Hit Song Science: Predicting Music Popularity
Data Analytics Final Project – Spring 2019
Ranjeeta Bhattacharya

1. Introduction:

Music is the universal language of communication and plays an integral part of most of our lives. Where words fail, music prevails. Writing a hit song is an art form. Many artists continue to search that perfect formula which will enable them to create kind of music which will appeal to the listeners ultimately turning into great hits. Many rappers of this generation boldly claim that they always know when a song will be a hit. Possibly there are patterns which can be unearthed to substantiate the claim. The purpose of this project is to try and investigate and analyze whether there are certain characteristics for hit songs. What are the features which actually contributes to the success of a track?

In 2018, the music industry generated a mid-year retail revenue of around $4.6 billion[1] in the United States alone. Thanks to the growing streaming services (Spotify, Apple Music, etc.), the industry continues to flourish. Streaming services alone amounted to 75% of overall revenue for the record industry securing the lion's share of revenue[2]. There has been work in the past to predict music popularity with varying degree of success. Making predictions regarding how popular a song can be based of range of audio features has gained lot of traction in recent times. Many private companies in the industry are known to work with major artists and labels to assist them to strike the right chord, however for competitive reasons the details are mostly help private and largely unknown to us.

Apart from audio features, there has been studies regarding lyrical contribution towards song popularity especially for "Rap" genre. Machine learning techniques suggest that for rap music, major weightage lies towards profanity of the lyrical content. However, studies also revealed that meaningful lyrical rap songs were also consistent hits. [3]

Another important popularity measure will be the timing aspect of a song. It has been noted that yesteryear hits continue to enthrall listeners, even the cover versions irrespective of performance quality continue to reign the charts. Launching a musician's career by performing several cover songs remains one of the most popular ways to come to limelight instantly.

Overall, one can confidently say that music analysis using suitable machine learning techniques might unearth valuable insights and guide artists in elevating overall music production process and having a meaningful roadmap.

---

[1] http://www.riaa.com/wp-content/uploads/2018/09/RIAA-Mid-Year-2018-Revenue-Report-News-Notes.pdf
[2] https://www.theverge.com/2018/9/20/17883584/streaming-record-sales-music-industry-revenue
[3] https://www.mic.com/articles/131092/these-students-are-using-data-science-to-predict-which-rap-songs-will-become-hits

2. Background / Motivation:

How does one write a hit song? Through decades several authors have written books on this based on analysis of previous hits to come up with recommendations.[4] Identifying and right combination of audio feature elements can actually make a big difference. DJ Khaled boldly claimed to always know when a song will be a hit. [5] Considering my keen interest in music, I decided to investigate further by forming the key business questions involving predictability of a hit song. Data gathering phase was initiated by creating an account in **Spotify developer platform** which exposes a range of fantastic api's through which we can read calculated audio features of tracks to learn about its danceability, energy, valence and more. For more advanced use cases, it is possible to read in-depth analysis data about tracks such as the segments, tatums, bars, beats, pitches and more. However, for this project I decided to restrict the scope to just analyzing the track related features. Through the Spotify api's, range of data could be collected based on release year. Spotify api's returned the response in JSON format which could be decoded easily using lightweight python library **Spotipy**. "Spotipy supports all the features of Spotify Web API including access to all end points, and support for user authorization." [6] This made the entire process of data collection very straightforward and convenient.

Billboards hot-100 remains one of the most recognized "music industry standard record chart in the United States for songs, published weekly by Billboard magazine. Chart rankings are based on sales (physical and digital), radio play, and online streaming in the United States." [7] The hot-100 chart is considered as a unified, all-encompassing popularity chart. Through Python web-scraping, it was possible to extract the list of songs featuring in the hot-100 chart for the year range 2015 – 2019. Unofficial web scraper "billboard.py" [8] was modified and suitably invoked to get the list of songs featured in the chart. Corresponding audio features for all these hit songs could be extracted from Spotify. So, at the end, Spotify and billboards hot-100 gave a decent mix of data which could be analyzed and mined to build a predictive model for popularity predictions.

3. Literature Review:

In preparing for this study, a literature search was completed. Among the articles found were "HITPREDICT: PREDICTING HIT SONGS USING SPOTIFY DATA"[9], "Show Me What You Got" [10], "Predicting A Song's Commercial Success Based on Lyrics and

---

[4] https://www.theguardian.com/music/musicblog/2011/jul/14/how-to-write-a-hit-song
[5] https://www.youtube.com/watch?v=M0be5674X9Y&feature=youtu.be&t=12m45s
[6] https://spotipy.readthedocs.io/en/latest/
[7] https://en.wikipedia.org/wiki/Billboard_Hot_100
[8] https://github.com/guoguo12/billboard-charts/blob/master/billboard.py
[9] Georgieva, Suta & Burton, 2018. HITPREDICT: PREDICTING HIT SONGS USING SPOTIFY DATA STANFORD COMPUTER SCIENCE 229: MACHINE LEARNING. Retrieved from http://cs229.stanford.edu/proj2018/report/16.pdf.
[10] Chen, Dixit, Sanyal & Ed, Yip (2018). Show Me What You Got, Song Popularity Prediction Using FMA Dataset. Retrieved from https://www.ischool.berkeley.edu/sites/default/files/sproject_attachments/info_251_-_final_project_report.pdf.

Other Metrics" [11], "A Bayesian Approach to Understanding Music Popularity" [12], "AUTOMATIC PREDICTION OF HIT SONGS" [13], and "Clustering Music by Genres Using Supervised and Unsupervised Algorithms" [14]. Of these articles, the first three have been reviewed for understanding and the results of these reviews are discussed below.

The **first** article titled "HITPREDICT: PREDICTING HIT SONGS USING SPOTIFY DATA" aimed at predicting the hit song science by extracting the audio features of ~4000 songs collated from different sources. To predict a song's success, six different machine learning algorithms were used namely Expectation Maximization (EM), Logistic Regression (LR), Gaussian Discriminant Analysis (GDA), Support Vector Machines (SVM), Decision Trees (DT), and Neural Networks (NN). Focus was mainly on accuracy of results, precision and recall was reported as well. Initially clusters were identified using EM algorithm assuming no labelled data. The process was repeated with semi-supervised EM algorithm with 20% labelled data. For supervised algorithms, the dataset was divided into training and validation examples using 75/25 split. LR and GDA both fit a decision boundary to the data. Both EM and semi-supervised EM algorithm gave poor accuracy of around 50%. It was concluded that un-supervised learning algorithms are inappropriate for the problem. LR and GDA yielded a reasonable accuracy of 75.9% and 73.7% against the validation data, with similar accuracy against the training data indicating no overfitting. The precision and recall on the validation set were acceptable. For the SVM, each kernel yielded reasonable accuracy on the training data but poor accuracy on the validation data, indicating significant overfitting. DT algorithm also caused high overfitting which was corrected using Random forests. Experimentation was done with different maximum depths for DT to prevent overfitting. Using 10 trials of 500 random samples was the most successful measure for each algorithm. LR and NN were the most successful algorithms for which analysis was performed to determine features with the greatest influence on predictions. Finally, it was concluded that more data can be used to reduce to variability of the results between different supervised algorithms. Set of audio features actually used for the analysis can be extended.

The **second** article titled "Show Me What You Got" did song popularity prediction based on a combination of music metadata features (e.g. genre, title and date_released), audio features (e.g., Mel-frequency cepstral coefficients and spectral contrasts), and musicality features (acoustic-ness, danceability and energy.) As part of feature engineering, hand engineered features were generated with the goal of capturing inherent characteristics of a song that listeners use consciously/subconsciously to determine if they would enjoy a song. Machine learning models were built and evaluated for different genres individually. Overall performance metric comprised of Accuracy, Precision, Recall and

---

[11] Xue, A. &Dupoux, N (2014). Predicting A Song's Commercial Success Based on Lyrics and Other Metrics. Retrieved from http://cs229.stanford.edu/proj2014/Angela%20Xue,%20Nick%20Dupoux,%20Predicting%20the%20Commercial%20Success%20of%20Songs%20Based%20on%20Lyrics%20and%20Other%20Metrics.pdf.

[12] Shapiro, H (2017). A Bayesian Approach to Understanding Music Popularity. Retrieved from https://dukespace.lib.duke.edu/dspace/bitstream/handle/10161/9747/Shapiro_Thesis_Final.pdf?sequence=1&isAllowed=y

[13] Dhanaraj, R., & Logan, B., (2005). Automatic Prediction of Hit Songs. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.9033

[14] Kim, K., Yun, W., & Kim, R., (2005). Clustering Music by Genres Using Supervised and Unsupervised Algorithms. Retrieved from http://cs229.stanford.edu/proj2015/129_report.pdf

AUC for individual classifiers: Gradient Boosting Classifier, Random Forest Classifier, and Logistic Regression. Performance metric was evaluated across different genres and listed accordingly. It was concluded that hand generated features can provide an effective way of differentiating which features by genre are most important for predicting popularity.

The **third** article titled "Predicting A Song's Commercial Success Based on Lyrics and Other Metrics" discussed about predicting a song's commercial success based on Lyrics and other metrics. A subset of songs was drawn from the Million Song Dataset[15] with their associated metadata, including genre, duration, artist familiarity, artist hotttnesss, year, tempo, danceability, energy, loudness, key, time signature, and" hotttnesss". The dataset also includes lyrical data for a large portion of the songs in it. Feature selection process was iterative – starting with a list of features which intuitively made sense followed by application of supervised learning techniques like linear regression, L1 penalized regression, and k-nearest neighbors. This allowed narrowing down of the feature vector and gave an idea of which features will be more or less predictive. As part of this study, although the problem of predicting "hotttness" was not very effective using the combination of chosen features+model, however there was some indication regarding the features which may be weakly indicative of the factor. Ultimately it was concluded that different set of features needs to be employed for carrying out further analysis.

4. **Approach:** High level overview of data preparation is depicted below:

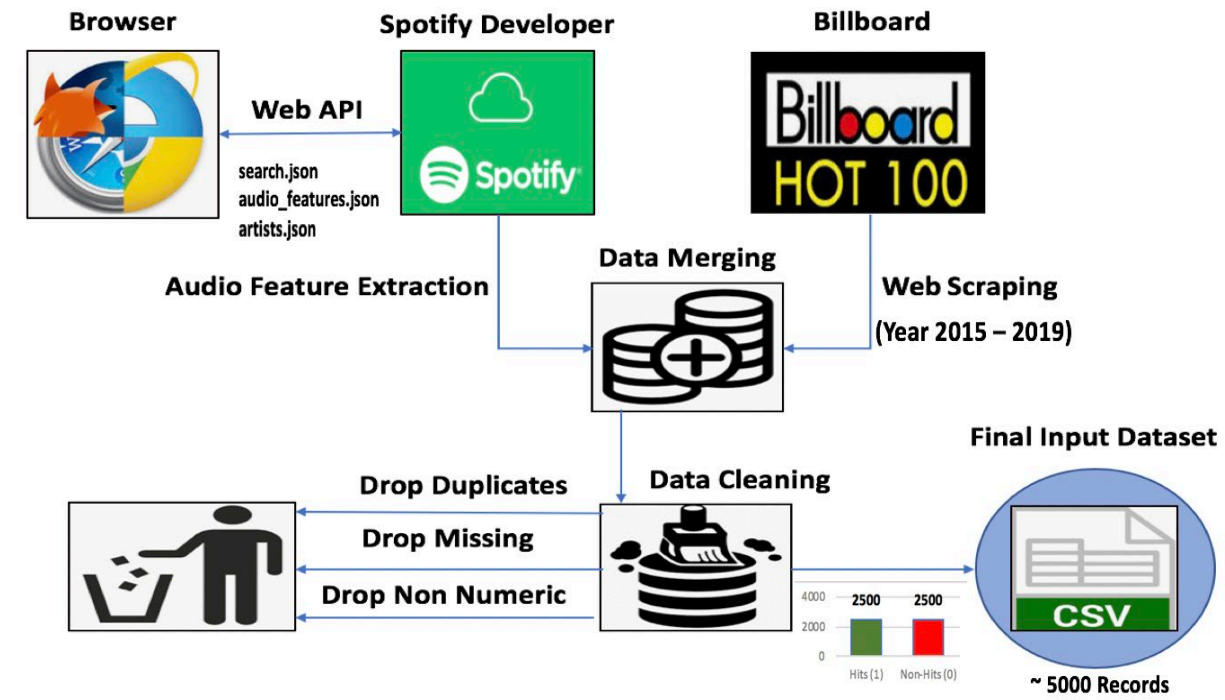


Figure - 1

[15] Bertin-Mahieux, Thierry & P. W. Ellis, Daniel & Whitman, Brian & Lamere, Paul. (2011). The Million Song Dataset. Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011). 591-596.

4.1. Data Collection, Wrangling, Merging and Cleaning: The first step of analysis involved data generation. Data was collected from multiple sources which were subsequently transformed, cleaned and merged as follows:

- **Billboards Hot-100:** Billboard Hot 100 is the music industry standard record chart in the United States and a popularity indicator. Data was scraped from the same using Python modules for web-scraping. [16] Cumulative hits were obtained for the year range 2015 – 2019 by extracting songs featured on the 1st day of each month (MM-01-YYYY). The entire data extracted were merged in a single CSV file and checked for duplicates which were removed. Data processing and cleaning operations were done using Python Pandas dataframe and the final cleaned data was stored in a CSV file.
- **Spotify Web API:** A developer account was created in **Spotify for Developers** platform[17] followed by creation of Client ID and Client Secret. Python library **"Spotipy"** [18] was used to generate access token followed by authentication using **OAuth2** authorization framework. After successful authentication, web API **"search"** was invoked to retrieve a list of tracks. For each track, web API **"audio_features"** was invoked with **"type=track"** to retrieve the list of audio features and **"type=artist"** to retrieve the list of artist features for each track. The artist features often included many special characters **(~, `, @, Ö, ^^, //, {})** which were handled and replaced with meaningful substitutions. This was essential for smooth invocation of API's. Finally, individual API results were aggregated and merged followed by removal of duplicate data and final creation of cleaned CSV file. This process was repeated for the year range **2015 – 2019**.
  Note: Spotify data had very little occurrence of missing / null values (< 0.1%). All missing values were replaced with the most common value(mode) for that particular attribute.
- **Merging Spotify and Billboards Dataframe:** Spotify and billboards CSV files were finally merged together. A categorical column named **"bbhot"** was appended to the Spotify CSV file. A value of **"1"** was added in the **"bbhot"** column if the track featured in the billboard's csv file and is considered as a **"hit"**. A value of **"0"** was added in the "bbhot" column if the track didn't feature in the billboard's CSV and is **not** considered a "hit". After merging, the final CSV file is checked once again for missing values, duplicate rows and special characters and handled accordingly.

The entire process of data collection, wrangling, cleaning and merging was done using **Python**.

4.2. Feature Engineering: Input data set consisted of the following 24 attributes [19]:

Attribute Name	Value Type	Value Description
duration_ms	int (numeric)	The duration of the track in milliseconds.
key	int (nominal)	The estimated overall key of the track. Integers map to pitches using standard pitch class notation. E.g. 0 =

[16] https://www.billboard.com/charts/year-end/2018/hot-100-songs
[17] https://developer.spotify.com/dashboard/
[18] https://spotipy.readthedocs.io
[19] https://developer.spotify.com/documentation/web-api/reference/

		C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
mode	int (numeric)	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
time_signature	int (numeric)	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
acousticness	float (numeric)	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	float (numeric)	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	float (numeric)	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
instrumentalness	float (numeric)	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveliness	float (numeric)	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
loudness	float (numeric)	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
speechiness	float (numeric)	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.

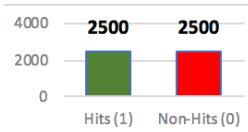
valence	float (numeric)	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
tempo	float (numeric)	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Id	string	The Spotify id of the track
uri	string	The Spotify URI for the track
artist_name	string	The name of the artist
track_name	string	The name of the track
popularity	int (numeric)	The Spotify popularity indicator (scale of 1-100) of a track based on the total number of plays compared to other tracks and how recent those plays are.
track_href	string	A link to the Web API endpoint providing full details of the track.
analysis_url	string	An HTTP URL to access the full audio analysis of this track. An access token is required to access this data.
type	string	The object type: “audio_features”
bbhot	int (categorical)	Either “1” or “0”. 1 indicates the song has featured in billboards hot 100 chart, 0 otherwise.
artist_popularity	int (numeric)	The popularity of the artist. The value will be between 0 and 100, with 100 being the most popular. The artist’s popularity is calculated from the popularity of all the artist’s tracks.
artist_followers	string (Format - href:total)	href: A link to the Web API endpoint providing full details of the followers; null if not available. Null values will be replaced with 0 in input file. total: The total number of followers.

Table - 1

Feature selection involved the following steps:

1) Non-numeric fields **'analysis_url', 'track_id', 'track_href', 'type', 'uri', 'artist_name', and 'track_name'** were dropped from the dataset. Detailed analysis revealed that even if these attributes are converted to numeric value using different encoding techniques which our predictive models can interpret, they will not have any contribution in determining the class label. Hence, they were removed.

2) Numeric field **"time_signature"** had very little importance in determining the class label. Outcome was mostly generalized and its presence in the dataset was bringing down the accuracy level of individual algorithms. Hence, the field was dropped.
3) Numeric field **"popularity"** was created by Spotify based on internal evaluation of how popular a song was. This was based on factors like total number of plays, recent plays etc. This field was dropped from the dataset because this was a pre-defined indicator of how popular a song was even before applying predictive modelling techniques. Moreover, this field was heavily weighted towards recent plays ignoring yesteryear hits producing biased results in algorithmic evaluation.
4) Not all features are created equal, and not all song features are measured equally. acousticness, for example, is measured from 1–10, while loudness is measured from -60–0. It's hard to compare these traits when they're on such different scales. This data was normalized to maintain parity. The final list of chosen attributes was normalized and transformed within the range of 0 – 1.
Note: Normalization of attributes was carried out using Python's "sklearn.preprocessing.MinMaxScaler" [20] function.



The final dataset contained around **~ 5000** records/instances and a list of **14** attributes (excluding class variable and depended variable) with normalized values. The dataset was created with 50% hit instances and 50% non-hit instances to ensure that the class variable is balanced.

Figure – 2

4.3 Use of Algorithms: The primary concept to be learned here is to accomplish the task of finding patterns and building a model which can be used to the predict popularity of a musical track relying on audio features of the track and details of the artist performing the same. Since we already extracted labeled data indicating past hits, a range of **supervised machine learning algorithms** will be used to train and build a model for predicting hits and non-hits. The same model will be validated against new unseen data and checked for accuracy of prediction.
Dimensionality Analysis Using PCA: Principal component analysis (PCA) revealed the following 14 attributes/features contributing to the overall variance and was retained as final input attributes for modelling:

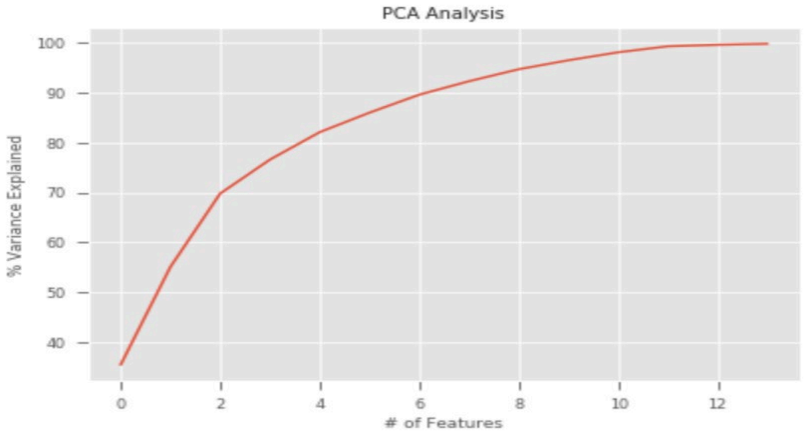


Figure – 3

[20] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Final Set of Input and Output Features Used for Modelling			
acousticness	energy	loudness	key
danceability	instrumentalness	speechiness	artist_followers
duration_ms	liveness	tempo	bbhot (Binary class variable for classification algorithms. It can have values “1” or “0” to indicate hits and non-hits)
valence	artist_popularity	mode	popularity (Dependent variable for regression algorithm. This is a continuous numeric attribute. This variable will be dropped for classification algorithms).

Table - 2

A range of supervised machine learning algorithms (classification) were employed on the input dataset to determine the value of binary class label **“bbhot”** as follows:

1) **K-Nearest-Neighbors (KNN):** KNN classification was used on the input dataset in three steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. K-Neighbors classifier model was fitted against training data set and tested against test dataset. The entire process was executed in a loop for **“K”** values ranging between 1-50. The final error rate for every K value was plotted in a graph and the best value of **“K”** is chosen accordingly.

   Step 2: With the best determined value of K, K-nearest neighbors classifier model was fitted against training data set and tested against test dataset and observations noted.

   Step 3: Step 2 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** K-nearest neighbors classifier implementation was carried out using python’s “sklearn.neighbors.KNeighborsClassifier” [21].

2) **Logistic Regression (LR):** LR classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. LR classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or

[21] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

underfitting. The chosen value of number of folds i.e. K=10. **Note:** Logistic Regression classifier implementation was carried out using python's "sklearn.linear_model.LogisticRegression" [22].

3) **Naïve Bayes (NB):** NB classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. Gaussian NB classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** Naïve Bayes classifier implementation was carried out using python's "sklearn.naive_bayes" [23].

4) **Support Vector Machines:** SVM classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. Linear SVC classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** SVM classifier implementation was carried out using python's "sklearn.svm.LinearSVC" [24].

5) **Decision Tree:** Decision Tree classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. Decision Tree classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** Decision Tree classifier implementation was carried out using python's "sklearn.tree.DecisionTreeClassifier" [25].

6) **Random Forest:** Random forest classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. Random Forest classifier model was fitted against training data set and tested against test dataset.

---

[22] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[23] https://scikit-learn.org/stable/modules/naive_bayes.html
[24] https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
[25] https://scikit-learn.org/stable/modules/tree.html

Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** Random forest classifier implementation was carried out using python's "sklearn.ensemble.RandomForestClassifier" [26].

7) **XG Boost:** Extreme gradient boosting classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. Gradient Boosting classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** XG Boost classifier implementation was carried out using python's "sklearn.ensemble.GradientBoostingClassifier" [27].

8) **Neural Network:** Multi-layer perceptron (MLP) classification was used on the input dataset in two steps as follows:

   Step 1: Input dataset was divided into training (70%) and testing (30%) datasets. MLP classifier model was fitted against training data set and tested against test dataset.

   Step 2: Step 1 was repeated for the training dataset with K-fold cross validation technique. This is used to assess the effectiveness of the model, especially for tackling overfitting or underfitting. The chosen value of number of folds i.e. K=10. **Note:** MLP classifier implementation was carried out using python's "sklearn.neural_network.MLPClassifier" [28].

   Apart from classification, regression was also used on the input dataset to determine the value of continuous numeric attribute **"popularity"**. The following algorithm was used for the same:

1) **Linear Regression:** Numeric field **"popularity"** is considered as the dependent variable here. Linear regression was carried out with multiple independent numeric input attributes to determine popularity value. Categorical variable "bbhot" (class variable for classification algorithms) is excluded from the input dataset for regression.

   For Linear regression, the input dataset in the is divided into training (70%) and testing (30%) datasets. LR model was fitted against training data set and tested against test dataset. **Note:** LR implementation was carried out using python's "sklearn.linear_model.LinearRegression" [29].

---

[26] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
[27] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[28] https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[29] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

5. Results:

5.1. Classification Algorithms: For classification algorithms, binary variable **"bbhot"** is the class variable. Spotify popularity indicator numeric attribute **"popularity"** is dropped from the input dataset for modelling as because the variable was already pre-defined by Spotify as a popularity indicator and its presence would have created significant bias in the modelling and determining output.

1) K-Nearest Neighbors: Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 58.47%
Confusion Matrix:
[[451 301]
 [322 426]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.60      0.59       752
           1       0.59      0.57      0.58       748

   micro avg       0.58      0.58      0.58      1500
   macro avg       0.58      0.58      0.58      1500
weighted avg       0.58      0.58      0.58      1500
```

```
Scores: [0.577664 0.59672  0.603128 0.642456 0.592976 0.569872 0.62856  0.655056
 0.547904 0.586936]
Mean: 0.6001272
Standard Deviation: 0.0316390910514193
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 60.01%
```

```
Best Hyper Parameters: {'algorithm': 'auto', 'leaf_size': 1, 'n_jobs': -1, 'n_neighbors': 41, 'weights': 'distance'}
```

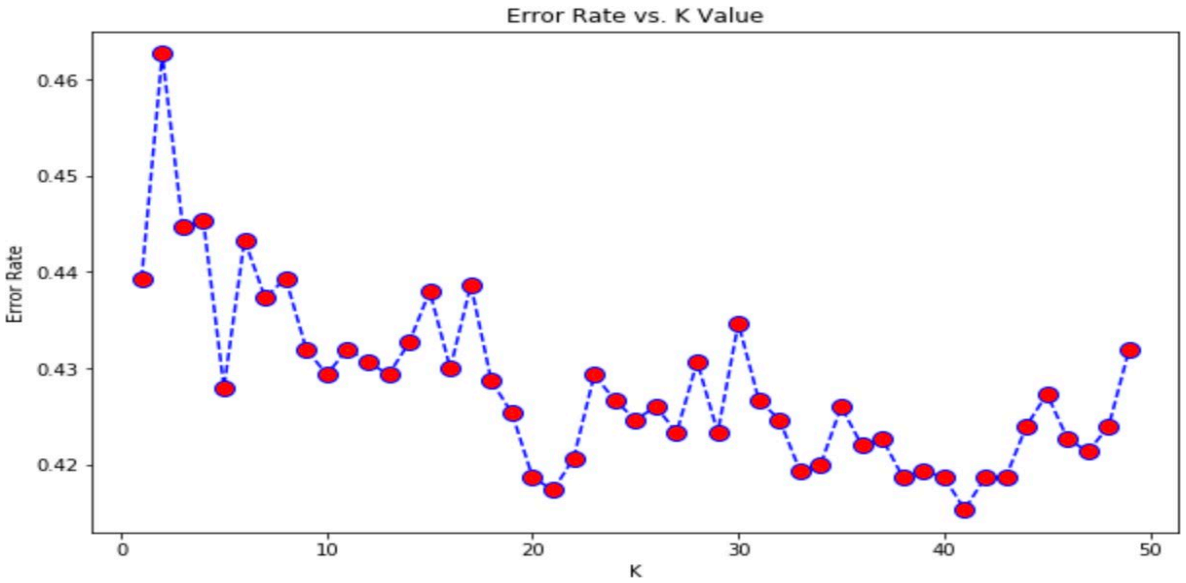


Figure – 4 (Best Value of K = 41)

2) **Logistic Regression:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 59.73%
Confusion Matrix:
[[452 305]
 [299 444]]
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.60      0.60       757
           1       0.59      0.60      0.60       743

   micro avg       0.60      0.60      0.60      1500
   macro avg       0.60      0.60      0.60      1500
weighted avg       0.60      0.60      0.60      1500


Scores: [0.64664  0.65896  0.669456 0.624    0.607056 0.683216 0.661696 0.648064
 0.634048 0.613344]
Mean: 0.6446479999999999
Standard Deviation: 0.023549588191728518
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 64.46%


Hyperparameters:  <bound method BaseEstimator.get_params of LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='liblinear',
          tol=0.0001, verbose=0, warm_start=False)>
```

3) **Naïve Bayes:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 55.07%
Confusion Matrix:
[[351 414]
 [260 475]]
Classification Matrix:
              precision    recall  f1-score   support

           0       0.57      0.46      0.51       765
           1       0.53      0.65      0.58       735

   micro avg       0.55      0.55      0.55      1500
   macro avg       0.55      0.55      0.55      1500
weighted avg       0.55      0.55      0.55      1500


Scores: [0.583936 0.63968  0.665568 0.620688 0.667184 0.612256 0.63088  0.614944
 0.623472 0.6084  ]
Mean: 0.6267007999999998
Standard Deviation: 0.02427483258356275
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 62.67%


Hyperparameters:  <bound method BaseEstimator.get_params of GaussianNB(priors=None, var_smoothing=1e-09)>
```

4) **Support Vector Machine:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 59.87%
Confusion Matrix:
[[476 252]
 [350 422]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.65      0.61       728
           1       0.63      0.55      0.58       772

   micro avg       0.60      0.60      0.60      1500
   macro avg       0.60      0.60      0.60      1500
weighted avg       0.60      0.60      0.60      1500


Scores: [0.610304 0.62104  0.632    0.651504 0.657264 0.661488 0.639504 0.65416
 0.64448  0.669072]
Mean: 0.6440816
Standard Deviation: 0.01759791034867493
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 64.41%


Prediction Accuracy with Grid Search: 62.47%
Best Hyperparameters for Grid Search:  SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

5) **Decision Tree:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 57.33%
Confusion Matrix:
[[449 313]
 [327 411]]
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.59      0.58       762
           1       0.57      0.56      0.56       738

   micro avg       0.57      0.57      0.57      1500
   macro avg       0.57      0.57      0.57      1500
weighted avg       0.57      0.57      0.57      1500


Scores: [0.572 0.546 0.59  0.568 0.552 0.55  0.556 0.578 0.558 0.536]
Mean: 0.5606
Standard Deviation: 0.0154415025175661
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 56.06%

Hyperparameters:  <bound method BaseEstimator.get_params of RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=None,
            oob_score=True, random_state=None, verbose=0, warm_start=False)>
```

6) **Random Forest:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 65.07%
Confusion Matrix:
[[496 266]
 [258 480]]
Classification Report:
              precision    recall  f1-score   support

           0       0.66      0.65      0.65       762
           1       0.64      0.65      0.65       738

   micro avg       0.65      0.65      0.65      1500
   macro avg       0.65      0.65      0.65      1500
weighted avg       0.65      0.65      0.65      1500


Scores: [0.732048 0.673672 0.677064 0.695808 0.705032 0.663992 0.681768 0.72784
 0.684656 0.678336]
Mean: 0.6920216
Standard Deviation: 0.02182076517081838
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 69.20%


Hyperparameters:  <bound method BaseEstimator.get_params of RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=600, n_jobs=None,
            oob_score=True, random_state=None, verbose=0, warm_start=False)>
```

7) **XG Boost:** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 63.93%
Confusion Matrix:
[[374 357]
 [380 389]]
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.51      0.50       731
           1       0.52      0.51      0.51       769

   micro avg       0.51      0.51      0.51      1500
   macro avg       0.51      0.51      0.51      1500
weighted avg       0.51      0.51      0.51      1500


Scores: [0.6848   0.711152 0.70512  0.707008 0.712384 0.65816  0.668816 0.740784
 0.696376 0.716752]
Mean: 0.7001351999999998
Standard Deviation: 0.02294349412273551
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 70.01%


Hyperparameters:  <bound method XGBModel.get_params of XGBClassifier(base_score=0.5, booster='gbtree', colsample_byle
vel=1,
       colsample_bytree=0.8, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=5, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=4, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=27, silent=True,
       subsample=0.8)>
```

8) **Neural Network (MLP Classifier):** Sample results are depicted below:

```
Prediction Accuracy Without Cross Validation: 60.32%
Confusion Matrix:
[[1589  911]
 [1073 1427]]
Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.64      0.62      2500
           1       0.61      0.57      0.59      2500

    micro avg       0.60      0.60      0.60      5000
    macro avg       0.60      0.60      0.60      5000
weighted avg       0.60      0.60      0.60      5000


Scores: [0.699952 0.673104 0.67072  0.618768 0.672256 0.6604   0.651664 0.634528
 0.652608 0.621936]
Mean: 0.6555936
Standard Deviation: 0.024001876406647873
Prediction Accuracy With 10-Fold Cross Validation(Mean Value): 65.56%


MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(5, 2), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
       random_state=1, shuffle=True, solver='lbfgs', tol=0.0001,
       validation_fraction=0.1, verbose=False, warm_start=False)
```

5.2 Regression Algorithm(s): For linear regression, continuous numeric attribute **"popularity"** is the dependent variable. Binary class variable **"bbhot"** is dropped from the input dataset for regression.

Linear Regression: Sample results are depicted below:

	Coeff
artist_popularity	24.867221
artist_followers	10.075167
danceability	6.727067
loudness	4.791572
acousticness	4.275859
energy	2.865170
tempo	1.356947
key	1.126680
valence	1.005988
mode	0.273224
instrumentalness	-0.441964
duration_ms	-2.182433
liveness	-3.759478
speechiness	-5.819595

Table – 3

Metrics	
MAE	6.663270
MSE	72.05261
RMSE	8.488381

Table - 4

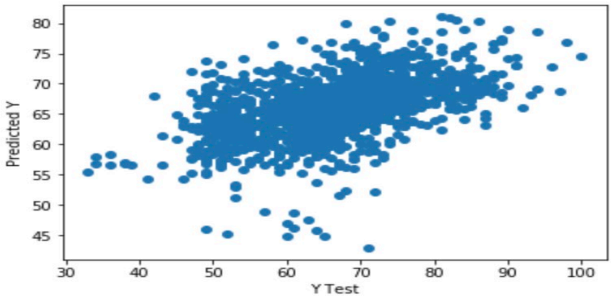


Figure – 5

5.3 Graphs:

ROC curve for Random Forest Classifier

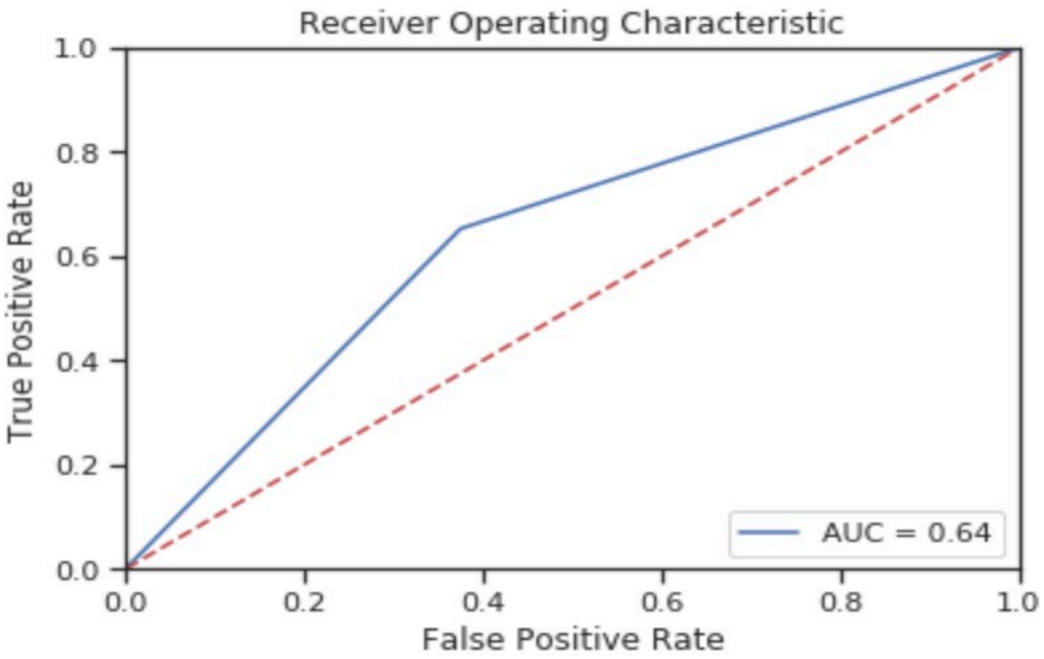


Figure – 6

Confusion Matrix for Random Forest Estimator

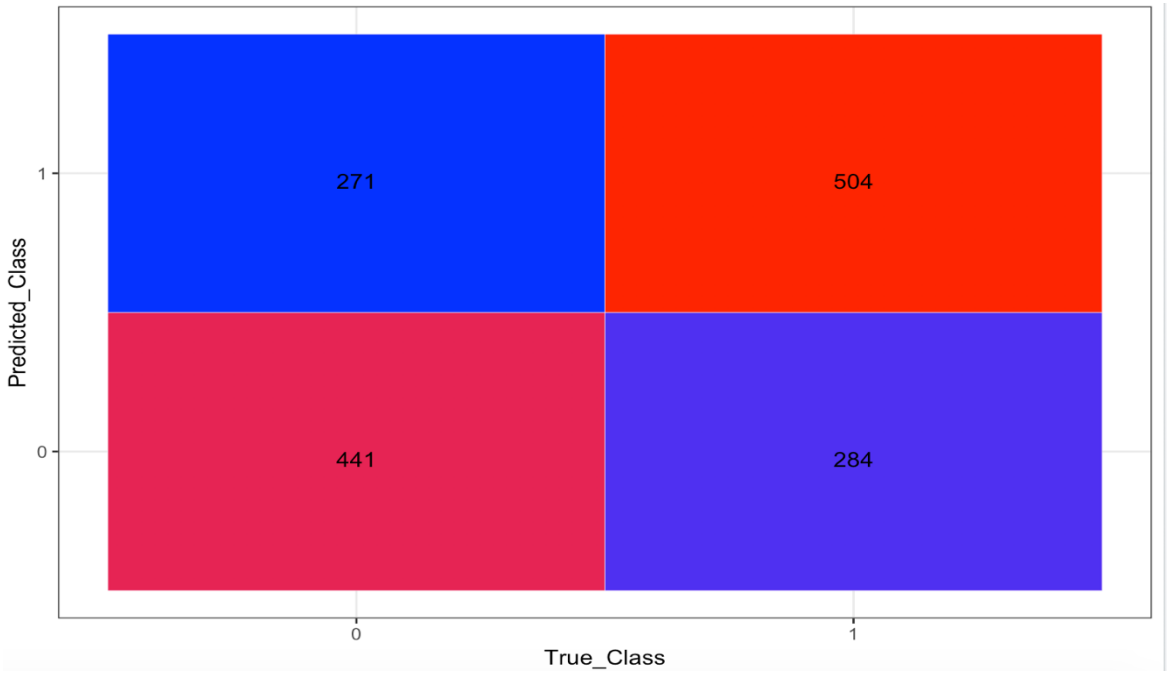


Figure – 7 [Class 1 = Hit, Class 0 = Non-Hit]

Pearson Correlation Coefficient Heatmap – Prepared Using Seaborn Library

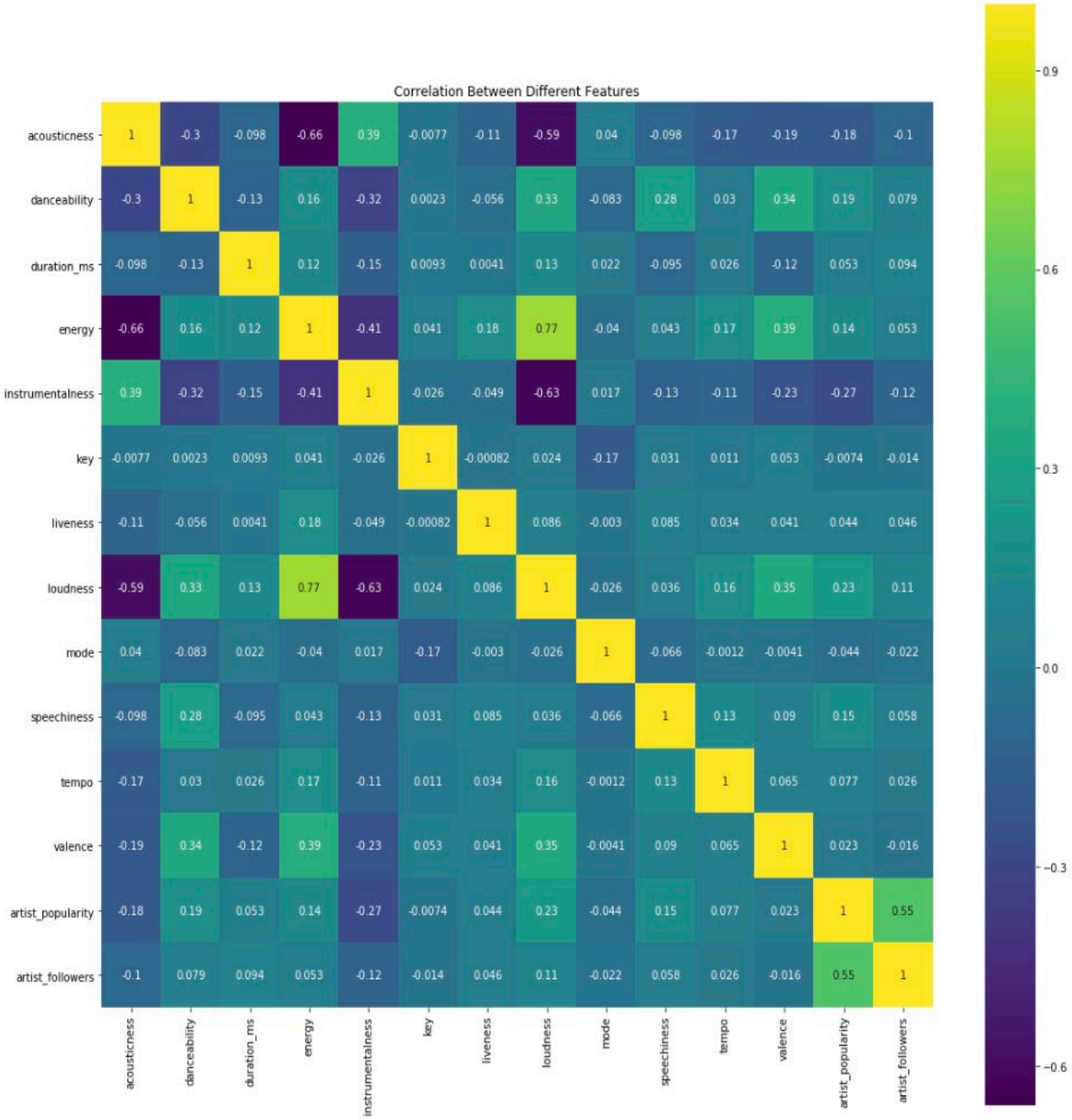


Figure - 8

	importance
artist_followers	0.126025
artist_popularity	0.099698
speechiness	0.091498
duration_ms	0.081139
danceability	0.080276
loudness	0.079845
tempo	0.079845
acousticness	0.072508
liveness	0.072076
valence	0.071644
energy	0.057833
instrumentalness	0.047475
key	0.033233
mode	0.006905

Table – 5

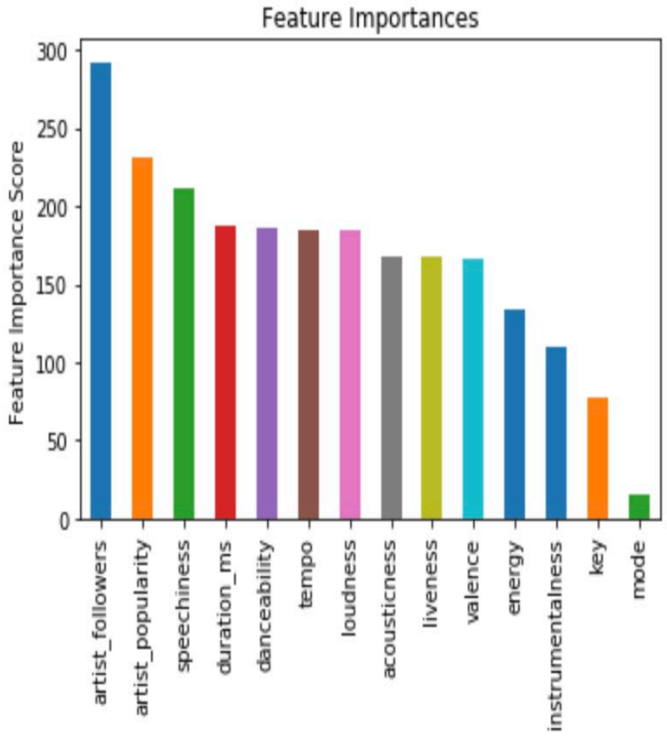


Figure – 9

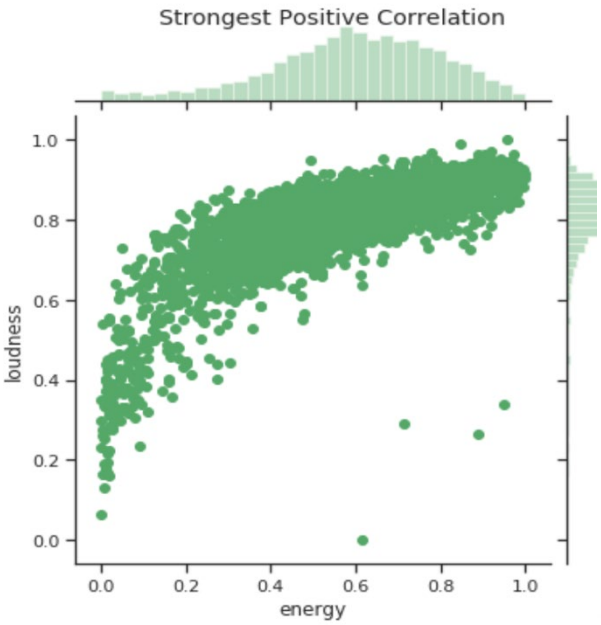


Figure – 10

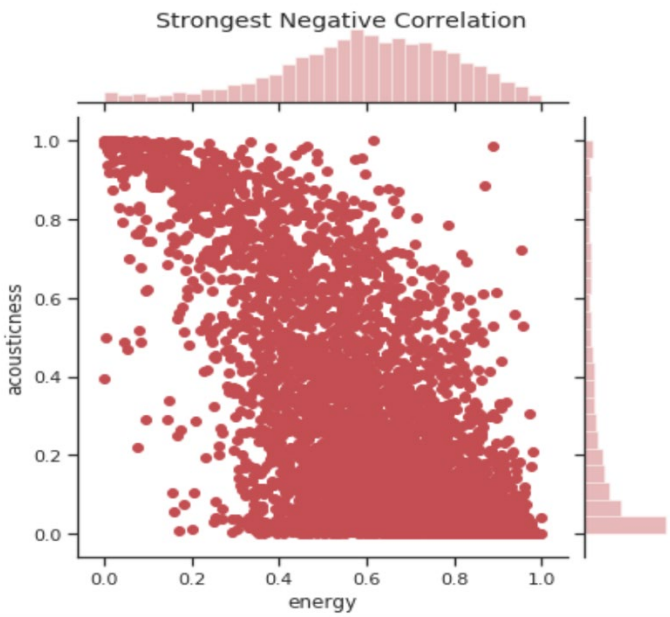


Figure – 11

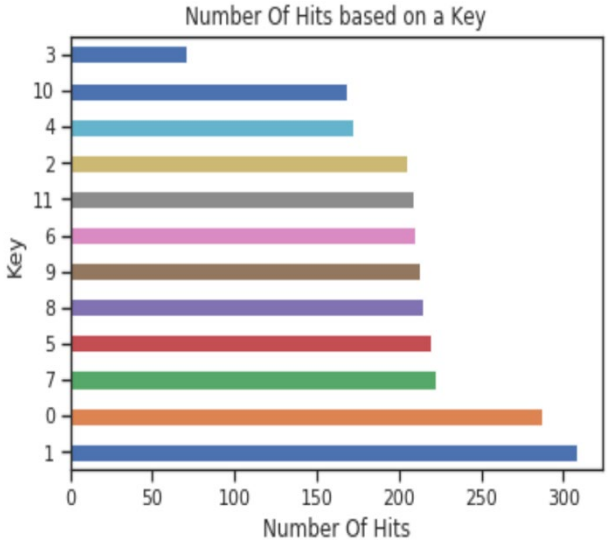


Figure – 12
Note: Most hits in key of C and C#

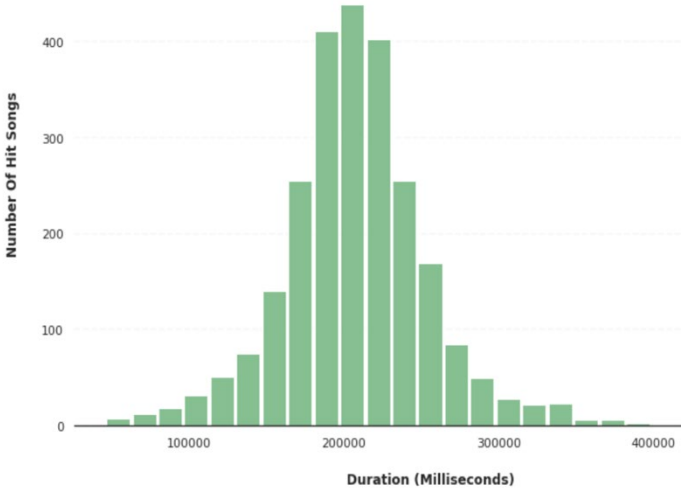


Figure - 13
Note: Most hits in duration of 3 – 3.5 mins

Chord Diagram Showing Correlations Between Different Attributes – Using R Package Circlize

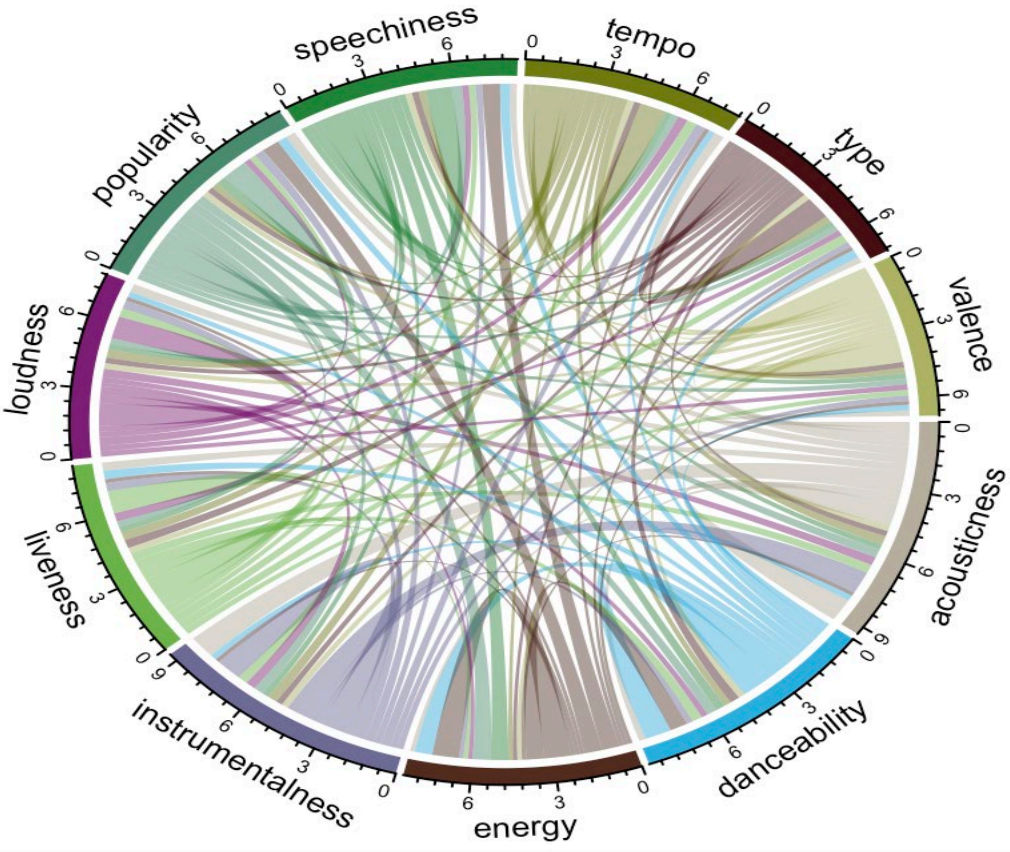


Figure - 14

5.4 Performance Results Summary for classification algorithms:

Algorithm	Overall Model Accuracy (Mean Score Of 10-Fold Cross Validation)
XG-Boost	70.01%
Random Forest	69.20%
Neural Network (Multilayer Perceptron)	65.56%
Logistic Regression	64.46%
Support Vector Machines	64.41%
Naïve Bayes	62.67%
K-Nearest Neighbors	60.01%
Decision Tree	56.06%

Table – 6

Algorithm	Precision (Class=1)	F1-Score (Class=1)
Random Forest	64%	65%
Neural Network (Multilayer Perceptron)	61%	59%
XG-Boost	51%	51%
Logistic Regression	60%	60%
Support Vector Machines	63%	58%
Naïve Bayes	53%	58%
K-Nearest Neighbors	59%	58%
Decision Tree	57%	56%

Table – 7

5.5 Statistical Results: The following table displays basic descriptive statistics of the final set of input attributes used for modelling (zoom to view):

Attributes	count	mean	std	min	25%	50%	75%	max
acousticness	5000	0.266222	0.284436	0.000003	0.038775	0.153000	0.415000	0.996000
danceability	5000	0.650610	0.161101	0.000000	0.552000	0.667000	0.767000	0.972000
duration_ms	5000	2.038833e+05	5.151627e+04	3.524000e+04	1.771382e+05	2.017070e+05	2.281582e+05	1.355938e+06
energy	5000	0.593527	0.203449	0.000900	0.474000	0.609000	0.741250	0.998000
instrumentalness	5000	0.065873	0.216466	0.000000	0.000000	0.000000	0.000182	1.000000
key	5000	5.275800	3.618612	0.000000	2.000000	5.000000	8.000000	11.00000
liveness	5000	0.171321	0.126541	0.019600	0.098575	0.120000	0.199000	0.979000
loudness	5000	-7.430599	4.247411	-41.53700	-8.463250	-6.443000	-5.010500	0.605000
mode	5000	0.58780	0.49228	0.00000	0.00000	1.00000	1.00000	1.00000
speechiness	5000	0.124946	0.118186	0.000000	0.041400	0.069400	0.175250	0.966000
tempo	5000	120.346522	30.120835	0.000000	95.980000	119.97550	141.98700	220.0990
valence	5000	0.436074	0.219097	0.000000	0.268000	0.422000	0.592000	0.990000
artist_popularity	5000	73.737000	14.919972	0.000000	65.000000	75.000000	85.000000	100.0000
artist_followers	5000	3.206522e+06	5.972459e+06	0.000000e+00	1.389320e+05	7.818130e+05	3.138132e+06	4.368682e+07

Table - 8

5.6 Result Analysis: The primary goal of this project was to try and build a predictive model which can be used to predict the popularity of a song based on its audio features before it is released commercially to the wider listeners. Also, it was important to know the set of features and their estimated range of values which significantly contributes to the popularity index. The best model was identified as a combination of prediction accuracy and f1-scores. The idea was mostly derived from the concepts gathered from literature review, section **[3]**.

Prediction accuracy can be defined as the percentage of correct predictions made by the trained model on test data. Mathematically it is defined as:

$$\textbf{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

Precision was considered as another important measure of the effectiveness of the model. It was important to keep the number of false positives to minimal because as a music producer, investing on a record label which will eventually not be a commercial success (non-hit) is not desirable. Similarly, a false negative could lead to a potentially popular song being rejected. Mathematically precision is defined as:

$$\textbf{Precision} = (TP)/(TP+FP)$$

Note: TP = True positive; FP = False positive; TN = True negative; FN = False negative

The problem was approached by dividing the input dataset into training (70%) and testing (30%) and applying range of known supervised machine learning classification algorithms. To avoid overfitting, 10-fold cross-validation was employed further. Ensemble methods like random forest and extreme gradient boosting (XGBoost) performed better than the other baseline models like SVM, Naïve Bayes, logistic regression, K-NN etc. Neural network (MLP classifier) was also used to train the input dataset which was close to the performance of ensemble classifiers. XG-Boost had the best accuracy **(~ 71% - best CV fold score)** and random forest had the best precision score for hits **(~64%).** Third best model overall was neural network. Please refer to section **[5.1]** and **[5.4]** for detailed and summarized results. ROC plot for the same can be referred from section **[5.3].** For XG-boost and random forest, the best features contributing to the output were extracted and plotted. Please refer to section **[5.3]** for the graphs. The top six most important features were ***"artist_followers, artist_popularity, speechiness, duration_ms, danceability and tempo"***. These feature importance scores were quite intuitive considering the fact that if an artist is popular and has large number of followers, there is a strong likelihood of his or her future songs attaining significant popularity. Similarly, the chances of fast paced, energetic, danceable songs becoming trendsetters in recent times was reconfirmed from the feature importance scores.

The ensemble models performed better overall because the data was highly non-linear. As an ensemble method, random forest is known to reduce bias by taking majority vote of a large number of decision trees. Hyperparameter tuning was done to a certain extent for the classifiers as depicted below based on trial and error combinations (tuned parameters in **bold**):

Algorithm	Hyperparameters
Random Forest	**n_estimators = 600,** min_samples_leaf=1, min_samples_split=2,
XG-Boost	**learning_rate =0.1, n_estimators=100, max_depth=5, min_child_weight=1, gamma=0, subsample=0.8, c**

	olsample_bytree=0.8, objective= 'binary:logistic', nthread=4, scale_pos_weight=1, seed=27
Multi-Layer Perceptron NN	activation = relu, **solver='lbfgs'**
K-Nearest Neighbors	**n_neighbors = 41**, algorithm='auto', n_jobs = -1
Logistic Regression	C=1.0, **solver='liblinear'**
Decision Tree	min_samples_leaf=1, min_samples_split=2, criterion='gini'
Support Vector Machines	gamma=1, **kernel='rbf', C=10**

Table – 9

Please refer to section **[5]** for detailed hyperparameters results for all algorithms.

For regression algorithm(s), section **[5.2],** the regression coefficients indicated that keeping everything else constant, just by increasing one unit of artist_popularity will increase **24** units of popularity measure which is significantly larger than all other coefficients. However, the Spotify popularity indicator is not a constant value. It keeps changing based on the number of recent plays. So, although the audio features indicated some sort of correlation between input features and dependent output variable "popularity", this is not a viable model because of the dynamically changing value of popularity based on external factors.

6. Conclusion & Future Recommendations:

- **XG-Boost** and **random forest** were the best performing model in terms of mean accuracy measure of around **70%** after 10-fold cross validation. Random forest also performed best in terms of precision of identifying hits.
- Model predicted non-hits marginally better than hits. Model predicted hits better for popular artists i.e. artists whose Spotify popularity score was 70+. Precision was also better for non-hits for all the different types of models.
- Naïve Bayes produced surprisingly good results which established the fact that every attribute had a significant individual weightage in determining outcome.
- **Principle component analysis (PCA)** revealed a set of **14 attributes** contributing to the variance in determining output which were used for final modelling.
- **Artist popularity** played a major role in making a song hit. Other strong signals are danceability, duration, speechiness, tempo etc. Higher values of these attributes ensured stronger possibility of the song becoming a hit.
- **Duration** of most of the hit songs typically ranged between **3 – 3.5** minutes. Most of hits are in the key of **C and C#** major scales.
- **Decision tree** was one of the worst performers due to severe **overfitting**.
- More feature engineering (introduction of other features like **genre, lyrics** etc.), hyper parameter tuning, grid search cross validation or randomized grid search for every algorithm will enhance modelling performance further.
- More hits need to be labelled apart from just billboards which typically lists pop hits. Input dataset needs to be diversified for modelling and predicting yesteryear hits. **The current model holds good in determining only recent (last 5 years) trend of hit songs.**
- More diversified dataset needed combining hits from different types of genres to enhance generalization performance of the final model.
- More combinations of K-Fold cross validations, different percentage of splits for input dataset etc. can be tried out to obtain different results.

7. Project Log:

Date	Timespan	Description
02/15/2019	2 Hours	Project Topic Research
02/18/2019	2 Hours	Project Topic Research
02/19/2019	2 Hours	Project Topic Research
02/20/2019	2.5 Hours	Data Collection
02/22/2019	2.5 Hours	Data Collection
02/23/2019	3 Hours	Data Collection
02/24/2019	3 Hours	Data Collection
02/26/2019	4 Hours	Proposal Preparation
02/27/2019	3 Hours	Proposal Preparation
02/28/2019	3 Hours	Proposal Preparation
03/01/2019	3 Hours	Proposal Preparation
03/03/2019	3 Hours	Proposal Preparation
03/11/2019	3 Hours	Develop/Coding/Testing
03/12/2019	3 Hours	Develop/Coding/Testing
03/15/2019	2 Hours	Develop/Coding/Testing
03/25/2019	2 Hours	Develop/Coding/Testing
04/02/2019	2 Hours	Develop/Coding/Testing
04/07/2019	2 Hours	Develop/Coding/Testing
04/10/2019	2 Hours	Develop/Coding/Testing
04/11/2019	3 Hours	Develop/Coding/Testing
04/15/2019	3 Hours	Develop/Coding/Testing
04/16/2019	4 Hours	Develop/Coding/Testing
04/17/2019	5 Hours	Develop/Coding/Testing
04/20/2019	4 Hours	Develop/Coding/Testing
04/22/2019	3 Hours	Develop/Coding/Testing
04/27/2019	3 Hours	Develop/Coding/Testing
04/30/2019	3 Hours	Poster Preparation
05/01/2019	2 Hours	Develop/Coding/Testing
05/03/2019	3 Hours	Final Documentation
05/05/2019	3 Hours	Final Documentation
05/07/2019	4 Hours	Final Documentation
05/08/2019	3 Hours	Final Documentation
05/09/2019	3 Hours	Final Documentation
05/10/2019	3 Hours	Final Documentation
05/11/2019	4 Hours	Final Documentation
05/12/2019	3 Hours	Final Documentation

Total Number of Hours: **106 Hours**

8. Appendix:

8.1. Tools and Technologies: The following set of tools and technologies were used for this project:

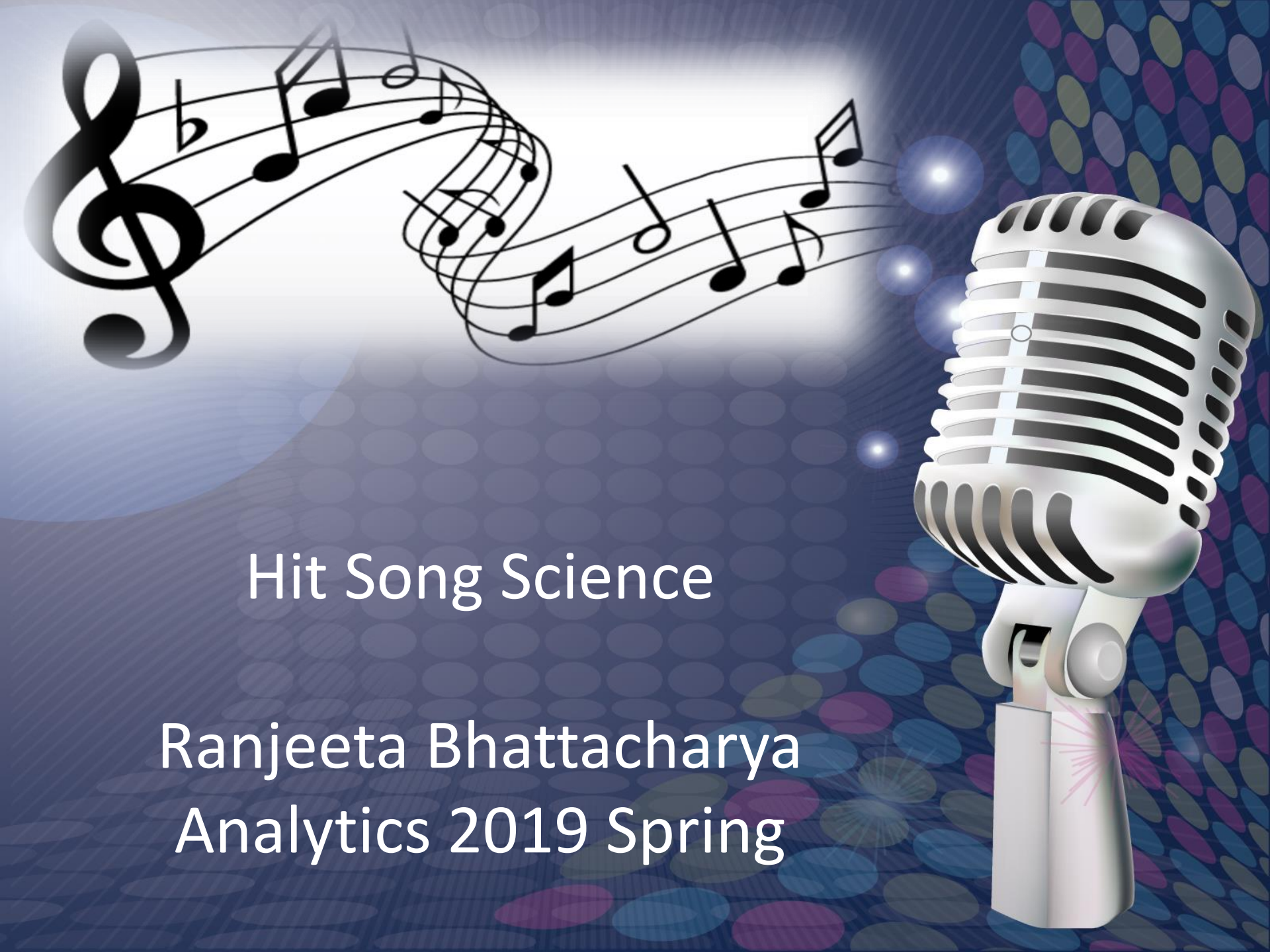
- Python scripting for web-scraping billboards.com (hot-100 chart)
- Python scikit-learn for performing machine learning
- Python Pandas, NumPy for data collection, wrangling, merging and cleaning operations.
- Python Matplotlib, Seaborn for visualization
- R circlize, ggplot2 for visualization
- Anaconda Jupyter Notebook for interactive scripting

8.2. Spotify API's: Complete API description used for retrieving data are listed below:

- **search: [Link]**
- **audio_features: [Link]**
- **artists: [Link]**

8.3. File: The input file used for this project is attached below:

Note: The file extension is renamed from **"csv"** to **"xlsx"** for attaching in this word document. File will be uploaded in drop box also.


Hit Song Science
Ranjeeta Bhattacharya
Analytics 2019 Spring

Introduction

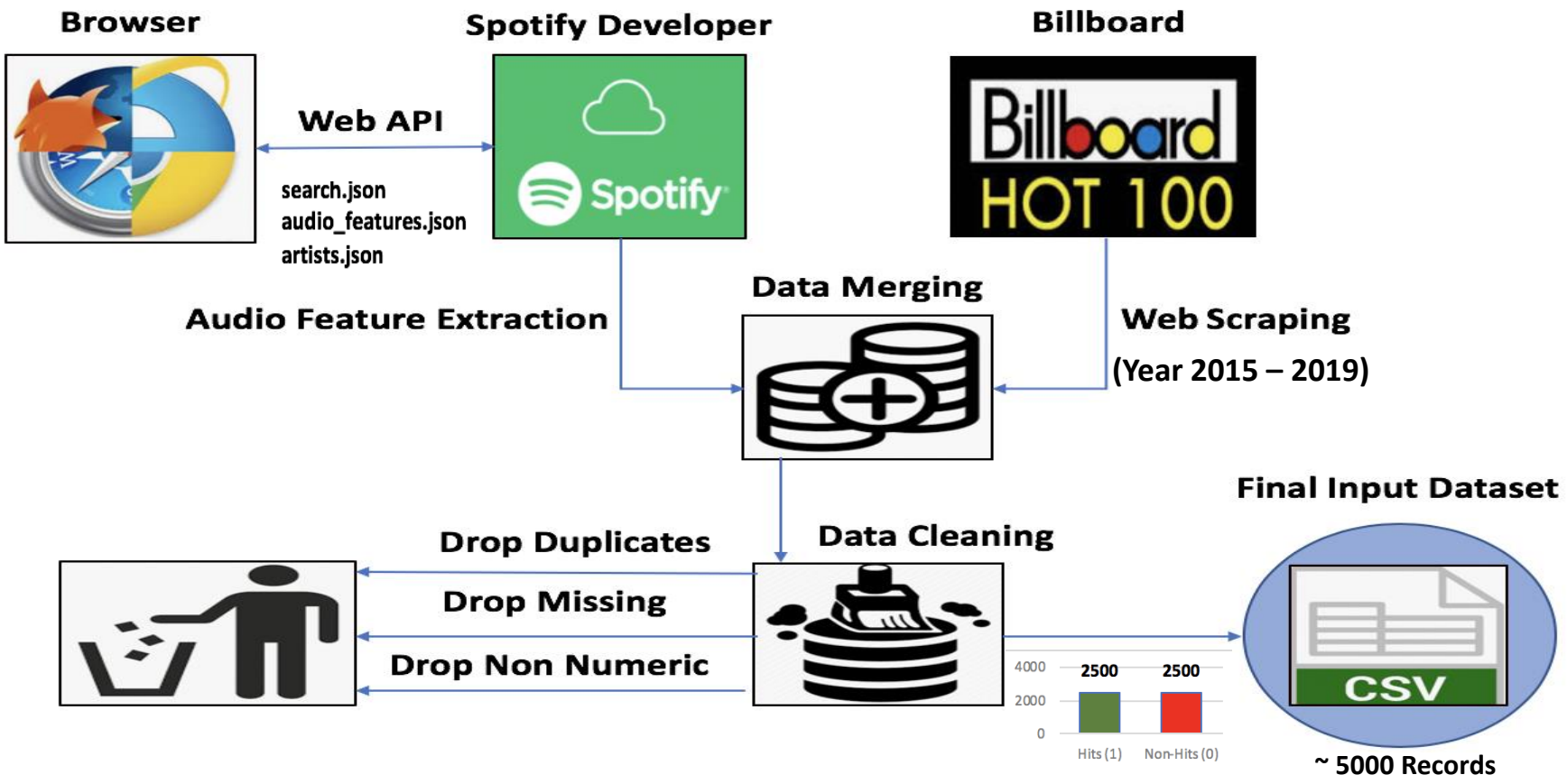
Music is the universal language of communication and plays an integral part of most of our lives. Where words fail, music prevails. Writing a hit song is an art form. Many artists continue to search that perfect formula which will enable them to create kind of music which will appeal to the listeners ultimately turning into great hits. The purpose of this project is to try and investigate and analyze on these lines:

- *Are there certain characteristics for hit songs? Is it possible to predict the popularity of a song based on certain set of features?*
- *What are the set of audio features which actually contributes to the success of a track?*
- *How important is the individual popularity score of an artist in making the song a hit?*
- *Is there any particular genre which produces more hit songs than the rest?*

Concept

To accomplish the task of finding patterns and building a model which can be used to the predict popularity of a musical track, different supervised machine learning algorithms are used. The features of an audio track are used for building the model. A range of parameters are considered for evaluation.

Data Preparation High-Level Overview



Final Set Of Audio Features Used For Modelling		
acousticness	energy	loudness
danceability	instrumentalness	speechiness
duration_ms	liveness	tempo
valence	artist_popularity & artist_followers	mode
key	bbhot (Binary class variable for Classification Algorithms)	popularity (Dependent Variable for Linear Regression. Attribute dropped for classification algorithms to remove bias).

Data Extraction

Relevant data is extracted directly from multiple sources as follows:

- **Spotify** : Online music streaming services platform which exposes a comprehensive set of Application Programming Interfaces (API's) which are suitably invoked to extract relevant data in the form of a JSON object . **"search", "audio_features"** and **"artists"** API's are invoked.
- **Billboard Hot 100** : Popular American entertainment media brand which is known for publishing music charts via their website . Details of popular songs appearing on the billboard Hot-100 chart are directly extracted (web-scraping) and compared with Spotify data.

Data Merging

Billboard Hot 100 list is matched with every song extracted from Spotify. If the song appears in the Billboard list, it is considered popular and tagged with value "1", "0" otherwise. Finally, the data is aggregated and merged to prepare the input dataset in CSV format for processing.

Data Preprocessing and Cleaning

- For many artists, corresponding tracks are there where no audio features are available. Those rows are entirely removed from the dataset.
- For many songs duplicate entries were there owing to the fact that the same song appeared under multiple albums. They were removed.

Data Preprocessing and Cleaning (Contd..)

- Since Spotify data is already organized, instances containing missing values are less. Typically, the missing values are replaced with the most common value of that particular attribute.
- Not all features are created equal, and not all song features are measured equally. acousticness, for example, is measured from 1–10, while loudness is measured from -60–0. It's hard to compare these traits when they're on such different scales. This data is normalized to maintain parity.

Post pre-processing, input dataset is split into 70% (training set) - 30% (testing set).

Note: Input dataset is formed by aggregating data from last 5 years (2015 – 2019) with a combination of recognized hits and non-hits to ensure that the class variable is balanced and not skewed. This is essential to improve prediction accuracy. Around ~ 5000 instances (~2500 hits and ~2500 non-hits) are selected for modelling purpose. A sample snapshot input file is attached for reference (zoom to view):

acousticness	analysis_url	danceability	duration_ms	energy	track_id	instrumentali	key	liveness	loudness	mode	speechiness	tempo	time_signatu	track_href	type	uri	valence	artist_name	track_name	popularity	bbhot	artist_popula	artist_followers
0.0405	https://api.sp	0.884	191938	0.698	2EgB4n6XyBs	0	0	0.195	-9.101	1	0.364	140.068	4	https://api.sp	audio_featur	spotify:track:	0.575	Rich The Kid	New Freezer	76	0	86	1585212
0.0375	https://api.sp	0.852	174653	0.691	0QVISdPWRI	0	6	0.0517	-5.175	0	0.139	61.115	4	https://api.sp	audio_featur	spotify:track:	0.658	Plies	Rock	59	1	68	870286
0.0217	https://api.sp	0.54	204991	0.381	3h3pOvw6hjI	0.000687	7	0.305	-14.323	1	0.0347	155.882	4	https://api.sp	audio_featur	spotify:track:	0.343	girl in red	i wanna be yo	65	0	65	197902
0.0141	https://api.sp	0.817	210368	0.539	116H0KvKr2Z	0.000496	6	0.099	-6.349	0	0.0621	97.062	4	https://api.sp	audio_featur	spotify:track:	0.158	Bad Bunny	MIA (feat. Dr	94	0	93	12415147
0.682	https://api.sp	0.452	255205	0.298	6MXJdqTuSD	0	2	0.128	-11.292	1	0.0444	77.491	5	https://api.sp	audio_featur	spotify:track:	0.158	Aquilo	Sorry	67	1	64	176979
0.0227	https://api.sp	0.739	189000	0.742	05mAIVLkIW	1.39E-06	7	0.229	-4.586	1	0.0329	124.016	4	https://api.sp	audio_featur	spotify:track:	0.659	Charli XCX	1999	82	1	78	1551587
0.000958	https://api.sp	0.493	210900	0.998	4Xc169q96Ja	1.86E-06	4	0.252	-3.487	0	0.109	74.99	4	https://api.sp	audio_featur	spotify:track:	0.437	Five Finger D	Fake	57	1	79	3106228
0.00329	https://api.sp	0.417	214387	0.669	2N977oXVcb	0.000169	1	0.0403	-5.544	1	0.0296	76.988	4	https://api.sp	audio_featur	spotify:track:	0.539	Wallows	Ground	59	1	72	267374
0.0416	https://api.sp	0.746	133407	0.622	2l4zJarZ3jSih	0	9	0.0933	-5.976	1	0.0533	108.015	4	https://api.sp	audio_featur	spotify:track:	0.361	Smooky Mar	Vlone Flex	64	0	63	87094

File Attribute Details (Zoom to view)

Attribute Name	Value Type	Value Description
duration_ms	int (numeric)	The duration of the track in milliseconds.
key	int (nominal)	The estimated overall key of the track. Integers map to pitches using standard pitch class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
mode	int (numeric)	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
time_signature	int (numeric)	An estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure).
acousticness	float (numeric)	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
danceability	float (numeric)	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
energy	float (numeric)	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
instrumentalness	float (numeric)	Predicts whether a track contains no vocals. “Ooh” and “aah” sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly “vocal”. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveliness	float (numeric)	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
loudness	float (numeric)	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.
speechiness	float (numeric)	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
valence	float (numeric)	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
tempo	float (numeric)	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
Id	string	The Spotify id of the track
uri	string	The Spotify URI for the track
artist_name	string	The name of the artist
track_name	string	The name of the track
popularity	int (numeric)	The Spotify popularity indicator (scale of 1-100) of a track based on the total number of plays compared to other tracks and how recent those plays are. **[Dependent Variable for Linear Regression]**
track_href	string	A link to the Web API endpoint providing full details of the track.
analysis_url	string	An HTTP URL to access the full audio analysis of this track. An access token is required to access this data.
type	string	The object type: “audio_features”
bbhot	int (categorical)	Either “1” or “0”. 1 indicates the song has featured in billboards hot 100 chart, 0 otherwise. **[Class Variable for Classification Algorithms]**
artist_popularity	int (numeric)	The popularity of the artist. The value will be between 0 and 100, with 100 being the most popular. The artist’s popularity is calculated from the popularity of all the artist’s tracks.
artist_followers	string (Format -href:total)	**href:** A link to the Web API endpoint providing full details of the followers; null if not available. Please note that this will always be set to null, as the Web API does not support it at the moment. **total**: The total number of followers.

Model Selection & Evaluation (Supervised Learning)

K-NN (K=41)

Mode	Accuracy
70-30 train-test split	~ 59%
With 10-Fold CV	~ 60% (Mean Score)

SVM

Mode	Accuracy
70-30 train-test split	~ 60%
With 10-Fold CV	~ 65% (Mean Score)

Logistic Regression

Mode	Accuracy
70-30 train-test split	~ 60%
With 10-Fold CV	~ 65% (Mean Score)

Naïve Bayes

Mode	Accuracy
70-30 train-test split	~ 55%
With 10-Fold CV	~ 63% (Mean Score)

Neural Network

Mode	Accuracy
70-30 train-test split	~ 62%
With 10-Fold CV	~ 65% (Mean Score)

Decision Trees

Mode	Accuracy
70-30 train-test split	~ 55%
With 10-Fold CV	~ 57% (Mean Score)

Random Forest

Mode	Accuracy
70-30 train-test split	~ 65%
With 10-Fold CV	~ 69% (Mean Score)

XG Boost

Mode	Accuracy
70-30 train-test split	~ 64%
With 10-Fold CV	~ 70% (Mean Score)

Model Selection & Evaluation (Cont..)

Algorithm	Accuracy (10-Fold CV Mean Score)
XG Boost	70.01%
Random Forest	69.20%
Neural Network	65.56%
Logistic Regression	64.46%
Support Vector Machines	64.41%
Naïve Bayes	62.67%
K-Nearest Neighbors	60.01%
Decision Tree	56.06%

Linear Regression

Metrics	
MAE	6.663270
MSE	72.05261
RMSE	8.488381

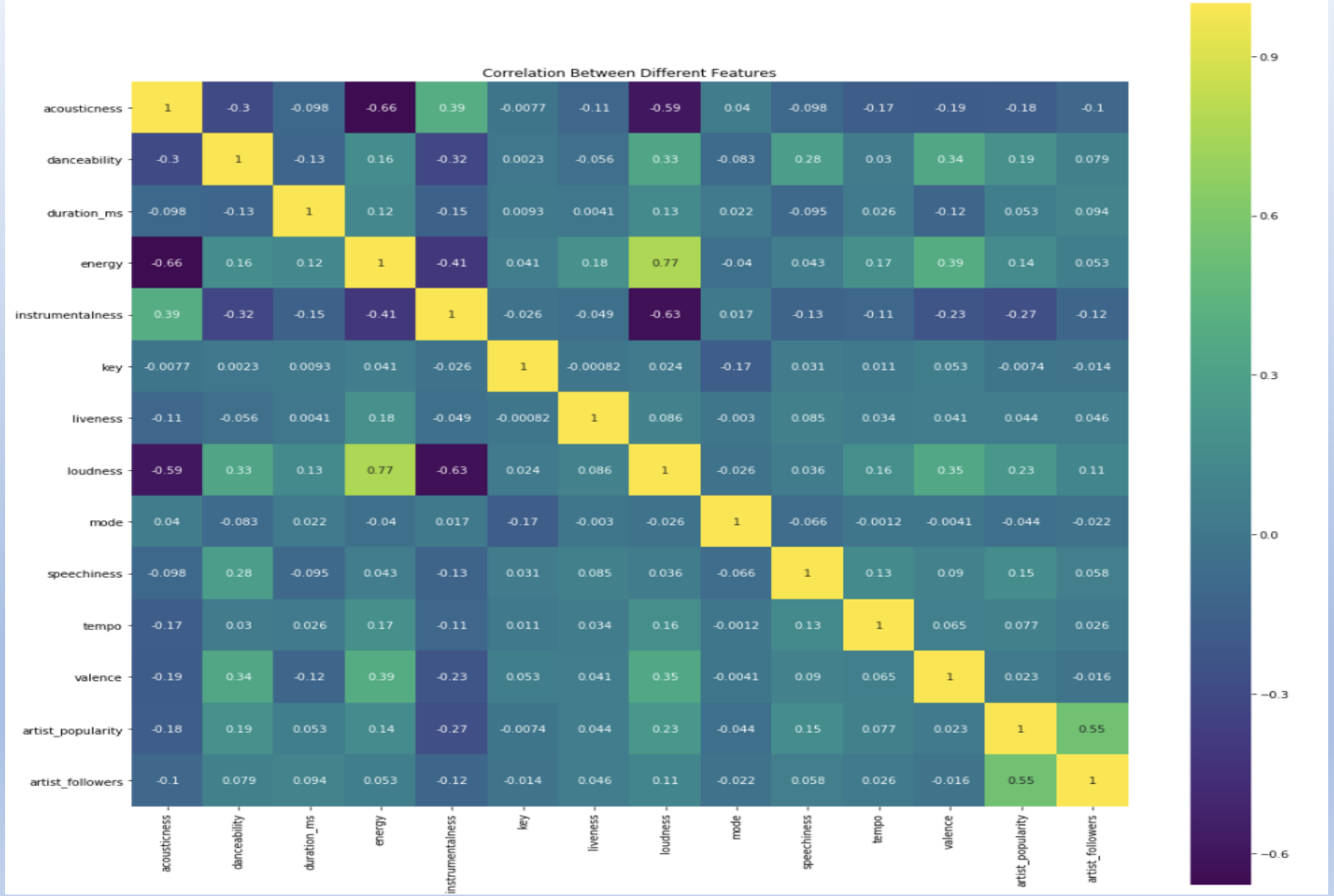
	Coeff
artist_popularity	24.867221
artist_followers	10.075167
danceability	6.727067
loudness	4.791572
acousticness	4.275859
energy	2.865170
tempo	1.356947
key	1.126680
valence	1.005988
mode	0.273224
instrumentalness	-0.441964
duration_ms	-2.182433
liveness	-3.759478
speechiness	-5.819595

Tools and Technologies

- Python scripting for web-scraping billboards.com (hot-100 chart)
- Python scikit-learn for performing machine learning
- Python Pandas, NumPy for data collection, wrangling, merging and cleaning operations.
- Python Matplotlib, Seaborn for visualization
- R circlize, ggplot2 for visualization
- Anaconda Jupyter Notebook for interactive scripting

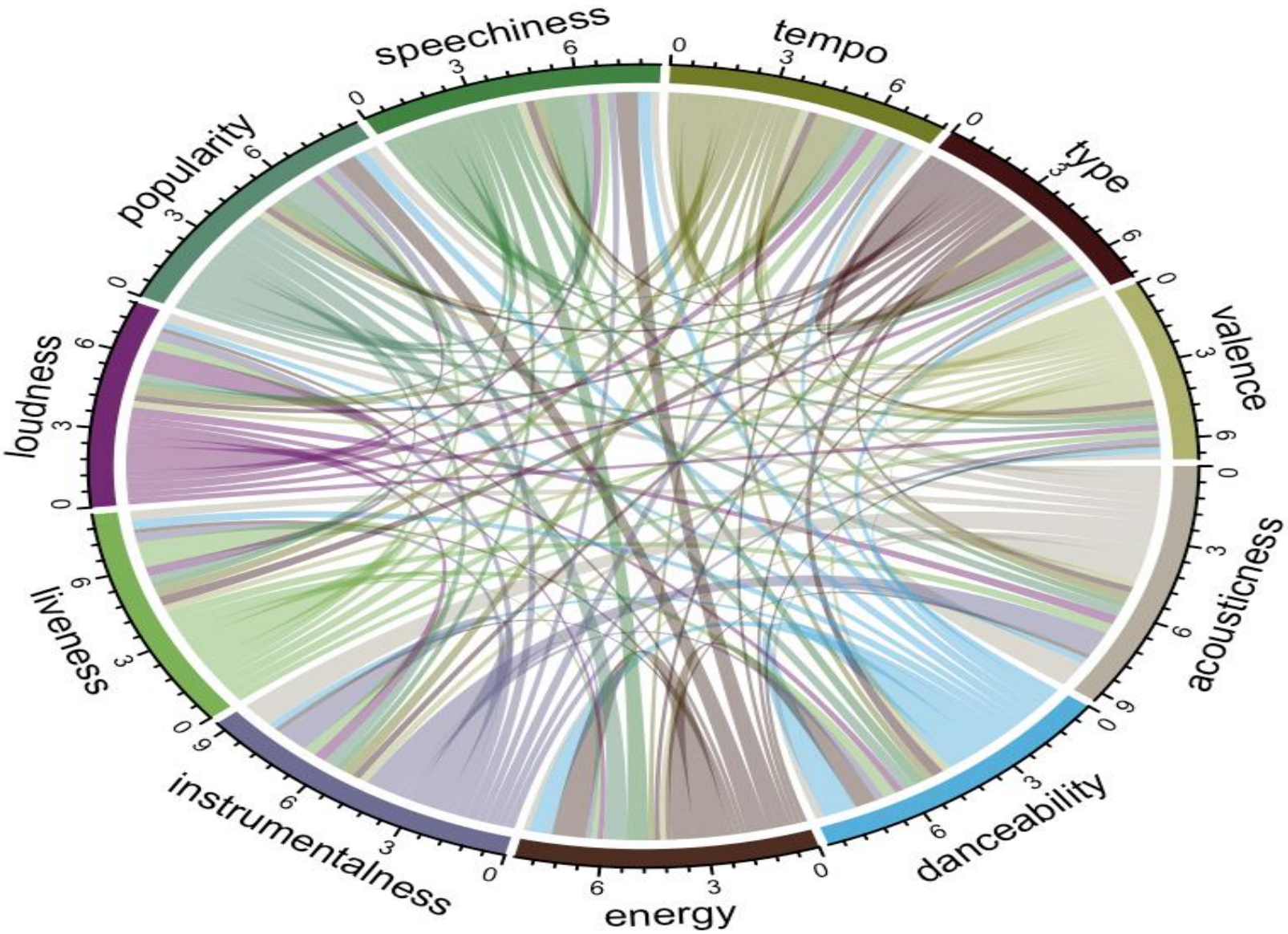
Exploratory Data Analysis (Zoom to view)

Pearson Correlation Coefficient Heatmap – Prepared Using Seaborn Library



Exploratory Data Analysis (Contd.)

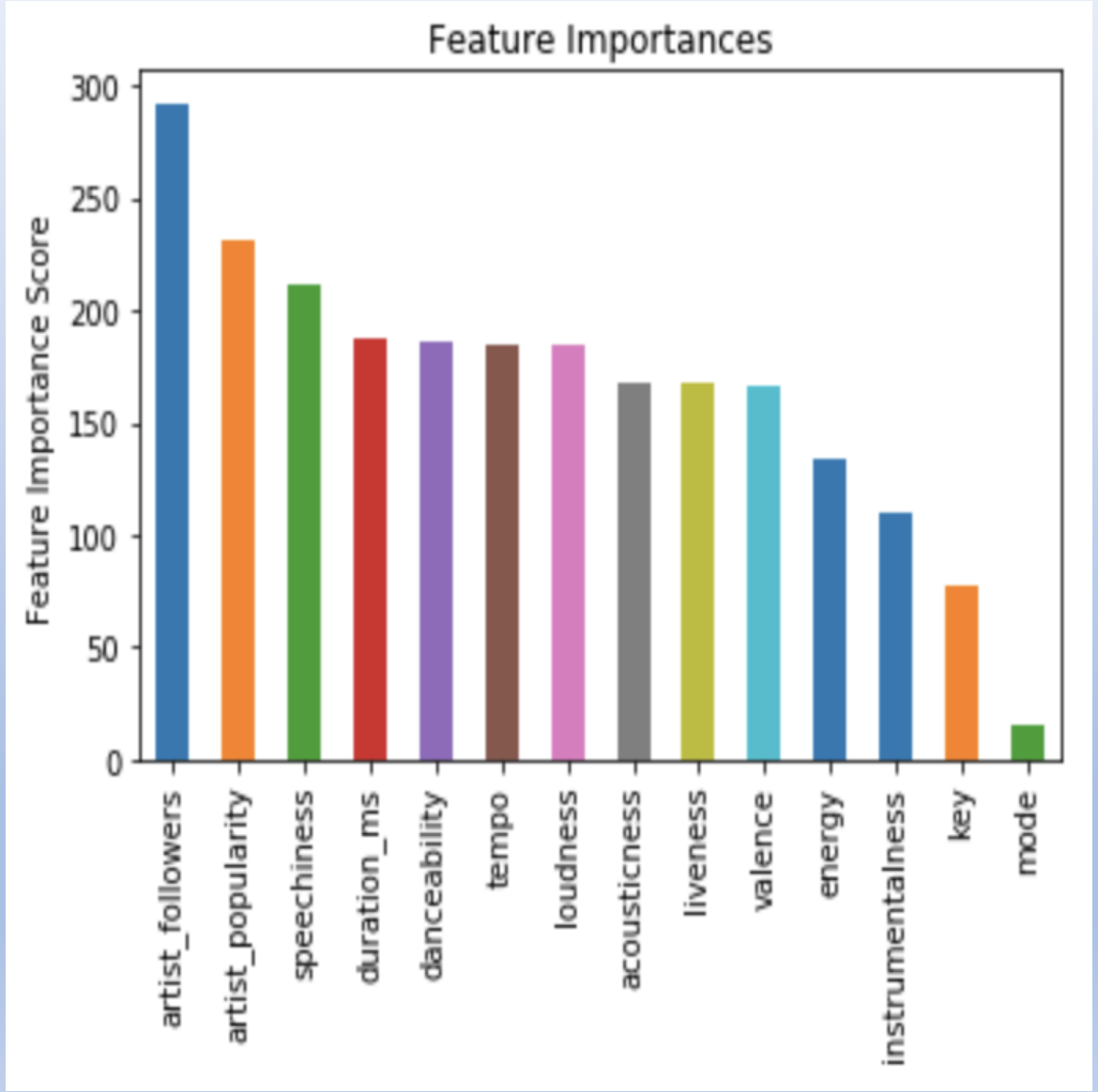
Circular Chord Diagram – Prepared Using R Package Circlize



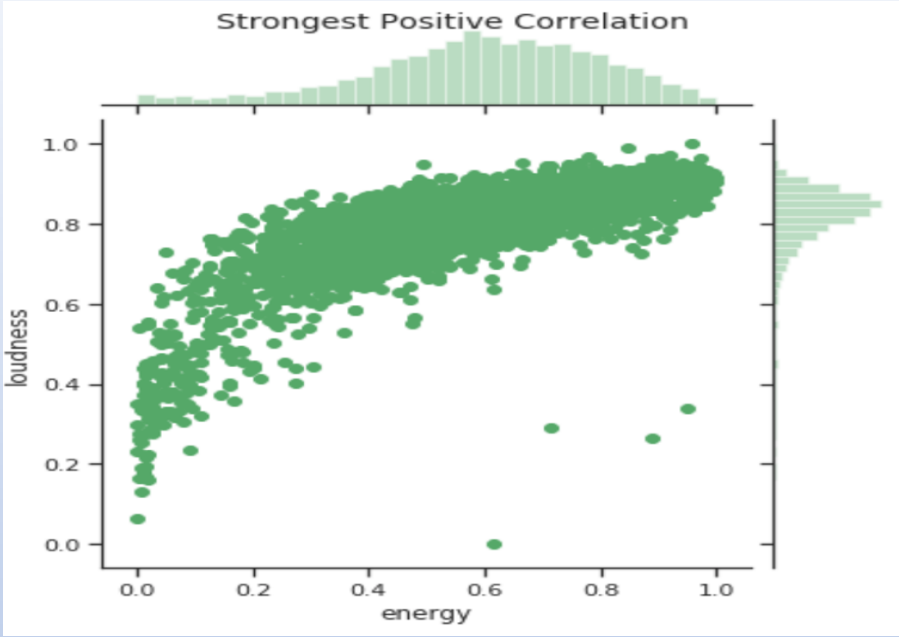
Exploratory Data Analysis (Contd.)

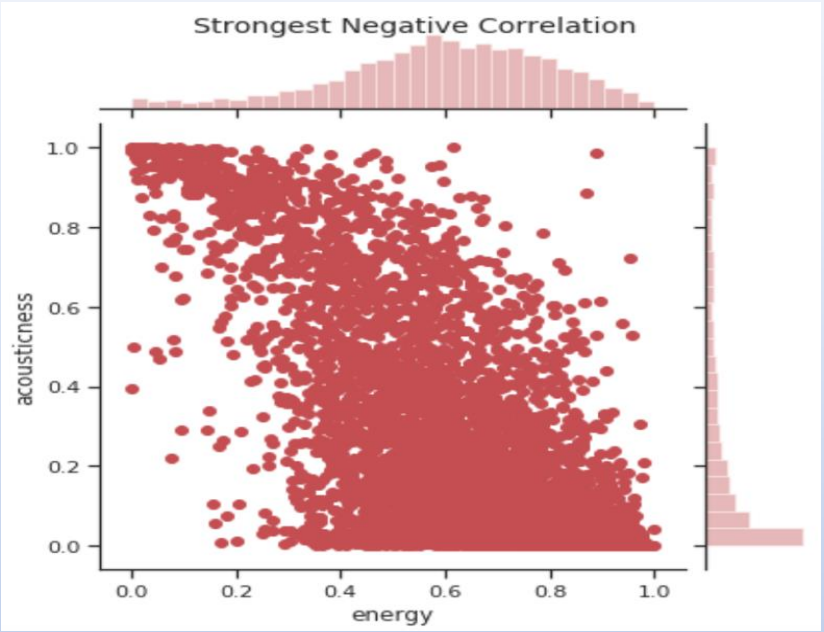
Feature Importance Score & Graph for XG-Boost

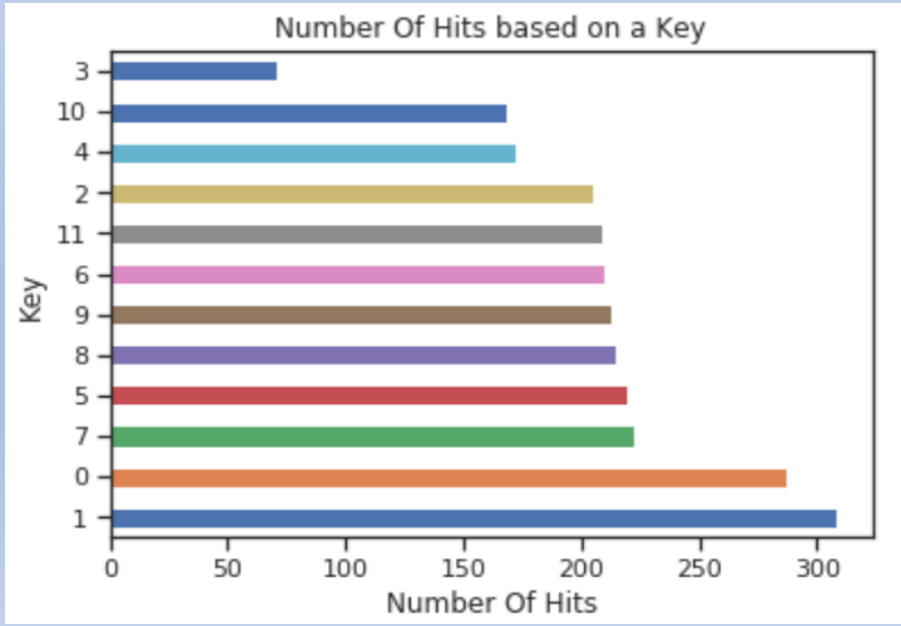
	importance
artist_followers	0.126025
artist_popularity	0.099698
speechiness	0.091498
duration_ms	0.081139
danceability	0.080276
loudness	0.079845
tempo	0.079845
acousticness	0.072508
liveness	0.072076
valence	0.071644
energy	0.057833
instrumentalness	0.047475
key	0.033233
mode	0.006905

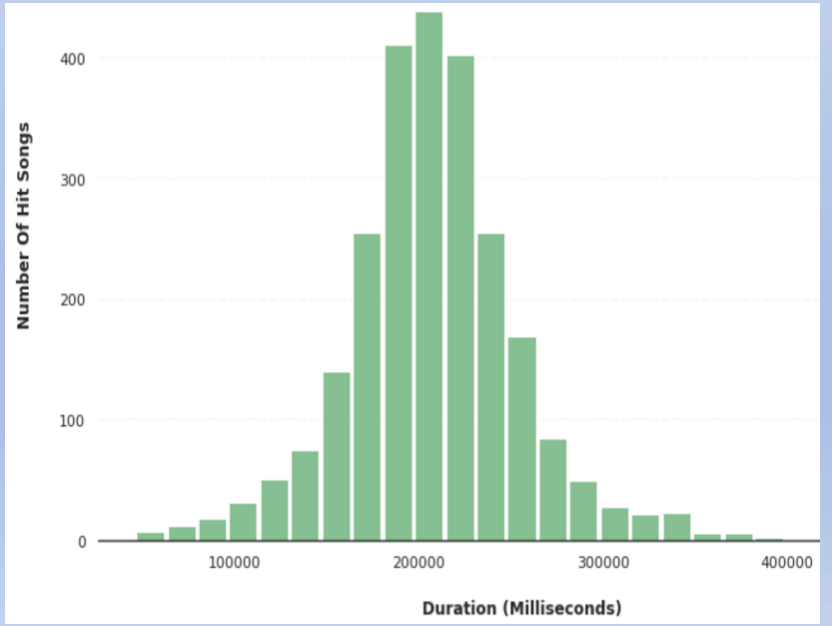


Exploratory Data Analysis (Contd.)









Basic Descriptive Statistics

Attributes	count	mean	std	min	25%	50%	75%	max
acousticness	5000	0.266222	0.284436	0.000003	0.038775	0.153000	0.415000	0.996000
danceability	5000	0.650610	0.161101	0.000000	0.552000	0.667000	0.767000	0.972000
duration_ms	5000	2.038833e+05	5.151627e+04	3.524000e+04	1.771382e+05	2.017070e+05	2.281582e+05	1.355938e+06
energy	5000	0.593527	0.203449	0.000900	0.474000	0.609000	0.741250	0.998000
instrumentalness	5000	0.065873	0.216466	0.000000	0.000000	0.000000	0.000182	1.000000
key	5000	5.275800	3.618612	0.000000	2.000000	5.000000	8.000000	11.00000
liveness	5000	0.171321	0.126541	0.019600	0.098575	0.120000	0.199000	0.979000
loudness	5000	-7.430599	4.247411	-41.53700	-8.463250	-6.443000	-5.010500	0.605000
mode	5000	0.58780	0.49228	0.00000	0.00000	1.00000	1.00000	1.00000
speechiness	5000	0.124946	0.118186	0.000000	0.041400	0.069400	0.175250	0.966000
tempo	5000	120.346522	30.120835	0.000000	95.980000	119.97550	141.98700	220.0990
valence	5000	0.436074	0.219097	0.000000	0.268000	0.422000	0.592000	0.990000
artist_popularity	5000	73.737000	14.919972	0.000000	65.000000	75.000000	85.000000	100.0000
artist_followers	5000	3.206522e+06	5.972459e+06	0.000000e+00	1.389320e+05	7.818130e+05	3.138132e+06	4.368682e+07

Individual Song Prediction Sample Results

1) Song: Never Recover, **Artist:** Lil Baby, Drake
Prediction: **Non-Hit** (Correct prediction even with high artist popularity score of 97)
Prediction Probabilities (Class 0 and 1):

```
 9  ## Prediction using Random Forest Model
10
11  rf.predict_proba(song_1)
```

```
Out[135]: array([[0.53, 0.47]])
```

2) Song Name: Talk, **Artist:** Khalid
Prediction: **Hit** (Correct Prediction)
Prediction Probabilities (Class 0 and 1):

```
 9  ## Prediction using Random Forest Model
10
11  rf.predict_proba(song_1)
```

```
Out[173]: array([[0.42666667, 0.57333333]])
```

Insights & Future Recommendations

- XG-Boost and random forest were the best performing model in terms of mean accuracy measure of around **70%** after 10-fold cross validation. Random forest also performed best in terms of precision of identifying hits.
- Model predicted non-hits marginally better than hits. Model predicted hits better for popular artists i.e. artists whose Spotify popularity score was 70+.
- Decision Tree was one of the worst performers due to severe over fitting.
- Naïve Bayes produced surprisingly good results which established the fact that every attribute had a significant individual weightage in determining outcome.
- PCA analysis revealed 14 attributes contributing to the variance in determining output.
- Artist popularity and number of followers played a major role in making a song hit. Other strong signals are danceability, duration, speechiness, tempo etc.
- Duration of most of the hit songs typically ranged between 3 – 3.5 minutes. Most of hits are in the key of C and C# major scales.
- More feature engineering, hyper parameter tuning, grid search cross validation or randomized grid search for every algorithm will enhance modelling performance further.
- More hits needs to be labelled apart from just billboards which typically lists pop hits. Input dataset needs to be diversified for modelling and predicting yesteryear hits.

Literature Review 

Literature search was carried out to examine and identify similar work. Brief overview is depicted below:

“Show Me What You Got. Song Popularity Prediction Using FMA Dataset”

- Chen, Dixit, Sanyal & Ed, Yip (2018)

This paper did song popularity prediction based on a combination of music metadata features (e.g. genre, title and date_released), audio features (e.g., Mel-frequency cepstral coefficients and spectral contrasts), and musicality features (acoustic-ness, danceability and energy.) Popularity was best predicted for Hip-Hop and Jazz genres.

“Predicting A Song’s Commercial Success Based on Lyrics and Other Metrics”

- Xue, A., & Dupoux, N., (2014)

Song popularity was determined by drawing a subset of songs drawn from the Million Song Dataset and their associated metadata, including genre, year, artist, rhythm/tempo, and instrumentation. The desired output for each of these songs is the value of the corresponding aggregate attribute in the dataset called ”hotttnesss”.

“**AUTOMATIC PREDICTION OF HIT SONGS”**

- Dhanaraj, R., & Logan,B., (2005)

This paper mostly characterized sounds using MFCC (Mel-frequency cepstral coefficients) features focusing on timbre aspects of the music and lyrics.

Thank You!

Hit Song Science: Music Popularity Prediction

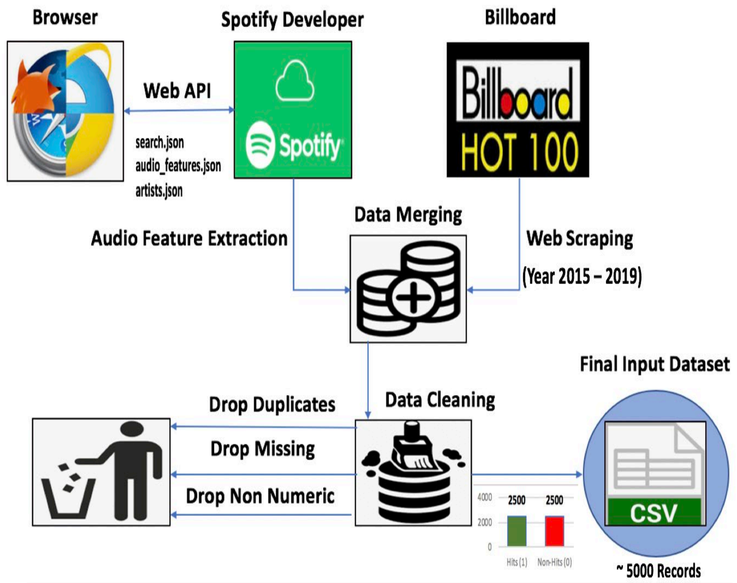
Ranjeeta Bhattacharya

Master of Business and Science RUTGERS

Background

- **Billboard Hot 100** is the music industry standard record chart in the United States and a popularity indicator.
- **Spotify** audio streaming platform exposes set of web API's displaying range of audio features.
- Suitable machine learning techniques can be employed on Spotify audio data to **predict future Billboard hits**.

Data Extraction & Preprocessing



Final Set Of Audio Features Used For Modelling		
acousticness	energy	loudness
danceability	instrumentalness	speechiness
duration_ms	liveness	tempo
valence	artist_popularity and artist_followers	mode
key	bbhot (Class Variable for classification)	popularity (Dependent Variable for regression)

Model Selection

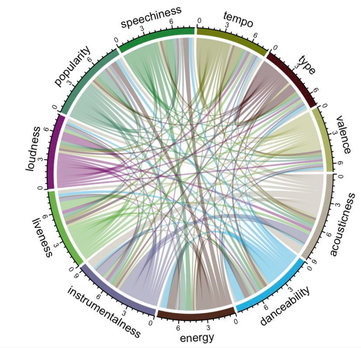
- Around **~5000** records were extracted from Spotify and Billboards Hot-100 chart for years from **2015 ~ 2019**. Separate datasets were used for training and testing.
- Binary class variable **"bbhot"** (values 0 or 1) is used to tag hits and non-hits.
- Range of **supervised** machine learning algorithms were used to predict hits and non-hits.

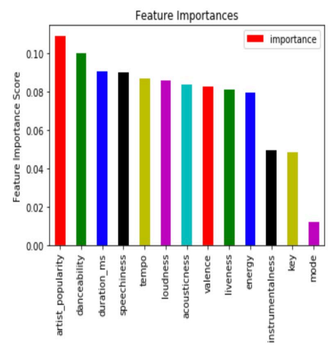
Model Evaluation

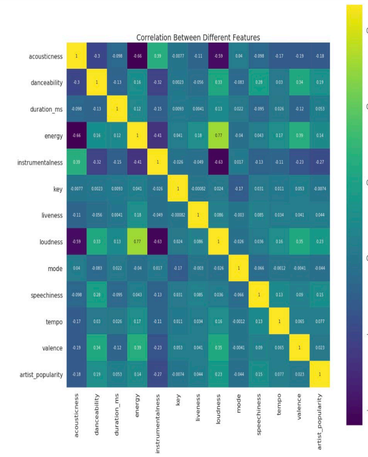
K-NN		SVM		Logistic Regression		Naïve Bayes	
Mode	Accuracy	Mode	Accuracy	Mode	Accuracy	Mode	Accuracy
70-30 train-test split	~ 59%	70-30 train-test split	~ 60%	70-30 train-test split	~ 60 %	70-30 train-test split	~ 55%
With 10-Fold CV	~60% (Mean Score)	With 10-Fold CV	~ 65% (Mean Score)	With 10-Fold CV	~ 65% (Mean Score)	With 10-Fold CV	~ 63% (Mean Score)

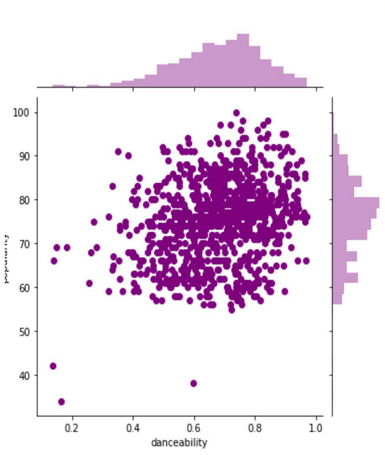
Neural Network		Decision Trees		Random Forest		XG Boost	
Mode	Accuracy	Mode	Accuracy	Mode	Accuracy	Mode	Accuracy
70-30 train-test split	~ 62%	70-30 train-test split	~ 55%	70-30 train-test split	~ 65%	70-30 train-test split	~ 64%
With 10-Fold CV	~ 65% (Mean Score)	With 10-Fold CV	~ 57% (Mean Score)	With 10-Fold CV	~ 69% (Mean Score)	With 10-Fold CV	~ 70% (Mean Score)

Visualization & Insights









- *'acousticness', 'danceability', 'duration_ms', 'liveness', 'loudness', 'speechiness', 'tempo', 'valence', 'artist_popularity', 'artist_followers'* – Best feature columns for prediction.
- Loud, energetic and fast tempo songs feature more in the hit list in recent years.
- Tagging more hit songs apart from Billboards which typically lists pop hits needed to extend analysis further.

References

[1] Billboard. (2019). Billboard Hot 100 Chart.
[2] Spotify Developer API https://developer.spotify/com.

Contact: ranjeeta.bhattacharya@rutgers.edu