Capstone Report for Udacity's Data Scientist Nanodegree

By: Ruben D. Nieves
March 31, 2021

Definition

Project overview

This project is about utilizing Apache Spark to perform Data Science tasks on the 'Sparkify' dataset. The Sparkify dataset is a hypothetical user log coming from a music-listening service such as Spotify or Apple Music. The high level goal of the project is to create a prediction model capable of identifying users that are considering to drop their subscription to the music listening service. The music-listening service could utilize the results of the predictive model to send targeted offers in hopes to preserve the user subscription.

Problem Statement

The core technical problem of this project is to find behavioral patterns in the available data, particularly for users that are considering to drop the service. To find these patterns, data science techniques such as data cleaning and data normalization need to be applied. A set of suitable features (numerical representations of user behavior) need to be identified or calculated based on available data. Then, machine learning algorithms such as LogisticRegression and RandomForestClassifiers are to be trained on these features. Finally, the trained models are utilized to make predictions from the stream of user activity logs and determine weather a user may be considering to drop his/her subscription.

The general strategy to solve the core problem lies on applying good data science techniques to clean the data and to later utilized this processed data to discover patterns in the data are the most indicative of an upcoming 'churn', or a dropped subscription. The discovered patterns are turned into feature vectors that can be fed to a variety of machine learning algorithms with the end goal of classifying a user as a potential churner or no.

An additional challenge of this project is the large amount of data that needs to be analyzed. This problem will be tackled by utilizing Apache Spark as the main underlying framework to process the data. Apache Spark is a distributed processing framework specifically designed to handle Big Data.

Metrics

The following metrics will be utilized to measure performance of the predictive model implemented for the capstone project.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions\ Made}$$

Considering a potential class representation imbalance, The F1 Score metric will also be utilized. An accuracy metric on its own could be misleading. It could potentially hide poor performance on classifying members with low representation. For instance, if all images for a given class X are wrongly classified, it may

not affect the accuracy metric if the class constitutes a small percentage of the overall training or validation data.

In order to compute F1 Score, two additional metrics, namely Precision and Recall, need to be computed. Explanations of what these metrics are and how they are computed is provided below.

- True Positive: When the model is given an input image of class X, the model classifies it as class X
- True Negative: When the model is given an image of a class that is NOT X, the model correctly labeled as NOT X
- False Negative: When the model is given an image of a class X, the model erroneously assigned it a NOT X label.
- False Positive: When the model is given an image of class NOT X, the model incorrectly labeled as class X.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Analysis

Data Exploration

In this section, characteristics, anomalies, features and statistics of of the input data are discussed. For this project, the dataset provided by the Udacity staff on the 'Sparkify Project Workspace (mini_sparkify_event_data.json) is utilized.

The input data is a stream of user-triggered events. Trough its various attributes, data scientists could track the activities of all users within the platform. Activities or states include: Song being listened by user, whether the user like or dislike a song, or what page within the platform has the user visited. The following table includes a list of the attributes of an event, along with their high-level description.

Table 1 Description of attributes in input data

Attribute	Description
`artist`	Artist of the song being listened to
`auth`	Whether is logged in or no
`firstName`	Name of user
`gender`	Gender of user
`itemInSession`	Count of events within a session (since last logged in)
`lastName`	User last name
`length`	Seconds between events
`level`	Whether user is on free tier or paid subscription
`location`	City and state from which user is triggering events
`method`	Attribute of the html message triggering the event

page	A label identifying the content a user is interfacing with at the time of the event
registration	[Unknown]
sessionId	A session idntifier
song	Title of song being listened to
status	Attribute of the html message triggering the event
ts	Time stampt of a message on unix/posix time (milliseconds)
userAgent	Identifier of the internet browser being utilized by the user at the time of the event
userId	Identifies the user

Of particular interest are the userId, level, page and ts. This set of attributes could be leveraged to create activity patterns for each user as a function of time. The patterns could be further associated (or correlated) with the event of dropping a paid subscription and should support the construction of predictive models.

A total of 22 different event (pages) are present on input data.

```
['Cancel', 'Submit Downgrade', 'Thumbs Down', 'Home', 'Downgrade', 'Roll A
dvert', 'Logout', 'Save Settings', 'Cancellation Confirmation', 'About', '
Submit Registration', 'Settings', 'Login', 'Register', 'Add to Playlist',
'Add Friend', 'NextSong', 'Thumbs Up', 'Help', 'Upgrade', 'Error', 'Submit
Upgrade']
```

The Data Visualization section provides additional insight into the distribution of this events.

The analysis of these events will focus on pages that are the strongest indicators of user engagement. This project is based on the premise or hypothesis that the following activities are the stronger indicators of user engagement:

- Thumbs Up
    - Indicator of user liking the content the platform provides.
- Thumps Down
    - Indicator of user disliking content the platform provides.
- NextSong
    - Indicator of overall activity in the platform
- Cancellation Confirmation Or Downgrade Or Submit Downgrade
    - Indicator of when a user churned the platform, or when it is considering doing it.

The Data Visualization section provides a sequence that illustrates the behavioral pattern that could be observed before a user drops the service subscription.

The original dataset contains a total of 286500 records spanning from October 1, 2018 through December 3 of 2018. It contains interactions of a total of 226 users. However, initial exploration of the data demonstrated the presence of null values and/or empty usernames. These values were removed from the dataset, and the approach to do so is discussed on the Methodology section.

Data exploration also demonstrated the existence of users that never subscribed to the platforms paid service. Keeping these users posed the risk of misleading the machine learning algorithms, as their activities

could not be correlated with dropping the paid service. On view of this, records for users that never subscribed to the service were also removed from the data frame.

One of the key steps on the data exploration phase of this project was to identify which users within the dataset became 'churners', or users that dropped their paid service. To meet the goals of this project, the machine learning model will be trained to recognize between the churners and non-churners classes.

The 'Canelation Confirmation' was identified as the page (or event) associated with dropping the service subscription. By analyzing the 'Cancelation Confirmation' events, a total of 76 'churner's and 89 non-churners were found. To achieve class representation balance during model training, additional steps will be required. The steps will be discussed in the Methodology section.

Data Visualization

The following table shows the distribution of event types

```
Distribution Event Types
+-------------------------+------+
|page                     |count |
+-------------------------+------+
|Submit Registration      |5     |
|Register                 |18    |
|Cancel                   |52    |
|Cancellation Confirmation|52    |
|Submit Downgrade         |63    |
|Submit Upgrade           |159   |
|Error                    |258   |
|Save Settings            |310   |
|Upgrade                  |499   |
|About                    |924   |
|Settings                 |1514  |
|Help                     |1726  |
|Downgrade                |2055  |
|Thumbs Down              |2546  |
|Logout                   |3226  |
|Login                    |3241  |
|Roll Advert              |3933  |
|Add Friend               |4277  |
|Add to Playlist          |6526  |
|Thumbs Up                |12551 |
|Home                     |14457 |
|NextSong                 |228108|
+-------------------------+------+
```

The plot of events shown on Figure 1 has activity for a specific user several days before downgrading the subscription. In the graphic, the various events are highlighted with different symbols and plotted against time. The Y axis represent the numeric label assigned to the event (or page), which simplifies visualization by spreading events on the Y axis.

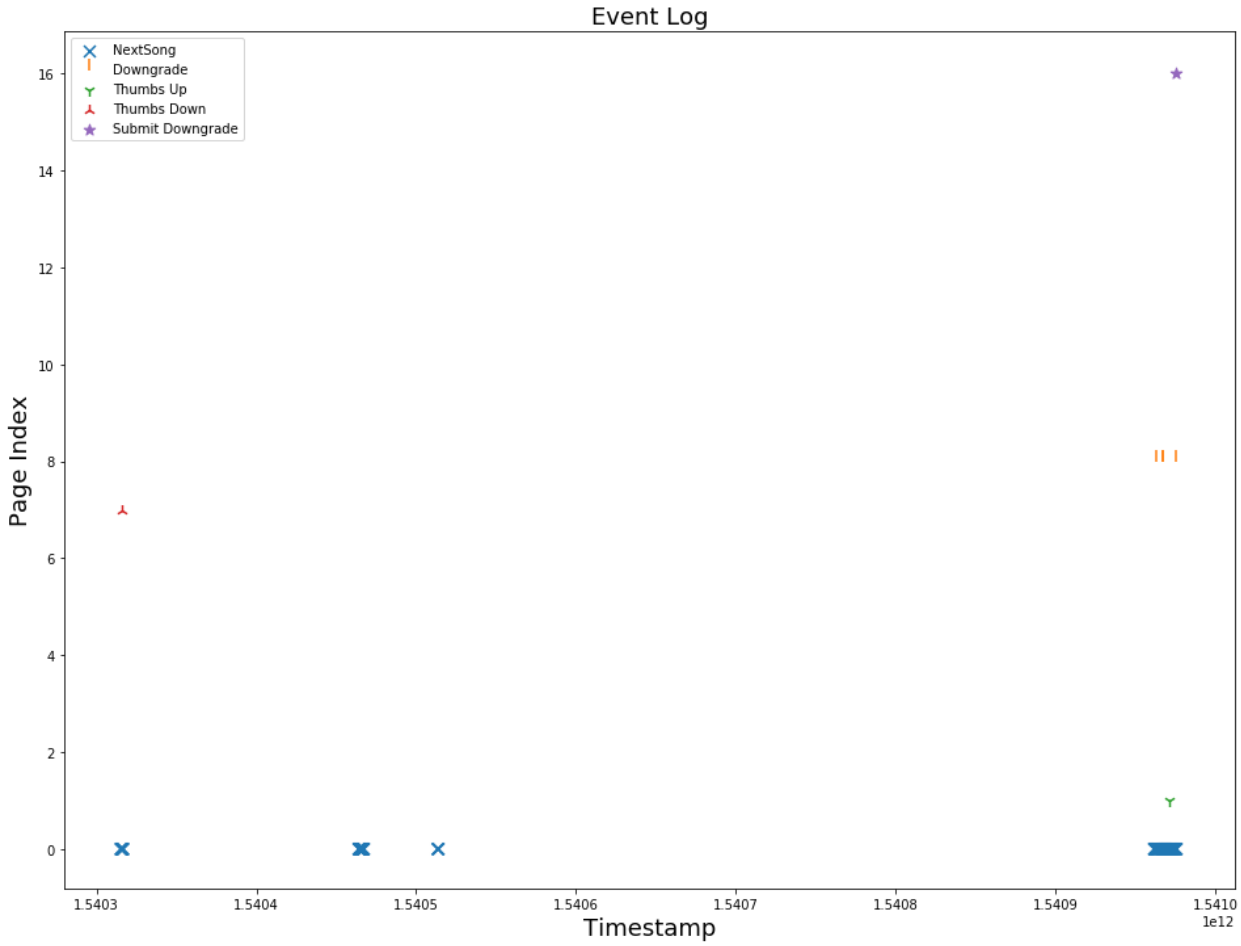

Figure 1 Visualization of Events Before Downgrade

The Figure 1 demonstrates that this user visited the Downgrade page several times before ending their subscription. This indicates that it could be possible to predict churn by using visits to the Downgrade page as a feature.

Methodology

The methodology of this project could be summarized on the following high-level components:

- Data Loading, Cleaning and Exploration
  - Typical steps of removing invalid values and unnecessary data are applied.
- Feature Engineering
  - The features selected for the project were the accumulation of event's a given type during a fixed time window of 7 and 14 days. This analysis was performed on data partitioned by user.
  - The events of interest are those that are good indicators of user engagement, listed below.
    - Thumbs up
    - Thumbs down
    - NextSong
    - Downgrade

    - Features were scaled and normalized to avoid undesired model biased. Experiments with both Max Min Scaling and Standard Scaling were performed.
        - MinMaxScaling transforms a dataset of vector rows, rescaling each feature to a specific range (often [0, 1]). It takes parameters number one is assigned to the maximum value a feature had.
        - Standard Scaling transforms a dataset of Vector rows, normalizing each feature to have unit standard deviation and/or zero mean.

- Data Preparation
    - Labels were assigned to the individual classes to be analyzed.
    - Data was randomly sampled and split on training, testing and validation.
- Modeling
    - The pyspark documentation was evaluated to determine the available machine learning algorithms.
    - From the available algorithms, those most suitable to the type of data being analyzed were selected.
        - Random Forest[1]
        - Logistic Regression[2]
        - Decision Trees[3]
    - Models are fit to training data.
- Testing
    - Test data was fed to the trained model, and accuracy metric was captured.
- Model Refinement
    - Based on initial results
        - Larger subset of features was selected
        - Model parameters were tweaked.
        - Different machine learning algorithms were tested.
- Additional Testing
    - Seeks to determine if model refinement succeeded.

Data Preprocessing

The following block diagram (Figure 2) illustrates the ETL steps followed as part of the data pre-processing, along with processing steps followed to reach the point of training and testing the predictive model.

[1] A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System (citizennet.com)

[2] Advantages and Disadvantages of Logistic Regression - GeeksforGeeks

[3] Top 5 advantages and disadvantages of Decision Tree Algorithm | by Dhiraj K | Medium

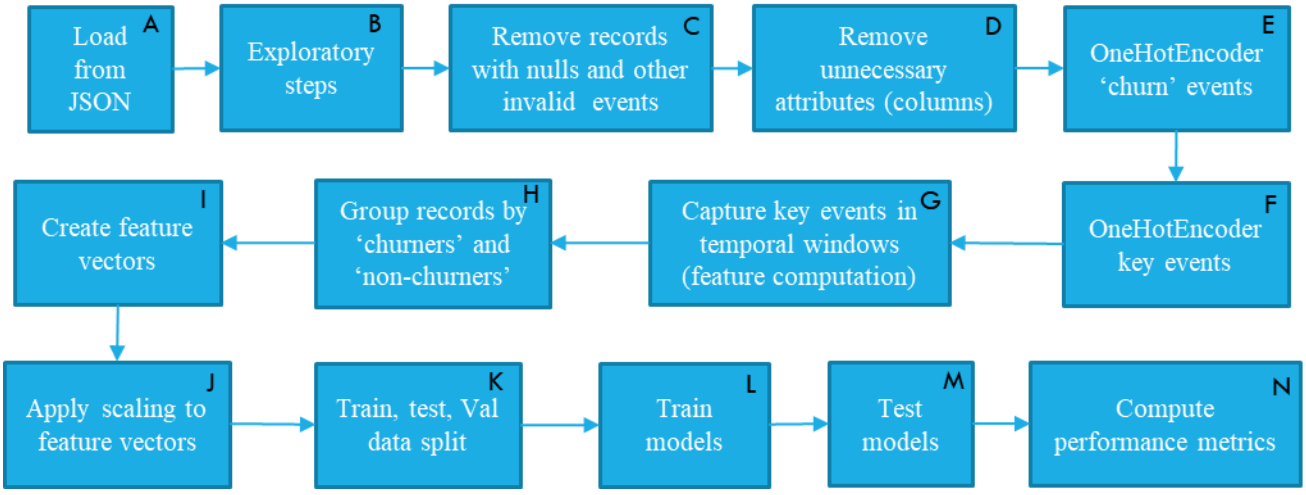

Figure 2 Sparkify Flow Diagram

Step description

A. Load from JSON – Loading Sparkify data from JSON file
B. Exploratory steps – gather and print various statistics and visualizations that allow the identification of cleaning and processing steps
C. Remove records with nulls and other invalid events – data cleaning step
D. Remove unnecessary attributes (columns) – Data reduction step
E. OneHotEncoder 'churn' events – Creates a new column with binary values. The column would have ones when a 'churn' (e.g. Cancelation Confirmation) event is found.
F. OneHotEncoder key events – 'key events' are those events identified as metrics of user engagement. 'key events' are: 'Thumb Up', 'Thump Down', 'Downgrade' and 'NextSong'
G. Capture key events in temporal windows (feature computation) – This steps refers to applying a sliding window focused on the activity of a specific user over the past 7 and 14 days. This window will be keep track of how many of each of the 'key events' occurred during those time windows.
H. Group records by 'churners' and 'non-churners' – Assign each user the label of 'churner' or 'non churner'.
I. Create feature vectors – groups all features into a single row vector. A necessary step to utilize pyspark's machine learning classes
J. Apply scaling to feature vectors – Applies scaling and normalizations to the vectors so that the disparities on feature magnitude do not create undesired model bias.
K. Train, test, Val data split – Of all valid records, 75% is utilized for training, and 12.5% percent for testing, and the final 12.5% for validation.
L. Train models – train instantiated machine learning models with the training data.
M. Test models – Feeds trained model with test data to compute accuracy metric.
N. Compute performance metrics – Select the best performing model, feed it with validation data and compute performance metrics.

Implementation

The following table describes some of the most important pyspark functionality applied to tackle the meet the goals of the project. The table pairs API calls to key steps on the block diagram of Figure 1

Step	Example Code Snippet
A	`path = "mini_sparkify_event_data.json"` `user_log = spark.read.json(path)`

B	`page_visit_distribution = user_log.groupBy("page").count()`
	`print("Distribution Event Types")`
	`page_visit_distribution.sort("count").show(page_visit_distribution.count(), False)`

C	`user_log = user_log.dropna(how="any", subset = ["userId", "sessionId"])`
D	`columns_to_drop = ['userAgent']` `user_log = user_log.drop(columns_to_drop)`
E	`churn_flagger = udf(lambda x: 1 if (x == 'Submit Downgrade') | (x == 'Cancellation Confirmation') else 0, IntegerType())` `user_log = user_log.withColumn("churn", churn_flagger(user_log.page))`
F	`like_flagger = udf(lambda x: 1 if x == 'Thumbs Up' else 0, IntegerType())` `user_log = user_log.withColumn("likes", like_flagger(user_log.page))`

G	`days = lambda i: int(i * 86400*1000)`
	`window_spec_7 = Window.partitionBy("userId").orderBy(func.col("ts").cast('long')).rangeBetween(-days(7), 0)`
	`user_log_for_ml = user_log.\`
	`withColumn("acu_dislikes_7", func.sum("dislikes").over(window_spec_7))`

H	`user_log_for_ml_churners = user_log_for_ml.filter(user_log.churn == 1)`

I	`feat_assembler_b = VectorAssembler(`
	`inputCols=["acu_downgrade_7", "acu_downgrade_14", "acu_next_7", "acu_next_14", "acu_dislikes_7", "acu_dislikes_14"],`
	`outputCol="features_b")`

J	`scaler_b = StandardScaler(inputCol="features_b", outputCol="nrm_features_b")` `user_log_for_ml = scaler_b.fit(user_log_for_ml).transform(user_log_for_ml)`
K	`train_dat, test_val_dat = user_log_for_ml.randomSplit([0.75, 0.25], seed=7)` `test_dat, val_data = test_val_dat.randomSplit([0.5, 0.5], seed=7)`

L	`log_reg_obj_b = LogisticRegression(featuresCol="nrm_features_b", labelCol="churner", predictionCol='logr_pred_b')`
	`log_reg_model_b = log_reg_obj_b.fit(train_dat)`
	`results= log_reg_model_b.transform(test_dat)`
	`print(results.filter(results.logr_pred_b == results.churner).count()/results.count())`

M	`print(results.filter(results.logr_pred_b == results.churner).count()/results.count())`

N	`y_test= results.select("logr_pred_b").rdd.flatMap(lambda x: x).collect()`
	`y_truth = results.select("churner").rdd.flatMap(lambda x: x).collect()`
	`print(classification_report(y_truth, y_test))`

Refinement

One of the initial observations made on the input data was that visiting the 'Downgrade' page seem like a strong indicator of future service downgrade. Initial experiments utilized the accumulation of such events over the 7 days pre-dating the churning event.

The best performing model achieve an accuracy of about 63% with this feature.

On the refinement stage, a variety of machine learning algorithms were used, along with different combinations of feature sets and scalers. Despite of this effort, no performance improvements were observed.

Results

Model Evaluation and Validation

A variety of performance metrics were gathered for each of the model trainings and testing experiments. The experiments included making different combinations of feature vectors, of machine learning algorithms and feature scaling and normalization. For individual metrics on these attempts, consult the Jupyter notebook provided with this report.

Experiment variables:

Features:

- Using accumulated number of visits to the Downgrade page over 7-day windows
- Using accumulated number visits to Downgrade page over 7 and 14 day windows
- Using accumulation number of visits to the following pages, over both 7 and 14 day windows
    - Thumbs Up
    - Thumbs Down
    - Downgrade
    - NextSong

Models:

- Logistic Regression
    - Parameters:
        - Default parameters as listed in the API Reference[4]
- Decision Tree
    - Parameters:
        - Default parameters as listed in the API Reference[5]
- Random Forest
    - Parameters
        - Default parameters as listed in the API Reference[6]

The best performing model was the Logistic Regression, when training occurred with the accumulated number of visits to the Downgrade page over 7-day and 14-day windows. It achieved an accuracy of 63%. The following table summarizes other metrics for the best performing model for the Test Dataset.

Label	Precision	Recall	F1-Score	Support
0	0.75	0.5	0.6	6
1	0.57	0.8	0.67	5
avg/total	0.82	0.64	0.63	11

[4] LogisticRegression — PySpark 3.1.1 documentation (apache.org)
[5] DecisionTreeClassifier — PySpark 3.1.1 documentation (apache.org)
[6] LogisticRegression — PySpark 3.1.1 documentation (apache.org)

This combination of features and models led to an 80% recall on detecting 'churners'. A high recall, which implies a low False Negative rate, indicates that the model is less likely to miss a 'churner', perhaps at the expense of classifying non-churners as such. For the use case of Sparkify, it is best not to miss a potential 'churner' so business with a customer is preserved.

Justification

Modestly optimistic results were obtained. However, for a model to be considered optimal, classification accuracies of over 90% are needed. Not achieving over 90% could be attributed the small amount of data. Despite of large number of records, these records only were representative of the activity of less than 200 users on 2 months of activity. More over, the activity occurred between October and December or holiday seasons. User spending habits vary significantly throughout the year, and on the holiday season, several external factors (such as competing expenses and competitor offers) could influence churning rate more than factors extracted from the activity data (user log).

Conclusion

Given the achieved accuracy, it is difficult to conclude that the Sparkify dataset is sufficient to construct predictive models capable of warning platform stakeholders about potential drops on paid subscription before they occur. The pre-processing steps applied to the data and the features selected for data analysis allowed the machine learning models to detect engagement patterns, and label a user as a 'churner' or 'non-churner' with a small margin over 50%, which not much better than luck or chance.

Reflection

Implementation to a solution to the Sparkify problem highlight key distinctions on working with Big Data vs traditional datasets. On working with big data, implementation of the various algorithmic steps need to leverage the distributed processing framework as much as possible. This project highlights the importance of utilization pysparks API to ensure that processing steps are scalable to large volumes of data.

Analyzing a stream of data over time is also a key distinction on the field of machine learning and artificial intelligence. For successful predictions, data scientist should look at data patterns that happened before an event of interest. Predictive models are effectively chasing a moving target, and more factors come into play.

Improvement

- Deployment on a compute cluster (AWS or IBM Watson)
- Modeling and testing on the entire 12GB dataset.
- Apply pyspark ParamGridBuilder to determine best model hyperparameters.
- Test other models such as Linear Regression, Linear Support Vector Machine and Naïve Bayes.
- The effects of feature normalization need to be studied more carefully. By default, pyspark normalizes the rows vectors. Applying normalization based on the entire population of a given feature (column) may lead to desirable statistical improvements.