Predicting Song Popularity

CSCI 4253 - Final Project

Evan James
Johnathan Kruse
Mario Ramirez

Contents

Introduction

Globally, the music industry is valued at approximately $16 billion [7]. Furthermore, the number of songs downloaded and streamed each year is estimated to be well over a billion and is growing each year [1]. As a musician, the market has become increasingly difficult to stay relevant. One of the greatest challenges is creating a song which will become “popular” and thus incur more consumption by listeners. Our goal is to help musicians around this problem by determining both if their song will be popular, and characteristics of what would make their song more popular.

For our project, we utilized the Million Song Dataset -- found at https://labrosa.ee.columbia.edu/millionsong/ -- to train a machine learning algorithm to predict whether songs will be popular or not [4]. The MSDS contains over one million songs along with information of the song taken from The Echo Nest API. The song attributes include tempo, time signature, key, energy, danceability, and many others. The most interesting metric is “hotttness” (note the spelling is intentional), which is a value which represents how popular the song was on a scale from 0 to 1 [11]. We will have a machine learning algorithm ingest the various attributes of each song and use it to be trained to guess whether any given song will be popular or not. Once the ML algorithm is trained, we will be able to query the Spotify API to get the necessary artifacts from a song and feed it into the algorithm to attempt to predict whether the song will be hot or not.

This is an interesting problem for many reasons. It can easily be seen how an algorithm like this would be useful in the music industry. It could be used to not only predict whether a song will be popular or not but could also be used to give insight into what kind of characteristics cause a song to be popular or not. The types of song attributes and their hotness can also be tracked over time, showing what kind of songs were popular each year and what kinds of trends happened over time with the sound of music. Having insight as to what makes music popular would be extremely useful for music streaming companies, such as Spotify, but also to record labels who have to decide on what kind of music to invest money and resources into.

A data-scale solution is necessary for this problem due to both the Million Song dataset only being accessible via an EC2 instance and the sheer size of the data can be handled much easier on AWS infrastructure than local resources. In the end, a model was able to use song attributes to predict popular songs with an ~80% accuracy rating, albeit this result is likely biased and it is recommended that more data be imported for further training and testing.

Related Work

Regarding experience, most of the team members have had little or no prior experience with many of the tools used for this project. Johnathan had taken a Machine Learning course prior, albeit had not worked with AWS SageMaker and had not used K-Nearest-Neighbours to this degree (for a small homework assignment, but not a project). This is mostly the extent of experience for machine learning. New technologies learned for this project included AWS SageMaker, working with Elastic Block Storage (EBS) snapshots and mounting them, converting HDF5 to CSV, and converting JSON to CSV.

On the “Towards Data Science” web page there is a story about an ECE student at the University of Texas at Austin who did a project very similar to ours back in May named “song Popularity predictor” [8]. The student, Mohamed Nasreldin, also used the Million Song Dataset to train a Scikit-Learn ML model to predict the popularity of a song. The findings in this project were similar to ours as there were no conclusive results to predict the popularity of a song that was better than chance which was the outcome of our project. Mohamed’s project also had a data exploration portion that included data cleaning since a lot of songs had empty tag fields including a lack of a “hotttness” field.

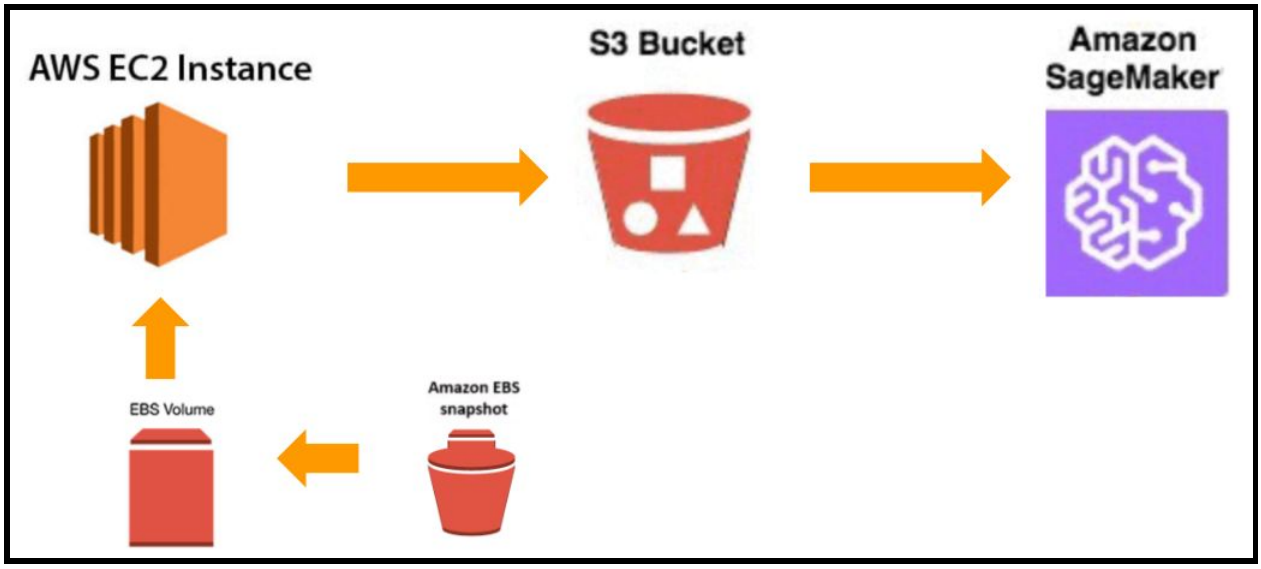
Mohamed did some dataset cleaning that we could have benefited from as well as other ML model testing as well. First, since a lot of the songs, about half of the subset that Mohamed was using, were missing the hotttness measure meaning Mohamed needed another way to determine the popularity to include a larger dataset. Thus, Mohamed decided to classify if a song was a hit or not by determining if the song appeared on the Top 100 Billboard at least once. We did not do this meaning our subset of data was much smaller which could have influenced our results to be worse.

Also, for training the data Mohamed conducted more trials on more ML models then we did and saw different results then we did. We only looked at two ML models which were Logistic Regression and KNN which from the results we determined that KNN gave us the better accuracy so we went with that model. Mohamed looked at six different models (listed in most accurate to least accurate): XGB, Logistic Regression, Random Forest, KNN, Decision Tree, Support Vector Machines. Unlike us, Mohamed found that XGB model gave the best accuracy and Logistic Regression was found to be better than KNN. So, looking at projects similar to ours we could have benefited from more model testing and more data cleaning to improve our results [8].

System Design

For the project, we used the Million Song Dataset (MSDS). The dataset contained 273 GB of song information compiled by Lab ROSA using data provided by The Echo Nest [5]. The Echo Nest, a Spotify owned company, is a data collection organization whose focus is specifically on music [11]. LabROSA is a Columbia University research organization whose focus is on intelligent sound recognition. LabROSA selected songs based on familiarity of the artist, song similarity and popularity. The dataset was last updated in November of 2015 with a majority of songs being from 2010 when the list was compiled. Regarding overlap, the songs selected by LabROSA are mostly unique (a few duplicates do exist) [4].

Figure 1: System Architecture



As seen in Figure 1, the Million Song dataset is stored on AWS as a public Elastic Block Storage (EBS) snapshot. An EBS snapshot is a data-store capture at a specific “point-in-time” [2]. In order to use an EBS snapshot, it must be mounted as a volume for an EC2 instance. A volume creates a replica of the snapshot and hosts it with an AWS account allowing an EC2 instance to mount it and make use of it. LabROSA has stored the data in an HDF5 format. HDF5 is a file storage technology which allows very large heterogeneous data sets, file compression and search optimization [12].

For each song, there are over fifty features (data about the song), all of which are not needed. To minimize storage and training costs, irrelevant columns were dropped and the data was compressed further to a single CSV file. This made it simpler to import into AWS SageMaker and further clean the data. Once an EC2 instance was set up (md5.large) through the console and the EBS volume mounted, the “parsing/h5_to_csv.py” script was ran on the EC2 instance

which converted all the HDF5 files into a single CSV. This CSV was then stored in an S3 bucket.

Per Figure 1, to run the machine learning model AWS SageMaker was selected. SageMaker provides a development environment for implementing machine learning models and allows users to spin up machines with customized resources (e.g. ml.t2.2xlarge). In the background, SageMaker can scale up and distribute computational tasks without exposing many of the underlying technical setup details to developers. There are two ways to use the service, one through the AWS SageMaker online console, and the other through the Command Line Interface (CLI). This project used the online console as it allowed access to a Jupyter Notebook GUI interface. Jupyter Notebooks natively support Python development, the language used for development. Furthermore, these notebooks make it easier to plot graphs and see visual results (e.g. accuracy graphs). Another benefit of SageMaker is that is already part of the AWS ecosystem which made it very easy to integrate with other AWS services [3].

Once setup, the Million Song CSV dataset was imported into SageMaker from an S3 bucket. Here the data was converted into a pandas dataframe from the Python pandas package. A pandas dataframe is a data structure most similar to a 2D list, albeit pandas provides much more support for manipulating, analyzing and cleaning than a native Python list [10]. Using the pandas dataframe, the data was cleaned, removing any incomplete data sets (e.g. missing year release, artist name, etc). To build the machine learning model, the library scikit-learn was used which offered support for the classification algorithm, K-Nearest-Neighbors. Lastly, to create a new prediction, a user can launch the Jupyter Notebook “ml_final.ipynb” on SageMaker and then import a CSV file with the prediction(s) (and proper headers) to have the trained model return a prediction.

Evaluation/Findings

Firstly, it is important to have a basic understanding of machine learning. A machine learning model is similar to a mathematical formula which takes inputs and returns an output. For example, consider y = mx + c. In this formula, “x” is an input value while “y” is the output. In comparison, an example of some of a song’s “input” features include:

- Duration
- Number of beats
- Loudness
- Tempo
- Mode
- Key
- Year

Thus, the “output” is “yes” or “no” the song will be popular. Initially, two models were tested, being K-Nearest-Neighbors (KNN) and Logistic Regression. KNN works by a majority vote. When a prediction is made for a new song, the euclidean distance for each of the input features (e.g. which song has the most similar duration, loudness, tempo, release year) are compared with songs of the existing dataset (known as “training data”). The closet K points (where K is a positive integer e.g. 4, 5, 10) are looked at. If a majority of those K songs are “yes popular” then the new song will be predicted as popular. Else if the majority is “not popular” then the new song will be predicted as not popular [9].

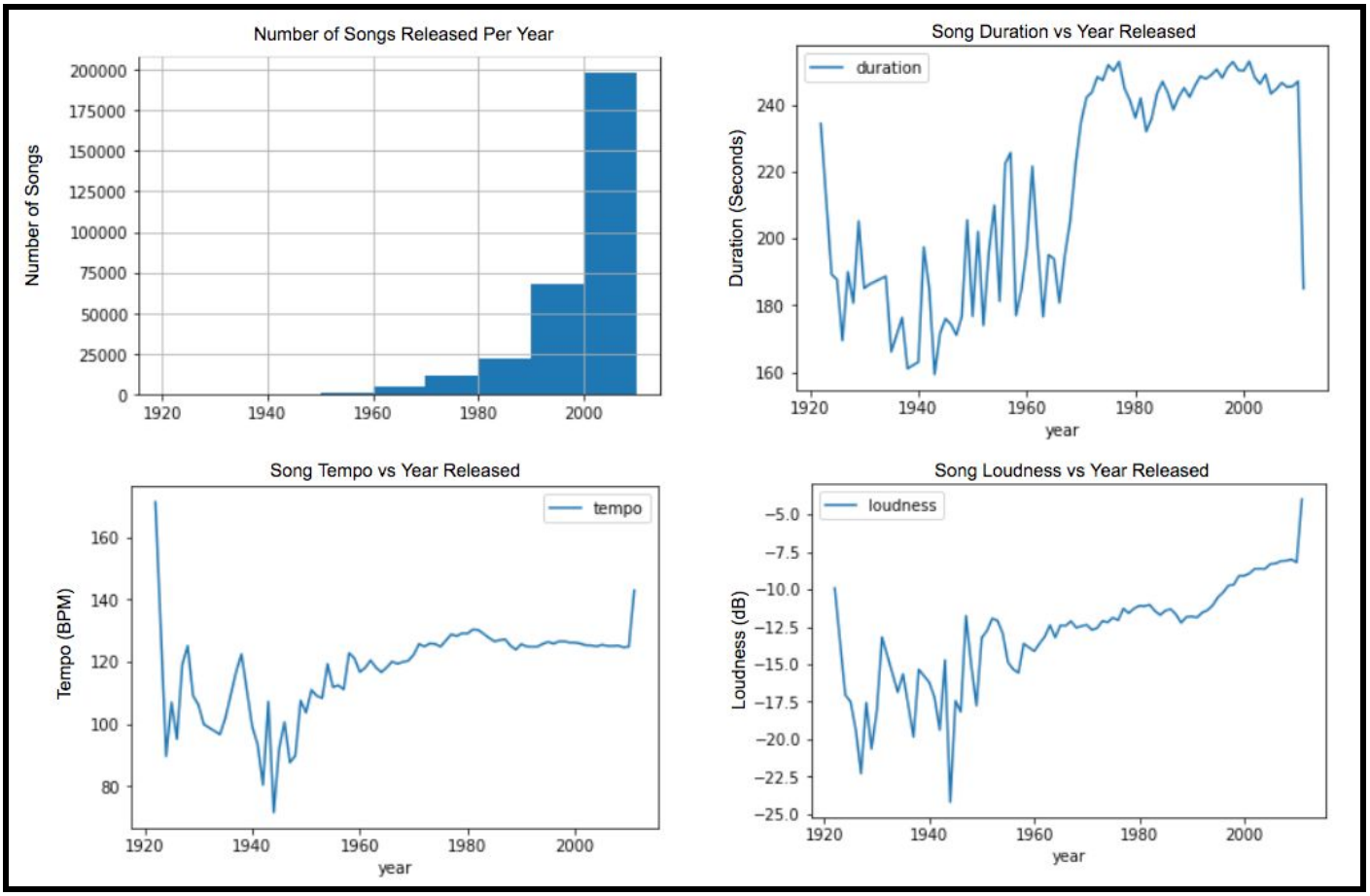
Comparatively, Logistic Regression is very similar to a straight line formula (such as y = mx + c). The only problem with using a straight line formula is a numerical score is returned whereas the output had to be a “yes” or “no” classification result. Logistic regression solves this by feeding a straight line formula into a sinusoidal function (details of formula are not important) which outputs a “yes” or “no” classification result [6].

To see which model was more appropriate for this use case, 20% of the Million Song dataset was randomly sampled for testing while the remaining set was used for training. Then the model predicted if a song will be “hot” or “not hot” with a confidence between [0, 1]. The Million Song dataset has a “hotttness” numerical score for popularity [5]. In order to turn this into a “yes” or “no” classification problem a threshold was imposed that said if the score was greater than 0.5, the song was “hot”, else not.

Resultantly, KNN offered a better performance with higher overall accuracy scores. Thus why KNN was selected. To make KNN more accurate, it was important to determine which input features provided the best correlation of “hot” songs. It was also important to determine which value for K gave the best accuracy for the given inputs.

From here, the team explored the data for common/popular trends.

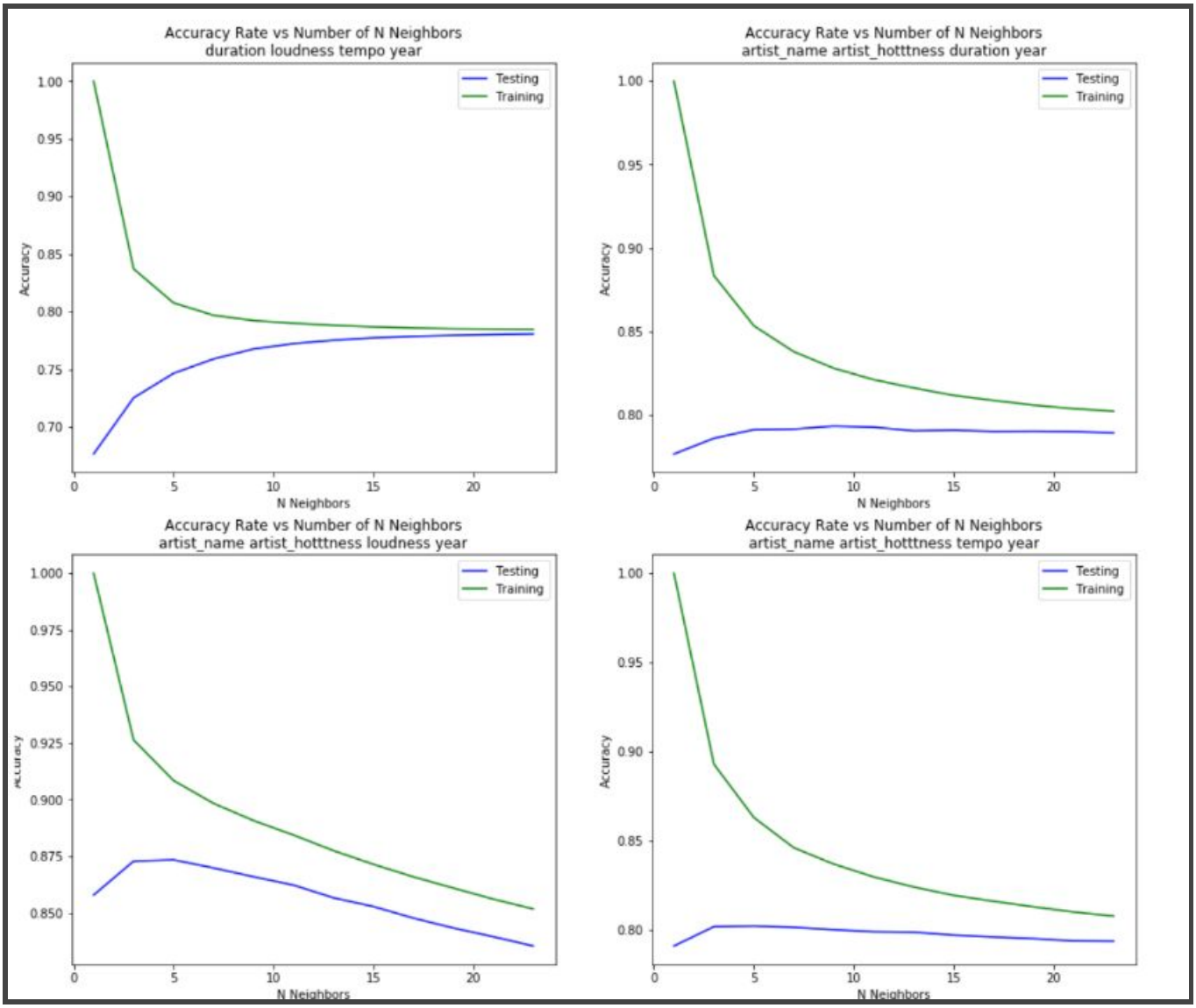
Figure 2: Data Trends



As can be observed in Figure 2, from 1960 and onward there was an increase in Tempo, Duration, and Loudness that was stayed relatively the same for the past three decades. However, we also see that the majority of our dataset is from the early 2000s so our data could be skewed to this lack of era diversity in our data.

Using these data trends, possible input features were determined to test for accuracy (duration, loudness, tempo, year and artist name).. Additionally, in order to determine which number of neighbours would make KNN most accurate for these input features, we tested KNN multiple times with different input feature combinations against the number of K neighbours. These results can be seen in Figure 3.

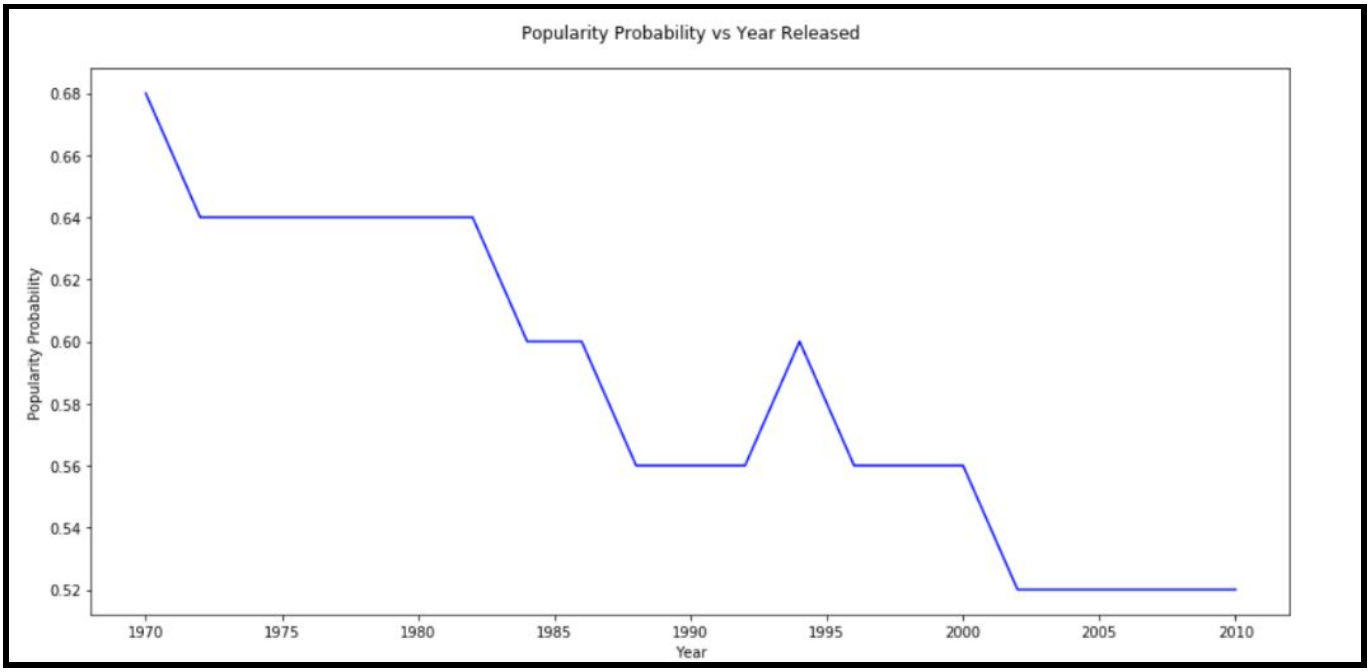
Figure 3: Performance Metrics of K-Nearest Neighbours (KNN)



A quick mention on Figure 3. The blue “Testing” line (the lower line in each graph) is the accuracy of the model on testing data. This data was not used for KNN to train on, rather is was a completely new set of data the model had not seen until this prediction. A higher value for this line indicates that the model is more accurate. The green “Training” line (the line on top in each graph) is the accuracy of the model on the training data. This is data the model used to train on in order to make predictions. Testing of the accuracy on the “Training” data shows looks for overfitting. Overfitting is where the model does not generalize and instead makes predictions based very strictly on what the training data is. A value for this line that is much greater than the blue line indicates that the model assumes all datasets will be nearly exactly like this training data. Hence any new data that comes in and is predicted on could have a low accuracy because not all data will be identical to the training set. This same idea can be applied to the number of K neighbours (N is used in Figure 3). A low K value can indicate overfitting.

Due to this rationale, duration, loudness, tempo and year were found to be the best predictive factors for the inputs given. Additionally, ~25 neighbours was found to be the best value for K for these inputs. Per Figure 3, This gave an ~80% accuracy rate and did not indicate strong overfitting.

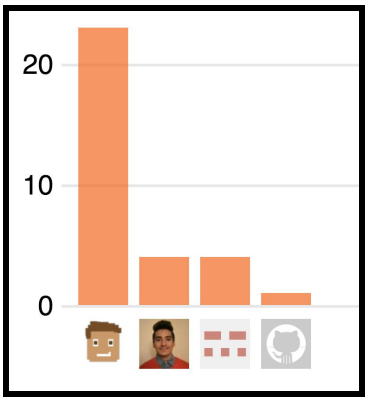
Figure 4: Song Popularity vs Year Released



Lastly, one area explored briefly was predicting the year a song would be most/least popular. Per Figure 4, KNN can provide a probability score for the popularity of a song. To generate the graph, the model was run for the same song with year released incrementing from 1970 to 2010. The song in the graph actually came out in 2009 and its true classification is “not popular”. Perhaps had it come out in previous years it would have had a better probability of success, or perhaps more data is needed to make a better-informed/accurate decision. As mentioned before, this was briefly looked at as to the potential further capabilities that this model could provide.

Review of Team Member Work

Figure 5: Github Commit Graph
Left to right: Johnathan, Mario, Evan



Johnathan Kruse:
I spent most of my time working with AWS SageMaker and the Machine Learning models. This involved most of the analysis and work with determining the machine learning model, looking for correlations for input features, parameter tuning (number of neighbours) and additional exploration features such as which year a song will be most popular. This also means that most of my development work was done in Python and can be seen in both the “ml_prototype.ipynb” and “ml_final.ipynb” Jupyter Notebooks. Note most of the work from the latter half of the findings section is from “ml_final.ipynb”. The graphs comparing input feature combinations to number of neighbours can be found here too. I also created some instructions and bash scripts to make it a little easier to get the Million Song dataset snapshot and then run Evan’s script for converting. This can be found in the README.md inside the parsing directory.

Evan James:
I spent most of my time working on cleaning the data and doing preliminary data exploration. The biggest hurdle was finding a clean way to parse the .h5 files and concatenate them all into a big .csv file with all million data points in it. This took a couple iterations of testing with the 10,000 song subset of the MSDS, with different approaches to parsing and collecting the data. Eventually I realized I was forgetting to use the utility library that is provided by LabROSA to quickly and easily parse the .h5 files. Once I found this, it was much easier to get the data we were interested in out of each file. By using the Python OS library it is relatively simple to do a walk of a directory including all files and subdirectories, so it just became a matter of allowing the cleaning script to run on a virtual machine long enough to process all million songs. This was done on an AWS VM, running overnight, and a mere 16 hours later we had a 175 MB .csv file containing information for all million songs in the dataset. This is a significant reduction in

size, from about 300 GB of .h5 files to a single .csv which is almost three orders of magnitude smaller.

The data exploration was fairly simple, using a Jupyter notebook and basic Python data visualization libraries. I looked at various trends, such as tempo or loudness over time, or the number of songs in the dataset for each chunk of years. This was interesting to look at, because knowing more in depth information about the formatting of the data allows us to make more sense of the results we get from the machine learning model. It is also just interesting to see trends in data like this over time, since it is such a large corpus there is plenty of ways to look at the massive amount of data.

Mario Ramirez:
I initially researched ways to implement scikit-learn model on sagemaker, focusing on creating a docker container to run on sagemaker but ultimately Johnathan implemented using jupyter notebook. Also, I aided in research of ways to convert mdf5 dataset to csv or json. Evan implemented a way to transform to csv but before that script was completed I discovered a temporary solution in the form of a subset of million song dataset that was transformed to csv to be used for initial testing of ML model.

Then, I looked into using the LastFM dataset which was a list of songs wich timestamps of when the song was listened too to predict the best time of year to release a song to maximize popularity. So, I created the Python script that converted the LastFM JSON dataset to CSV format. However, the timestamps were found to be all in the year 2011 and from three specific months so this data was determined to be not usefull for us.

Conclusion

In conclusion, per the evaluations section a machine learning model was successfully created for predicting song popularity based on general attributes/metrics about a song. This demonstrated that the year of release, duration and loudness can contribute to being strong predictive factors of song popularity. More modern songs tend to be louder than older. Coupled with the year, it is likely that a quieter song will be less popular in modern times.

While an ~80% accuracy rating was achieved, further testing on more datasets is strongly recommended. This is because while a million songs were available, cleaning the dataset led to less than 500,000 songs for training. As Figure 2 demonstrated, while there is nearly half-a-century span of songs, a majority of the songs are skewed towards 2010. A good majority of these modern songs are fairly popular (not all of them). This means that if a song is from a more recent/modern time then it is likely to be predicted as popular and indeed actually have been popular, increasing the accuracy rate.

However, this skew is not representative of all cases in the industry because all modern songs are not popular. Furthermore, of the more modern songs that are popular it is likely a minority of songs released in a year will be popular. Thus additional data is highly recommended to confirm, adjust or refute these results. Note that this skew does not apply to all songs in the dataset as Figure 4 showed. In that case a more modern song was predicted to be more popular farther back in time as opposed to closer.

If this project were continued, in addition to more data, it would be interesting to explore predicting which year a song would be most popular as well as obtaining more information about song genres to see correlations in both the popularity of a song in a certain genre, what year a song would be most popular and give insights into which genres were most popular during different eras. For this to be done, another dataset would be required. A possible dataset to join with the Million Song dataset would be *tagtraums industries* Million Song dataset annotations found at their website: http://www.tagtraum.com/. This dataset combines song id's from the Million Song dataset with genre data from the *All Music Guide* organization which collects additional song data [4].

References

[1] “2016 U.S. Music Year-End Report.” Nielsen, January 9, 2017. http://www.nielsen.com/us/en/insights/reports/2017/2016-music-us-year-end-report.

[2] “Amazon EBS Snapshots Documentation - Amazon Elastic Compute Cloud.” Amazon Web Services, Inc. Accessed December 14, 2018. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html.

[3] “AWS Public Data Set - Million Song Dataset.” Amazon Web Services, Inc., February 7, 2011. https://aws.amazon.com/datasets/item/.

[4] Bertin-Mahieux, Thierry. “The Million Song Dataset.” Million Song Dataset, official website, 2011. http://labrosa.ee.columbia.edu/millionsong/.

[5] Lamere, Paul, Brian Whitman, Daniel P.W. Ellis, and Thierry Bertin-Mahieux. “The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011),” 2011.

[6] “Logistic Regression.” PennState Eberly College of Science, 2018. https://onlinecourses.science.psu.edu/stat504/node/149/.

[7] McIntyre, Hugh. “The Global Music Industry Grew By 6% In 2016, Signaling Brighter Days Ahead.” Forbes, April 25, 2017. https://www.forbes.com/sites/hughmcintyre/2017/04/25/the-global-music-industry-grew-by-6-in-2016-signalling-brighter-days-ahead/#5a14e7bc63e3.

[8] Nasreldin, Mohamed. “Song Popularity Predictor.” Towards Data Science, May 5, 2018. https://towardsdatascience.com/song-popularity-predictor-1ef69735e380.

[9] “Nearest-Neighbor Methods.” PennState Eberly College of Science, 2018. https://onlinecourses.science.psu.edu/stat857/node/21/.

[10] “Pandas: Powerful Python Data Analysis Toolkit Documentation.” Pandas, December 12, 2017. https://pandas.pydata.org/pandas-docs/version/0.21/index.html.

[11] “The Echo Nest.” We Know Music, 2018. http://the.echonest.com/.

[12] “What Is HDF5?” The HDF Group, January 27, 2017. https://support.hdfgroup.org/HDF5/whatishdf5.html.