Sparkify Music Service Churn Prediction Project with Spark

Project Definition

Project Overview

In this project, we will attempt to identify the users that potentialy will churn from our music service. Sparkify (similar to Spotify) is a service with a subscription-based model where each user can try out the free-tier plan or use a specific monthly-based subscription plan. The users/customers can cancel their subscription at any time, so it's critical to to be able to identify those users in time and make an attempt to keep them using our services.

Problem Statement

Customer retention is one of the primary growth pillars for products with a subscription-based business model. In the current market the customers are free to choose from plenty of providers of the same service. There are many factors that drive the customers to churn and that could lead to both material losses and damage the damage to the reputation would be significant.

The project's goal is to provide an answer to the question ‘***’Which users are at risk to churn***” by identifying the users before they churn, Sparkify can engage with them and make them a more appealing offer and keep the customers active. That can potentialy reduce the costs of acquiring new customers.

By identifying high risk users, the service provider can proactively engage with them in order to keep them active instead of spending money to get them back or aquire new ones.

Strategy

Pre-processing, cleaning & EDA

A basic pre-processing and an analysis of the data is performed. From that initial step, a basic data cleaning is performed e.g. null value investigation and handling and questions like which customers churn the most, what is their gender, from which country, do they have premium or trial membership, etc. are answered. The goal was to gain a better understanding of the customer's behaviour and the data.

Feature Engineering

In the second step, based on the analysis in the previous step some new features were created (feature engineering). e.g. the user's state, avg_songs_per_day, time feature transformation, etc.

Pipeline Creation & Experimentation

The next step includes the ML pipeline creation and experimentation. In this stage, all the necessary data transformation steps are integrated into one pipeline. e.g. string indexer, encoders, assemblers and scalers.

ML Hyperparameter tuning and cross-validation

The hyper-paramter tuning step includes the process of finding the near-optimal model parameter candidates using data cross-validation. The grid-search and the BinaryClassificationEvaluator emthods were used to identify which pipeline/model performs the best in terms of performance metrics.

ML Pipeline Performance Evaluation/Metrics

The model's performance is evaluated using the unseen subset of the data e.g. test set. In order to have a more holistic view of the pipeline's performance the confusion matrix, accuracy, precision, recall and f-beta score are used. However, due to the highly unbalanced data only the confusion matrix and the f-1 score are used in order to evaluate the model's performance.

Flask-Based Web App

Finally, an app is built where someone can enter the information about the user and get a classification of whether or not that user will chrun or not along with a probability estimate based on the model prediction.

Finally, we built a **web app** using **Flask** web framework on the back-end and **Bootstrap** CSS framework on the front-end.
In the web app, you can **enter the information about the user** and **get her classification as a customer who is likely or not likely to churn** based on the model prediction.

Metrics

Typically, the metrics upon which we will assess the performance of the trained model are chosen based on the business context. More specifically, the typical objective is to minimize close to zero (not completely zero – crystal ball) the FP and FN. Also, we must consider the fact that the classes in our dataset are highly imbalanced. Usually, when we work with imbalanced datasets we try to choose between precision and recall. In this business context, it is wise to balance between those two metrics and as a result the F1 score.

In simple terms, F1 score is the harmonic mean of **Precision** ('out of all customers who were labeled as "churned," how many did we correctly label as such?') and **Recall** ('out of all customers who were labeled as "churned," how many actually churned?'). Ideally, we wouldn't care so much about the FP, meaning the model would identify a customer will churn whereas this specific customer would not churn. On the other hand, we care about the FN, meaning that the model would classify a customer as non-churned whereas the customer will cancel their subscription.

Analysis

Data Exploration and Visualization

As input data we tried the following datasets that contain Sparkify music events:

- Mini-dataset file (128MB) `*mini_sparkify_event_data.json*`
- Full dataset available (12GB) `*s3n://udacity-dsnd/sparkify/sparkify_event_data.json*`

Description of columns available in both datasets.

```
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)
```

In this end, we used just a sample (10%) of the mini dataset, because it was very computationally expensive to train/experiment with a model both in the IBM Watson and the Udacity platform. The **mini-dataset file,** which is a small subset of the full available dataset for **data exploration**. The raw dataset contains **286,500 events** and **18 features**.
All the details of the analysis are reported on the Jupiter Notebook `Sparkify.ipynb`.

Example of a row.

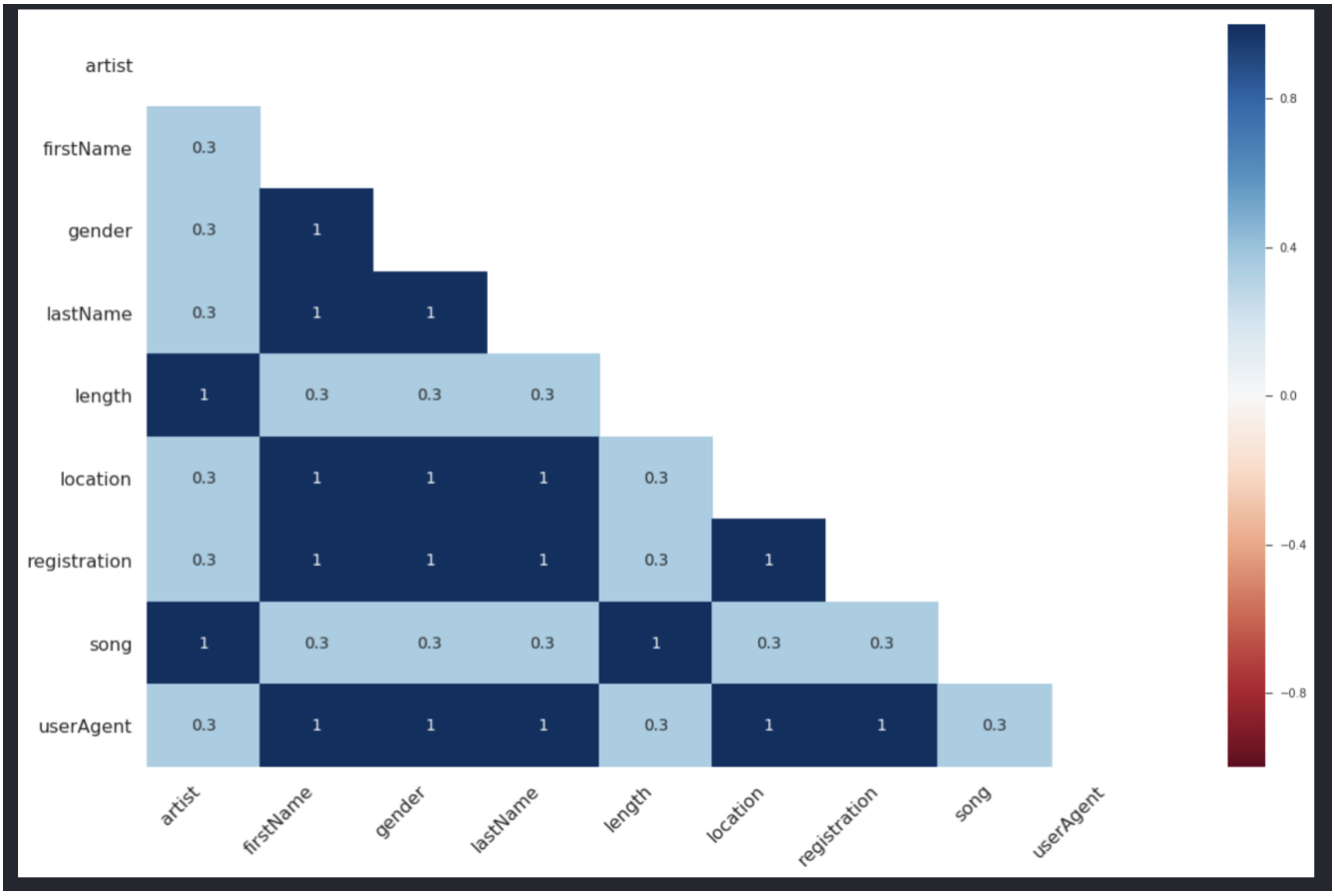
```
Row(artist="O'Rosko Raricim", auth='Logged In', firstName='Madison', gender='F', itemInSession=74, lastName='Morales', length=90.56608, level='paid', location='Tampa-St.
Petersburg-Clearwater, FL', method='PUT', page='NextSong', registration=1536287099000, sessionId=222, song='Terre Promise', status=200, ts=1538384924000,
userAgent='"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36"', userId='25')
```

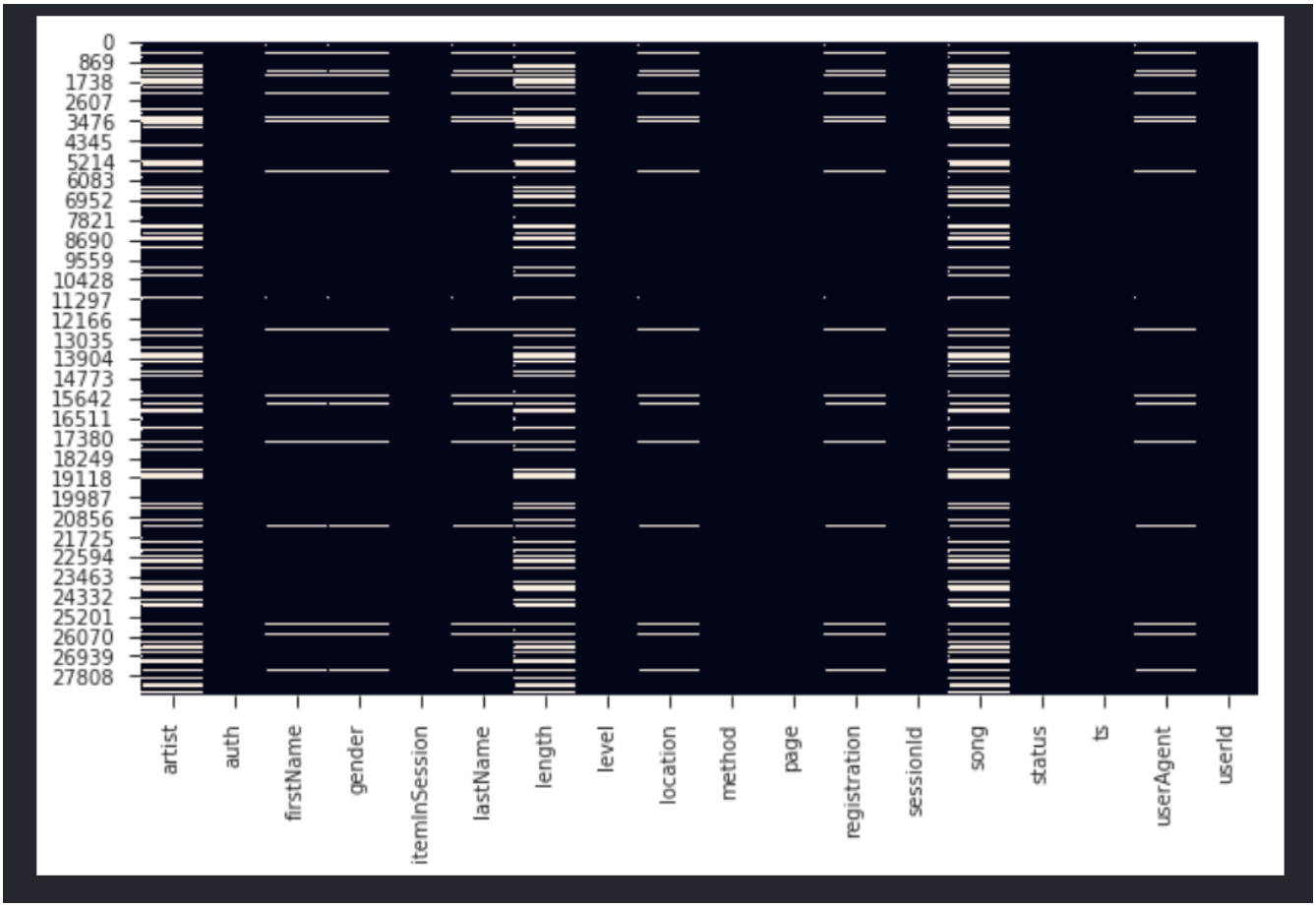
Most of the **events** in the dataset happened between **October** and **November** 2018. The **registrations at the service** in the dataset happened during **March-November** 2018.

Missing values

We explored each column of the dataset, and we found the following **number of missing values**:

	Null Values
artist	6007
auth	0
firstName	872
gender	872
itemInSession	0
lastName	872
length	6007
level	0
location	872
method	0
page	0
registration	872
sessionId	0
song	6007
status	0
ts	0
userAgent	872
userId	0


artist
firstName
gender
lastName
length
location
registration
song
userAgent
0.3
1
0.8
0.4
0.0
−0.4
−0.8



From the above counts and visualizations, we observe that:

1. There is missing information for users not logged in, corresponding to empty ***userID***, ***firstName***.
2. There are a lot of missing values in *song, length, artist* columns which correspond to the events where the user is not listening to music.
3. There is a correlation between most of the missing values in the investigated columns. For example we can see that every time there is a missing value in the firstName there is also a missing value in the gender (1-on-1).

We decided to **focus only on data about users that were logged in** and hence remove rows corresponding to empty ***userId*** or ***firstName***. Indeed, records without any useful information about the user are not useful in predicting customer churn.

Churn definition

In the specific context we consider users as churned when they cancel their subscription from the

service. The event from the ***'Cancellation Confirmation'*** page, are the events that are used to define churned users.

EDA and Data Visualization: Churned vs Stayed

Does the gender affect the user churn status

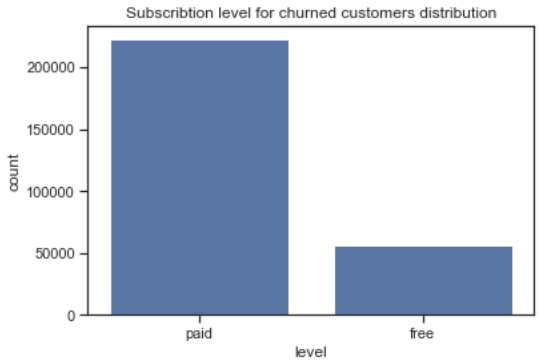
There are 120 male and 103 female users in the dataset. 60.0 of users who churned are male. 53.52% of users who are active are male. 46.47% of users who are active are female. Among female users 3.88% churned. Among male users 5.0% churned.

Comparing female and **male users** we observe that:

- There are **more male users** (121) than female users (104).
- **Most of the users** who have **churned** are **male** (62%).
- Among **female** users **19% have churned**, while among **male** users **26% have churned**.
- **Female users** seem to be **more active**. They have generated **56% of the events**.
- **Female users who churned generated fewer events** than male users who churned.

Does the subscription plan (free vs paid) affect the user churn status?

There are 165 paid and 195 free plan accounts in dataset. The 43.90% of churned users were on a paid plan and 46.40% of active users are on a free plan.

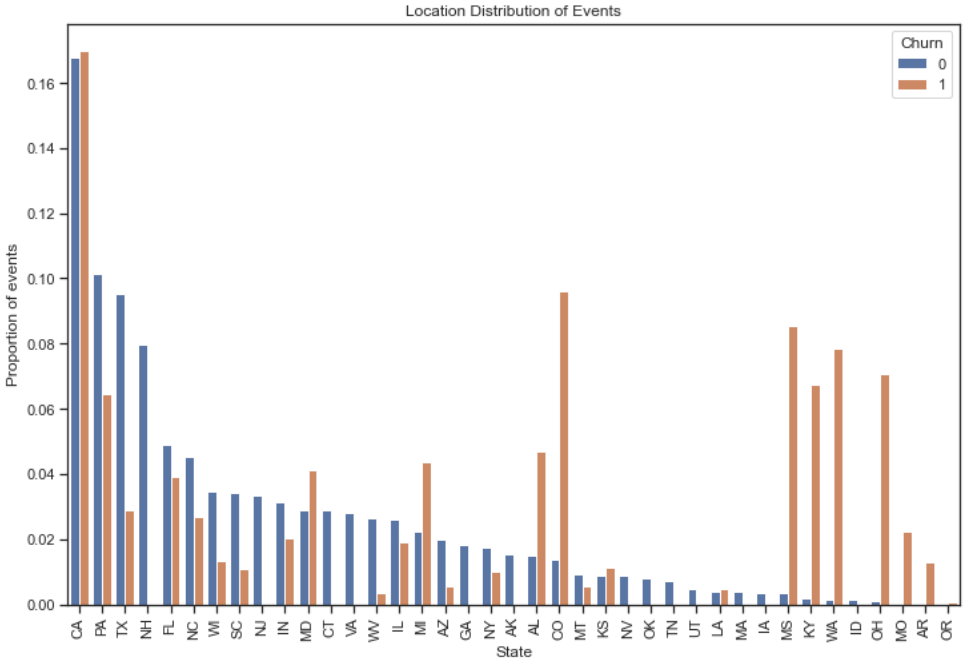


From the investigation below we found that:

- The total number of paid and trial accounts exceeds the total number of unique users registered in the platform. This may be because of service change (upgrade or downgrade).

From the above we see that **users who paid for their subscription** are **more active** than those using the free plan, for active users and those who churned.

Does user location affect churn status?



- Overall, **there appear to be differences in the distribution of active and churned users across states**. In addition, there are some states that have no churned users (e.g., Connecticut and New Hampshire), and some states that have only churned users (e.g., Arkansas, Missouri, Oregon).
- **California, Pennsylvania, Texas, Florida have the most users and events** for active and churned users.

- It appears to be differences in the distribution between active and churned users across states. In addition, there are some states that have no churned users (e.g.New Hampshire), and some states that have only churned users (e.g., Arkansas, Oregon).
- California, Pennsylvania, Texas, Florida have the most users and events for active and churned users.

Methodology

Data Preprocessing

Before proceeding with the implementation of machine leaning models, we preprocessed the raw dataset following these steps:

1. **Remove rows where the user was not logged in** corresponding to missing userId, firstName, type casting ,etc., and duplicates values.
2. **Define churn as 1 if the user cancel her subscription** as per '*Cancellation Confirmation*' page event and 0 otherwise.
3. Create a new dataset which contains the features extracted for each user.

To predict whether the user is going to churn, we extracted the following features:

- **Gender**: EDA showed that male and female users may have a different behavior.
- **Days since registration**: from EDA it appears that active users have been registered onto the platform for a longer time than churned users.
- **Last state**: state in which the most recent session is recorded.
- **Average songs played per day**: to see how much the user is active on the platform.
- **Last level**: the most recent user subscription plan.
- **Thumbs up proportion** (ratio thumbs up over thumbs down): to see the level of appreciation.
- **Number of add a friend events**: to see the level of engagement.
- **Average roll adverts per day**

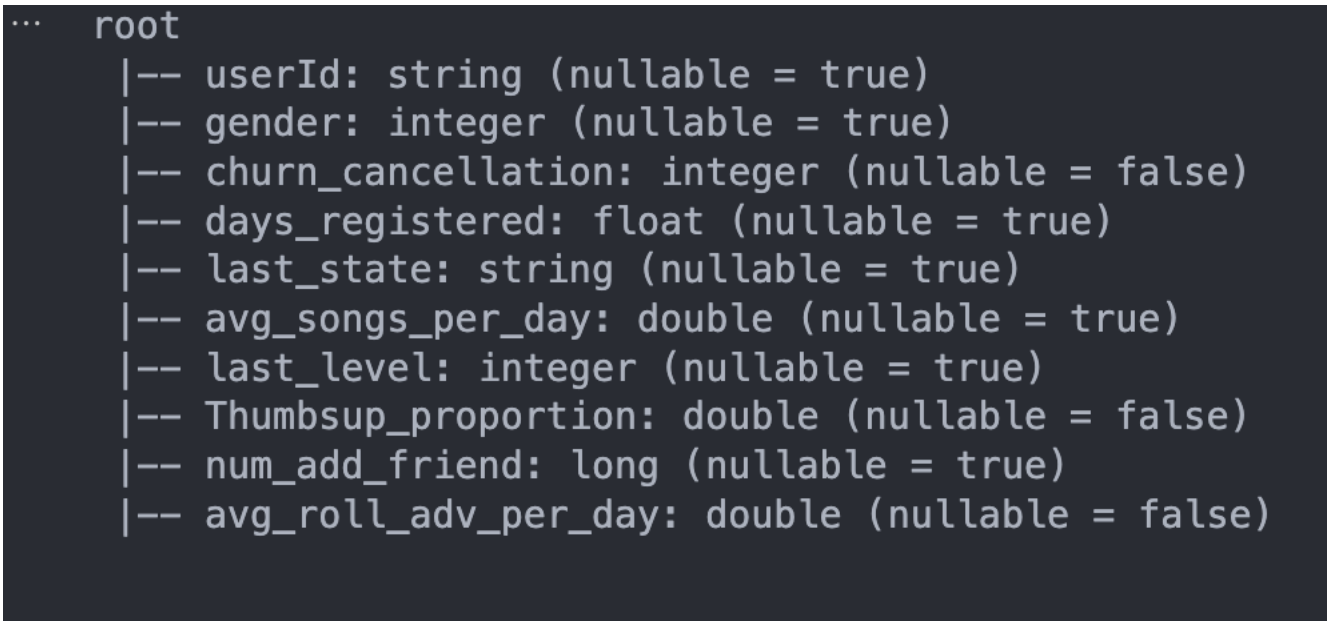
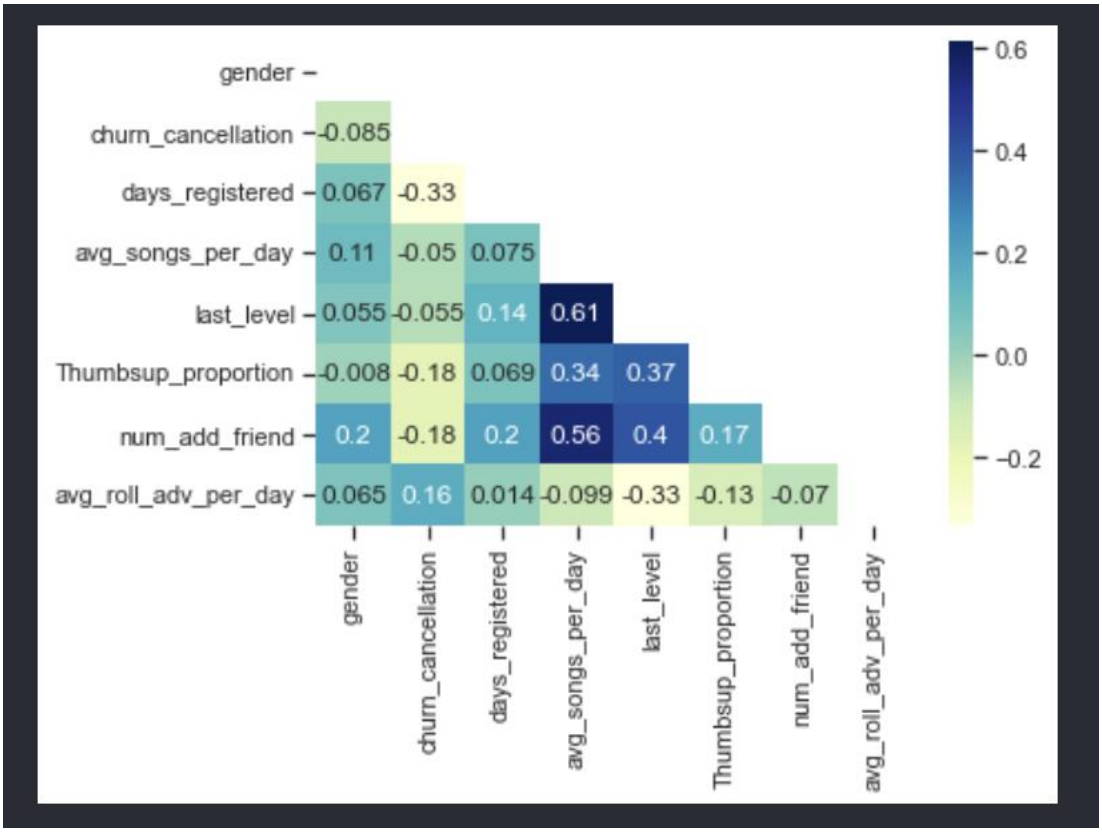


Figure 1 - Final Features

The correlation with the target variable and the rest of the features is explored to check for multi-collinearity. Multi-colinearity affects linear models like Logistic Regression but doesn't affect models like decision trees or specific versions of logistic regression like ridge or lasso regressors.

Correlation among features



Implementation

The data analysis and model building with PySpark and SparkML the entire subset and then, the implementation was run in the IBM Watson studio clusters.

Implementation steps:

1. The goal was to train/hyper-tune and evaluate several different ML models inside the pipeline, however the resources were very limited and only one kind of model is used.
2. Develop a flask-based web app to demonstrate the classification of users based on the underlying model chosen to make the predictions.

Machine learning pipelines

Preparation

- Dataset preparation (pre-processing)
- Dataset train/test splits (80% - 20%),

ML pipeline to predict user churn that consists of:

- String indexer and encoder for categorical features.
- Features assembler.
- Scaler: we use *MinMaxScaler()* which transforms each column value into the range [0,1], preserving the shape of the data.
- Classifiers: Gradient-Boosting Tree.

Evaluation metrics

Given the business context and the class imbalance, the performance of the models was evaluated using the metrics accuracy, precision and recall with emphasizing on the F1 score as the metric. The reason is that during class imbalance high model accuracy, may be due to the fact that the model is biased towards the majority class (‘not churned’, 0). In that case, the model does not serve our problem at all, which is to identify users who are likely to churn.

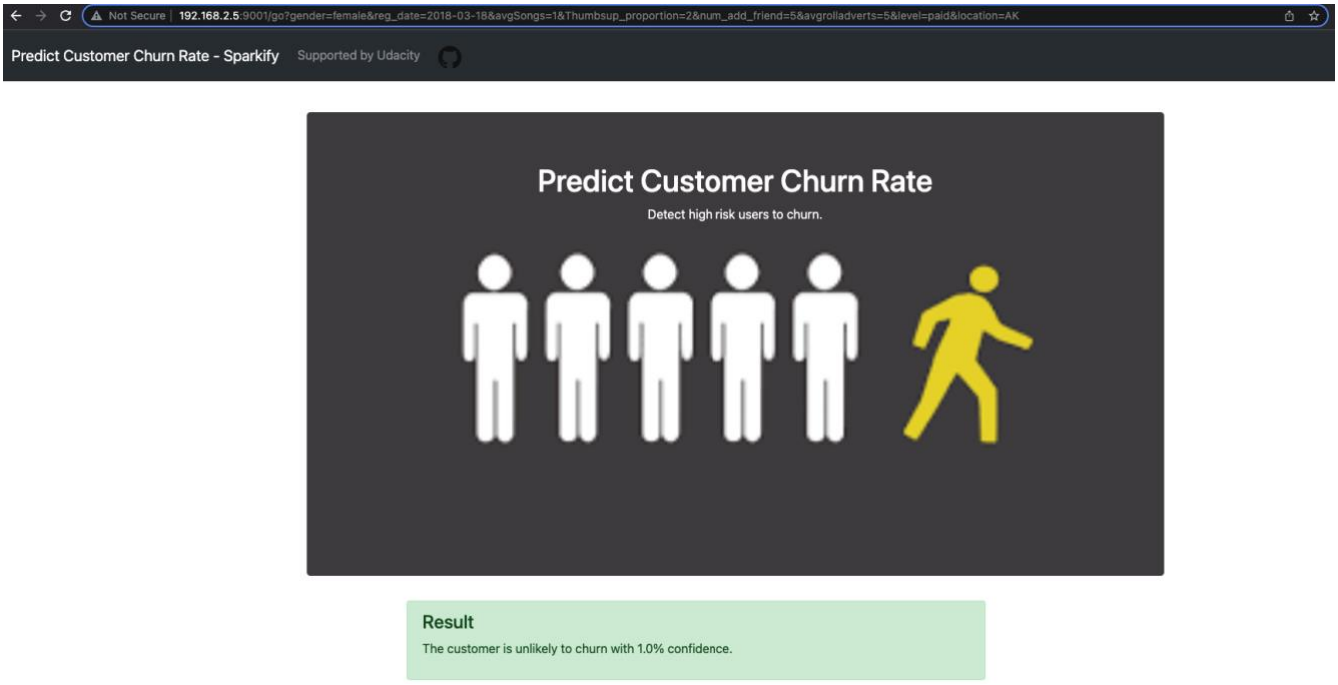
To calculate the performance metrics a function was developed (accuracy, precision, recall, F1), that also extracts the confusion matrix.

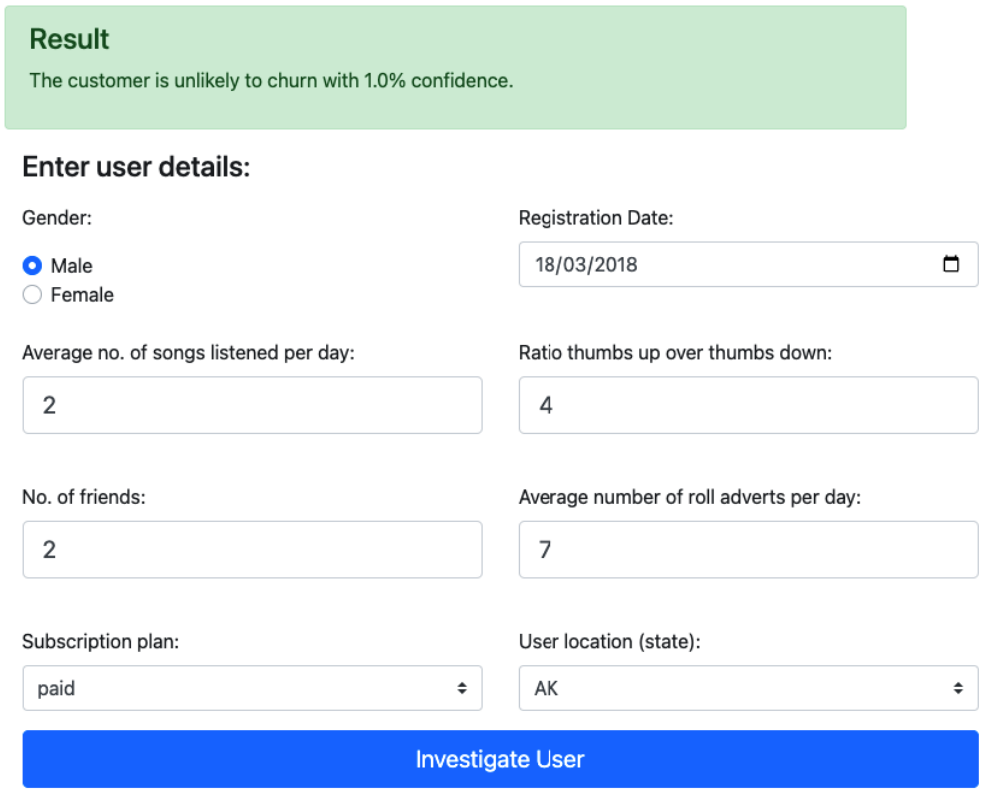
Web application

A flask-based web app has been implemented with html-bootstrap on the front-end.

The flask app includes the following artefacts:

- *run.py* that contains the logic of the application. It handles the queries and display results on the web app.
- Folder templates:
  - *master.html* controls the main page
  - *go.html* controls classification result page
- Folder static:
  - githublogo2.png is the github logo used in the main page
  - churn_image.png is the background image downloaded from (https://www.shutterstock.com)
- Folder cvModel_gbt.mdl contains all the files of the model that we chose to make churn predictions.




Result
The customer is unlikely to churn with 1.0% confidence.
Enter user details:
Gender:
Male
Female
Registration Date:
18/03/2018
Average no. of songs listened per day:
2
Ratio thumbs up over thumbs down:
4
No. of friends:
2
Average number of roll adverts per day:
7
Subscription plan:
paid
User location (state):
AK
Investigate User

Results

Model Evaluation and Validation

Results obtained running the ML models using the full available dataset (12GB)

- **Models with cross-validation and parameter tuning**

Classifier (best parameters)	Accuracy - train	Accuracy - test	F1 - train	F1 - test
Gradient-boosted tree	0.85	0.84	0.61	0.61

Justification

The reason for choosing Gradient Boosted Trees is because this algorithm belongs to the ensemble methods and is famous for performing very well in tabular datasets and is generally stable and less prone to overfitting.

Conclusion

Reflection

The project's goal is to provide an answer to the question '**Which users are at risk to churn'** by identifying the users before they churn, Sparkify can **engage with them** and make them a more appealing offer and keep the customers active. That can potentialy reduce the costs of acquiring new customers.

The solution has the following steps:

- Data cleaning and characterization.
- Exploratory data analysis to understand the data at hand, and possibly extract useful information on the customer base in terms of demographic and interactions with the service.
- Feature engineering to identify features that are useful in predicting churn.
- Binary classification modeling with parameter tuning and cross-validation: Gradient-boosted tree.

- Model evaluation considering (F1 Score) the highly imbalance dataset at hand.
- A flask-based web app where someone can insert user's information and get an idea of whether the user will churn or not along with a probability estimate.

The most challenging part of the work on this project was the feature engineering and model training. Hand crafting features requires domain knowledge, and it wasn't an easy process.

Improvement

- Spend more time on hyperparameter tuning.
- Find another way to deal with unbalanced data.
- Couldn't use the ppscore package - Power Predictive Score is a bivariate analysis
- Experiment with oversampling or undersampling