since there are large differences depending on the weekday (e.g. lower listening frequency on the weekends), daily changes were not included in the model.

9 https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
10 https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html
11 https://www.datascienceblog.net/post/machine-learning/interpreting-roc-curves-auc/

Another challenge was learning to use PySpark for most steps of the analysis (except visualizations). Even though the PySpark course on Udacity provided a good starting point, there were some challenges along the way.

Potential Improvements

Lastly, it is important to acknowledge that the provided model is far from perfect. The model should be trained on a larger set of data to get a better and more stable performance. Moreover, other classification algorithms such as Naive Bayes or a neural network[12] should be tested and compared to the logistic regression and random forest. Also, it may make sense to analyze the importance of the features used for modeling to simplify the models. By removing unimportant information we could reduce noise and improve performance. Moreover, simplifying some of the features may also be a viable path for improvement. For example, including all US states as separate variables caused issues with the random forest classifier. Grouping states together (e.g. by geographical location or based on characteristics such as average income) could be a solution here.

12 https://towardsdatascience.com/top-10-binary-classification-algorithms-a-beginners-guide-feeacbd7a3e2