Capstone Project

Joana Martins dos Santos

Data Science Nanodegree

September 4th, 2024

Predicting user churn for Sparkify

I. Definition

Project Overview

Customer churn is a problem that affects companies in a variety of industries. When a client departs, there is a significant loss to the company[1]. Determining which consumers are most likely to cancel a service subscription based on how they use it is what customer churn prediction is[2]. It identifies which customers are at a high risk of canceling their subscriptions or abandoning the product. Churn prediction allows us to tell whether a customer will leave and why[3]. Data scientists and analysts regularly encounter predicting churn rates in any customer-facing business, and it is a challenging and common problem.

In this project, I predicted churn for a fictional music streaming company, Sparkify. I used realistic datasets with Spark to engineer relevant features for predicting churn and Spark MLlib to build a machine learning model with large datasets, far beyond what could be done with non-distributed technologies like scikit-learn.

Udacity provided the project data in the Udacity Spark course. The original dataset was 12 GB. This project used a small part of it, 128Mb. The project was developed on a local computer using a Spark local session and on IBM Watson Studio using a medium-sized dataset (236Mb).

Problem Statement

This project aims to predict churn for Sparkify. The tasks involved are the following:

1. Load and preprocess the data: check for invalid or missing data.
2. Exploratory Data Analysis: define churn into cancellation, downgrade, and none and explore the dataset.
3. Feature Engineering: Build features to train the model.

[1] EditorJournals and Conferences. (2024, May 18). A REVIEW ON CUSTOMER CHURN PREDICTION USING MACHINE LEARNING APPROACH. https://doi.org/10.17605/OSF.IO/ACNKJ

[2] Userpilot (2024). How to Build a Churn Prediction Model to Predict Customer Churn. https://userpilot.com/blog/churn-prediction/

[3] Userpilot (2024). How to Build a Churn Prediction Model to Predict Customer Churn. https://userpilot.com/blog/churn-prediction/

4. Modeling: Split the full dataset into train and test sets. Test several of the classification machine learning methods. Hyper-tune parameters and evaluate the accuracy of the various models.

The final model is expected to predict churn with the highest score and the best model parameters.

Metrics

I used the accuracy and F1-score metrics for a multiclassification problem in this project. Since the churned users are a small subset, the F1 score is the best metric to optimize the models since it's a harmonic mean of the precision and recall metrics.

These metrics are defined as follows:

Accuracy is the most used metric and is defined as:

$$Accuracy = \frac{(\text{True Positive } + \text{ True Negative})}{(\text{Total Sample Size})}$$

Precision - Correct positive predictions relative to total positive predictions:

$$Precision = \frac{\text{True Positive}}{(\text{True Positive } + \text{ False Positive})}$$

Recall - Correct positive predictions relative to total actual positives:

$$Recall = \frac{\text{True Positive}}{(\text{True Positive } + \text{ False Negative})}$$

The f1-score is defined as the Harmonic mean of precision and recall:

$$F1\ Score = 2 \times \frac{(\text{Precision } * \text{ Recall})}{(\text{Precision } + \text{ Recall})}$$

II. Analysis

Data Exploration

A standard spark session was developed on a spark notebook on an IBM Watson Studio Project for a medium subset (238Mb) of the original data (12Gb). The data was formatted on JSON. The schema provided the following columns:

- root
- |-- artist: string (nullable = true)
- |-- auth: string (nullable = true)
- |-- firstName: string (nullable = true)
- |-- gender: string (nullable = true)
- |-- itemInSession: long (nullable = true)
- |-- lastName: string (nullable = true)
- |-- length: double (nullable = true)
- |-- level: string (nullable = true)
- |-- location: string (nullable = true)
- |-- method: string (nullable = true)
- |-- page: string (nullable = true)
- |-- registration: long (nullable = true)
- |-- sessionId: long (nullable = true)
- |-- song: string (nullable = true)
- |-- status: long (nullable = true)
- |-- ts: long (nullable = true)
- |-- userAgent: string (nullable = true)
- |-- userId: string (nullable = true)

The number of events was 543 705. When checking for invalid and missing data, it was found that there were users without a UserID. These users are probably unregistered or about to log in. At this stage, users without a UserID or gender defined were removed from the database, leaving the database with 528 005 valid events.

I proceeded with the data exploration to find the unique pages, number of unique users by gender, registration year, and user's location. In terms of the site pages, the top 20 pages in alphabetic order are the following:

- About
- Add Friend
- Add to Playlist
- Cancel
- Cancellation Confirmation
- Downgrade
- Error
- Help
- Home
- Login

- Logout
- NextSong
- Register
- Roll Advert
- Save Settings
- Settings
- Submit Downgrade
- Submit Registration
- Submit Upgrade
- Thumbs Down

The most important pages to detect churn are “Cancelation confirmation” and “Submit Downgrade”.

Regarding the users’ profiles, of the 448 users in the database, most (56% or 250) are male. The users are located in several states within the United States of America. The registration date is 2018 for most users and 2017 for a small part.

The song statistics indicate that Kings of Leon was the most listened to artist, played 3497 times. On average, users listen to 24 songs when visiting the home page.

In terms of core metrics, to understand the current state of the Sparkify business, 381 unique users listened to at least one song in the last month of data, and the number of free users (370) exceeded the number of paid users (321).

The churn was determined by considering the users who canceled or downgraded the service. In total, 99 users canceled the service, and 76 were downgraded from paid to free service.

Exploratory Visualization

This section will discuss statistics on how users used Sparkify's service.

Figure 1 shows the number of songs played per hour. Users mostly use Sparkify in the afternoon and evening, with a peak around 4 PM.

Figure 2 presents the average number of songs played per user and average listening time per user daily. The plot shows an upward trend in both the average number of songs played per user and the average listening time. However, a sharp drop is detected in early December.

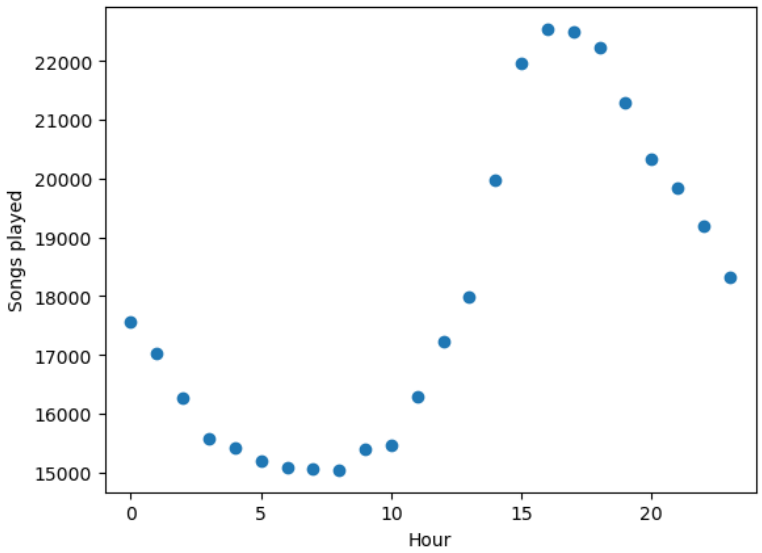

Figure 1 – Number of songs played per hour.

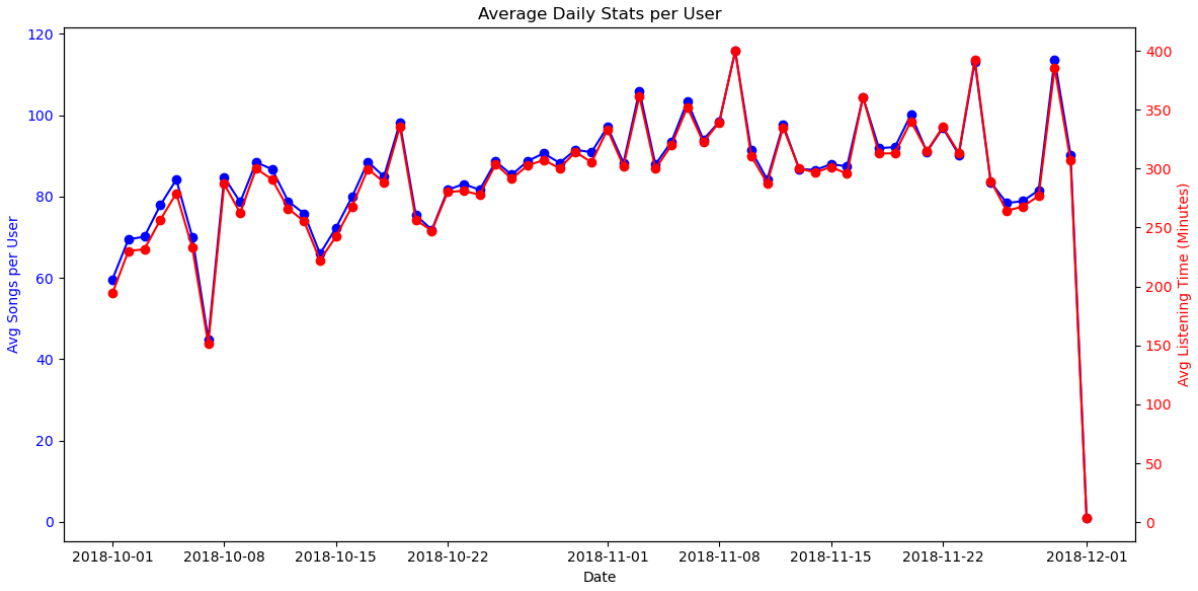

Figure 2 – Average number of songs played per user and average listening time per user daily.

The data's daily active users for the past month are presented in Table 1—the total number and the percentage of the monthly active users. The daily active users help us understand how engaged the users are. Do they casually use Sparkify occasionally, or does the service keep them coming back daily? What we can observe from Table 1 is that the percentage of daily active users had been better in the previous months. Additionally, we can observe a sharp drop in December, from 96 active users on November 30th to 4 on December 1st. This may explain the similar drop in the average number of songs played and average listening time daily observed in Figure 1.

Table 1. Total number and the percentage of the monthly active users for November and early December of 2018.

Date	Daily active users	Percentage of monthly active users
2018-11-01	106	27.7
2018-11-02	110	28.7
2018-11-03	80	28.9
2018-11-05	125	32.6
2018-11-07	111	30.0
2018-11-08	110	28.7
2018-11-10	81	21.1
2018-11-12	101	26.4
2018-11-13	104	27.1
2018-11-15	114	29.8
2018-11-16	105	27.4
2018-11-17	67	17.5
2018-11-21	86	22.4
2018-11-22	68	17.8
2018-11-24	66	17.2
2018-11-26	96	25.1
2018-11-28	96	25.1
2018-11-29	89	23.2
2018-11-30	96	25.1
2018-12-01	4	1.0

Algorithms and Techniques

The dataset was handled using a spark session and spark algorithms to simulate churn processing, analysis, and prediction using machine learning algorithms on a 12Gb dataset. To this end, the algorithms and techniques were limited to the spark environment. A jupyter notebook was used with a spark session in an IBM Watson Studio project to process, analyze, and model the data.

To use the machine learning algorithms, the data needed to be feature-engineered to have these variables in the right form for the machine learning pipeline. Given the multiclass problem of predicting churn regarding cancellation and downgrade, the classification algorithms used were Logistic Regression, Decision Trees, Random Forests, and Multilayer Perceptron Classifier. The Gradient-Boosted Tree Classifier[4] was not used since it is designed for binary classification. All the algorithms were first tested with the default parameters in a spark pipeline for a first score. Then, the CrossValidator was used with hyperparameter tuning to get the best model with the best parameters and highest score.

[4] Sparkitecture, 2024. Gradient-Boosted Trees. https://www.sparkitecture.io/machine-learning/classification/gradient-boosted-trees

Benchmark

In this project, the benchmark used was the Logistic Regression model, a basic model for comparison with the other, more complex models. The metrics obtained with the best parameters for the accuracy and f1-score were 0.51 and 0.36, respectively.

III. Methodology

Data Preprocessing

Considering the problems encountered in section II in the data exploration, the data was checked for invalid and missing data. It was found that there were users without a UserID and gender defined. These users were removed from the database so to be able to use the algorithms.

To predict churn, I selected the variables most related to the user profile: gender, location, and days since registration. To use the machine learning algorithms, the data needed to be feature-engineered to have these variables in the right form for the machine learning pipeline. For the gender, which was F for female and M for male, I converted this variable to a numeric format of 1 and 0, respectively. For the location in text form, I used One-Hot encoding with StringIndexer to transform the location into an indexer. For the registration variable, I transformed the timestamp in days since registration from the current time to give a sense of the effect of time in the churn.

The features needed to be vectorized and scaled to use the variables in the machine learning algorithm, so the VectorAssembler and StandarScaler were used.

Implementation

The processed data was implemented in a spark pipeline with the default parameters for the algorithms Logistic Regression, Decision Trees, Random Forests, and Multilayer Perceptron Classifier. Then, each pipeline was implemented in a spark CrossValidator for hyperparameter tuning, with the same metrics for scoring, accuracy, and f1 score, with the best Logistic Regression model as a benchmark of the chosen metrics. No complications were observed in this process.

Refinement

The initial solution was defined as the best cross-validation model used with the Logistic Regression model. The initial pipeline with default parameters result scores were saved for each algorithm. In the CrossValidator, the required parameters were changed for each algorithm defined in the Algorithms and Techniques section, and the metrics were registered. The algorithm with the best f1-score in the CrossValidator results was considered the best model.

IV. Results

Model Evaluation and Validation

Given the multiclassification setting of this project, with a churn variable predicting cancelation and downgrade, the algorithms used in the spark environment were Logistic Regression, assumed as a benchmark as indicated in that section, Decision Tree, Random Forest, and Multilayer Perceptron Classifier. Each of these algorithms was introduced in a pipeline and the spark CrossValidator was used to tune the hyperparameters selected for each and cross-validate the data three times.

Table 2 presents the parameters used to tune each algorithm, the accuracy, and the f1-score obtained. The Multilayer Perceptron Classifier data refers only to the local spark session due to time concerns in the IBM Watson Studio Project. These parameters were selected based on the tuning parameters normally used and the time available to train the algorithms. The initial score refers to the initial score obtained in the pipeline with the default settings for each algorithm before hyperparameter tuning.

Table 2. Hyperparameters tuned by algorithm in cross-validation and respective metric results.

Algorithm	Parameters tuned	Accuracy	F1-score
Logistic Regression	regParam = [0.01, 0.1, 1.0] elasticNetParam = [0.0, 0.5, 1.0]	Initial: 0.50 Best: 0.51	Initial: 0.36 Best: 0.36
Decision Tree	maxDepth = [2, 5, 15] maxBins = [16, 32, 64]	Initial: 0.61 Best: 0.96	Initial: 0.54 Best: 0.96
Random Forest	maxDepth = [2, 5, 15] numTrees = [50, 100, 200]	Initial: 0.64 Best: 0.95	Initial: 0.58 Best: 0.95
Multilayer Perceptron Classifier	stepSize = [0.01, 0.05, 0.1] blockSize = [64, 128, 256]	Initial: 0.50 Best: 0.50	Initial: 0.33 Best: 0.33

The best model observed in this analysis was a Decision Tree model with an F1-score of 0.96. The Logistic Regression and Multilayer Perceptron Classifier were the worst prediction-scoring models.

The best parameters for the Decision Tree model were the same for both the local and standard session:

- impurity: gini

- maxBins: 64

- maxDepth: 15

- maxMemoryInMB: 256

- minInfoGain: 0.0

- minInstancesPerNode: 1

To evaluate whether the model generalizes well to unseen data, considering already the best parameters obtained from the cross-validation, I made a sensitivity analysis in the local session due to time constraints in the standard session. I varied the size of the test set and observed the difference in the f1-score to see the robustness of the model to data variations.

Reducing the training set size from 80/20 to 70/30 did not affect the f1 score, as it remained at the obtained level (0.99). However, increasing the training set more to 90/10 reduced the f1 score to 0.98. The differences are small and indicate the robustness of the model with varying dataset compositions and a very high prediction score.

A CSV with the predictions obtained in the local session was also obtained.

Justification

The final model achieved a significantly higher f1-score than the benchmark model, defined as the best logistic regression model. The increase in performance, considering the f1-score, was 270%, from 0.36 to 0.96. This solution, which was very similar to the Random Forest model, is the best solution for this problem. The sensitivity analysis also helps to support this case.

V. Conclusion

Free-Form Visualization

In this project, I predicted churn for a fictional music streaming company, Sparkify. I used realistic datasets with Spark to engineer relevant features for predicting churn and Spark MLlib to build a machine learning model with large datasets.

Udacity provided the project data in the Udacity Spark course. The original dataset was 12 Gb. This project used a small part of it, 128Mb in the local Spark project. The project was also developed on an IBM Watson Studio project using a medium-sized dataset (236Mb).

A jupyter notebook was used with a spark session in an IBM Watson Studio project to process, analyze, and model the data. Given the multiclass problem of predicting churn regarding cancellation and downgrade, the classification algorithms used were Logistic Regression, Decision Trees, Random Forests, and Multilayer Perceptron Classifier. All the algorithms were first tested with the default parameters in a spark pipeline for a first score. Then, the CrossValidator was used with hyperparameter tuning to get the best model with the best parameters and highest score.

In Figure 3, we can observe the results of the cross-validation with the metrics used in this project to assess the performance of the models against the benchmark, the logistic regression model.

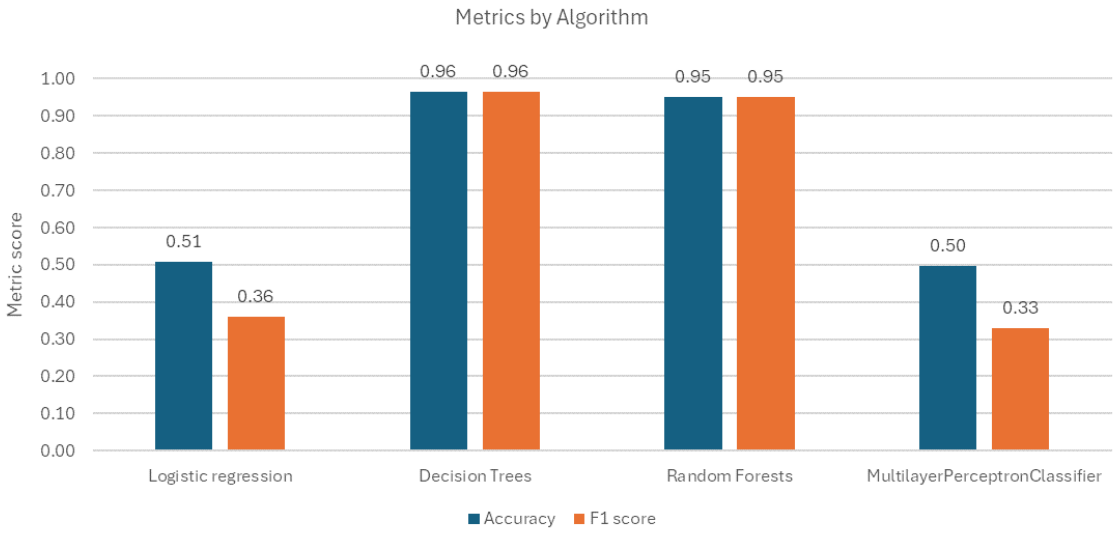

Figure 3 – Metrics obtained by algorithm in the cross validation.

The best model observed in this analysis was a Decision Tree model with an F1-score of 0.96. The Logistic Regression and Multilayer Perceptron Classifier were the worst prediction-scoring models.

To evaluate whether the model generalizes well to unseen data, considering already the best parameters obtained from the cross-validation, a sensitivity analysis was where the size of the test set was varied, and the observed difference in the f1-score allowed to see the robustness of the model to data variations.

Reflection

This project aimed to predict churn for Sparkify. In jupyter notebooks, both on the local session and a standard session on the IBM Watson Studio project, I first loaded and cleaned the data by checking for invalid and missing data. Then, I proceeded to exploratory data analysis, where I defined churn into cancellation, downgrade, and none and explored the dataset. To build the models, I first had to do feature engineering to properly configure my chosen features (gender, location, and days since registration) to train your model. In the modeling phase, I split the full dataset into train and test sets and tested several classification machine learning methods using cross-validation with hyperparameter tuning. I evaluated the accuracy and f1 score of the various models to select the best model with the best parameters. The final model aimed to predict churn with the highest score and the best model parameters.

Some interesting aspects of this project for me were working in the spark environment and the fact that the Decision Tree model performed better than the Random Forest model, which is a boosted version of the Decision Tree model and would be expected to perform better.

The difficult aspect for me was to work in the spark environment, which is somewhat different from the scikit-learn environment. However, it was a good challenge.

This solution was the best one for the problem at hand. However, each churn problem has its own data, and generalizing this solution to other similar problems is not a good option.

Improvement

I used the available spark classification models for big data in this project. Potential improvements would be to test more hyperparameters that I did not test for time constraints. Using a script instead of a notebook could also improve to have a more general solution.