Udacity Capstone Report

Brandon Nutting

**Initial Project Overview:**

I decided to select this project due to my personal interest in detecting churn as well as learning pySpark. This project is focused on predicting churn. For this project the definition of churn is when a user visits the cancellation confirmation page.

**Initial Steps:**

This dataset contains no missing values for sessionId or userId. However, be careful as after checking for the empty string in sessionId or userId it shows there are 8346 values. These values need to be dropped, as we need to be able to identify users.

**Exploratory Analysis**

First, I decided to look at the total number of users who churned, after performing the UDF to identify churn it is shown only 52 users in the dataset have churned, something that should be noted as that is an extremely small sample size. Within the churned users, the gender distribution is relatively close, 32 vs 20. I then looked at the average songs played per hour between churned and non churned users, which were pretty close too (.811 (Churn) vs .821 (Non-Churn)). Looking at churn per hour the only real insight that can be gathered is that most churned at hour 7. Something I wasn’t really expecting I would have initially thought most users would have churned later in the night.

I then looked at the number of help requests between churned and non churned users. This resulted in 239 vs 1215 for churned vs non-churned users. This is a huge difference, however 239 help requests is pretty high for only 52 users who churned. These findings helped me move towards the features I decided to generate.

**Feature Engineering:**

The following features were generated [Thumbs Up Per Day, Thumbs Down Per Day, Errors Per Day, Help Visits Per Day, Number Of Friends, Average Songs Played Until Home Visited, Average Songs Played Until Error Encountered, Average Songs Played Until Help Needed]. I thought these collection of features would give us the most information for our model to ingest.

For scaling I decided to use MinMaxScalar. This would be a point where some experimentation could be done. Different Scaling techniques could be investigated to see how performance metrics change.

**Modeling And Results:**

I first tested a logisticRegression model. This model performed pretty well out of the box, correctly predicting 15/16 out of our testing set. The F1 score for this model was .93.

I then tried using CrossValidator to experiment using different parameters. I didn’t find any increase in accuracy using this step so I started investigating different options for classification models.

This led me to test the RandomForsestClassifier, which resulted in a perfect 16/16 correct predictions for our testing set. The f1 score was a perfect 1.0.

**Improvements:**

There are many options for improving this project. The first that comes to mind would be increasing the number of churned users in the dataset. One potential way would be to use the full dataset hosted on AWS, this should increase the number of churned users. The next idea I have for improving this model would be to experiment with different parameters with the RandomForestClassifier, since this was the best model.