Classifying Songs in Spotify Playlists

Capstone Project for IBM Data Science Professional Certificate Specialization

By Tauno Tanilas - 2019

Table of Contents

1. Introduction

This is the Capstone Project for IBM Data Science Professional Certificate course. The subject of final project in this course is left to the author's choice. I decided to apply my acquired knowledge into the field of Data Classification and implement it on the data available in Spotify platform.

Business problems

- Understand the user's needs and preferences in evaluating musical tracks.
- Knowing these characteristics, provide better services.

Target audience

- Audio streaming providers.
- Any person willing to get insights about using classification modeling in solving machine learning problems.

Main goals

- Determine characteristics that define the user's musical taste.
- Using these characteristics compare the music user likes or dislikes.
- Create a predictive model on whether user likes or dislikes a song.
- Determine the main musical genres that are in user's preferences.
- Using Foursquare API, construct a map of accommodation options in one of the selected concert places.

2. Data

The data set is constructed from the Spotify playlists that I have created for myself. There are 80 playlists in total - 45 liked and 25 disliked for training and 10 for evaluation. Liked playlists are labeled as 'GOOD' and disliked as 'BAD'. For later evaluation purposes a separate file is for ten different musical genres that contain tracks that were not used in training stage.

To get the info of playlists tracks and the audio features of them, I used the Spotipy API. The data used for modeling is described by the following ten audio features:

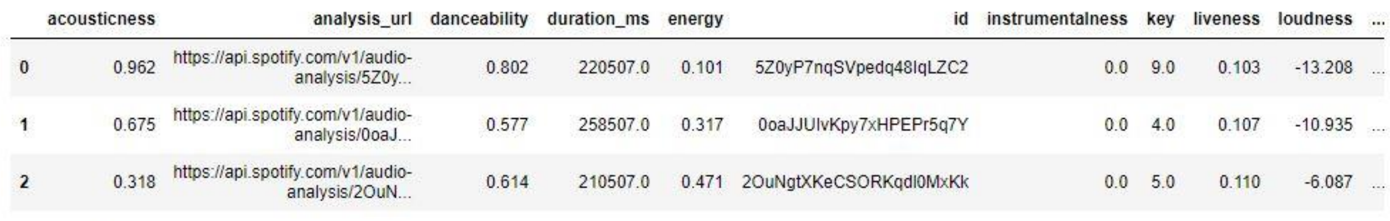
- **Acousticness:** A measure from 0.0 to 1.0 of whether the track is acoustic.
- **Energy:** A measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

- **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.
- **Instrumentalness:** Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
- **Liveness:** Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
- **Loudness:** The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track. Values typical range between -60 and 0 db.
- **Speechiness:** Detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talkshow, audio book, poetry), the closer to 1.0 the attribute value.
- **Tempo:** 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.
- **Valence:** 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).
- **Duration:** The duration of the track in milliseconds.

2.1. Data Acquiring

Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform.

After implementing functions to get data of all my Spotify playlists, their tracks and audio features, the following dataframe was created. Let's display the first three rows of it:



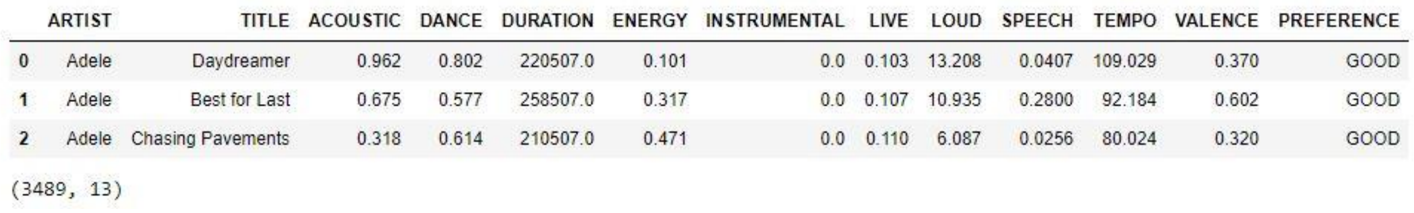
	acousticness	analysis_url	danceability	duration_ms	energy	id	instrumentalness	key	liveness	loudness	...
0	0.962	https://api.spotify.com/v1/audio-analysis/5Z0y...	0.802	220507.0	0.101	5Z0yP7nqSVpedq48IqLZC2	0.0	9.0	0.103	-13.208	...
1	0.675	https://api.spotify.com/v1/audio-analysis/0oaJ...	0.577	258507.0	0.317	0oaJJUIvKpy7xHPEPr5q7Y	0.0	4.0	0.107	-10.935	...
2	0.318	https://api.spotify.com/v1/audio-analysis/2OuN...	0.614	210507.0	0.471	2OuNgtXKeCSORKqdI0MxKk	0.0	5.0	0.110	-6.087	...

3 rows × 21 columns

2.2. Data Cleaning and Preparation

Before more detailed analysis the data needs to be cleaned and transformed to suitable format. The process of data cleaning typically involves tasks like removing duplicate rows, checking missing values, converting data values to another format etc.

After writing and executing data cleaning code, the following dataframe was created. Let's display the first three rows of it:



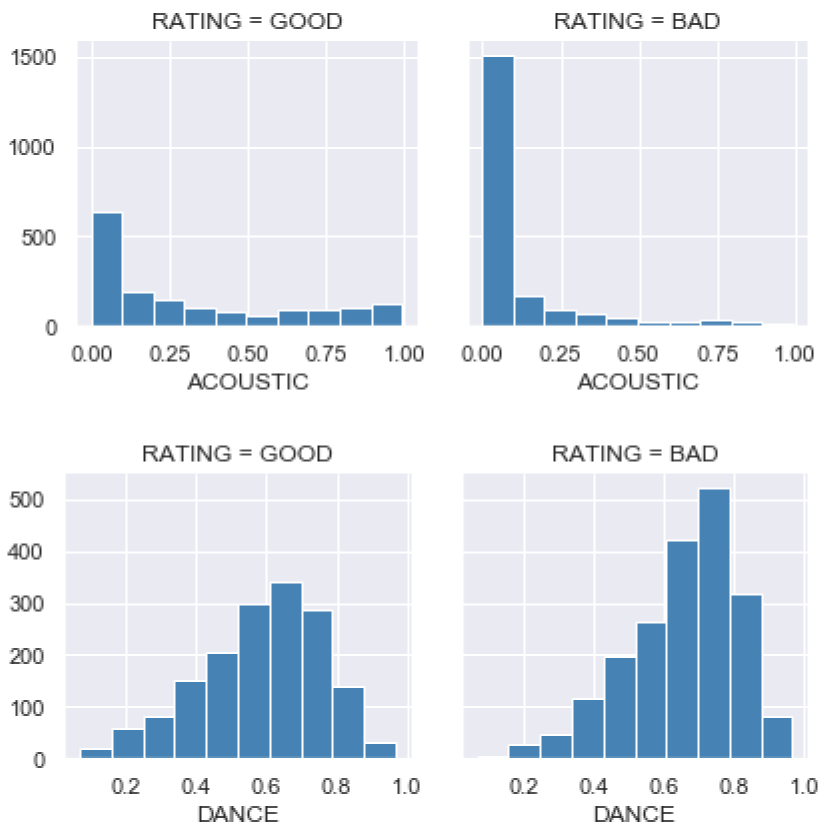
	ARTIST	TITLE	ACOUSTIC	DANCE	DURATION	ENERGY	INSTRUMENTAL	LIVE	LOUD	SPEECH	TEMPO	VALENCE	PREFERENCE
0	Adele	Daydreamer	0.962	0.802	220507.0	0.101	0.0	0.103	13.208	0.0407	109.029	0.370	GOOD
1	Adele	Best for Last	0.675	0.577	258507.0	0.317	0.0	0.107	10.935	0.2800	92.184	0.602	GOOD
2	Adele	Chasing Pavements	0.318	0.614	210507.0	0.471	0.0	0.110	6.087	0.0256	80.024	0.320	GOOD

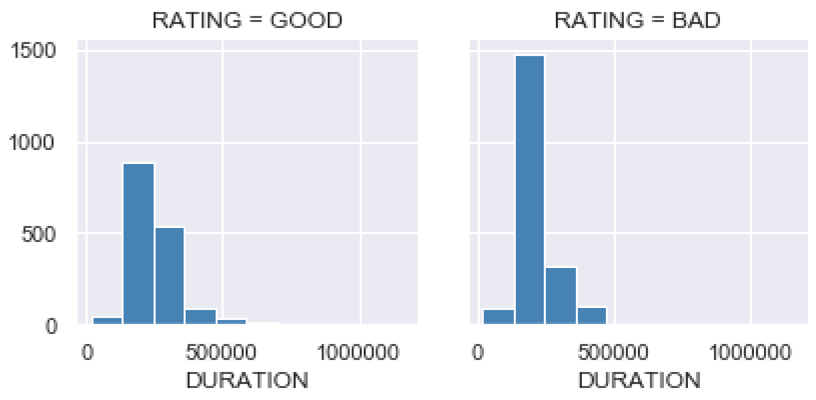
```
(3489, 13)
```

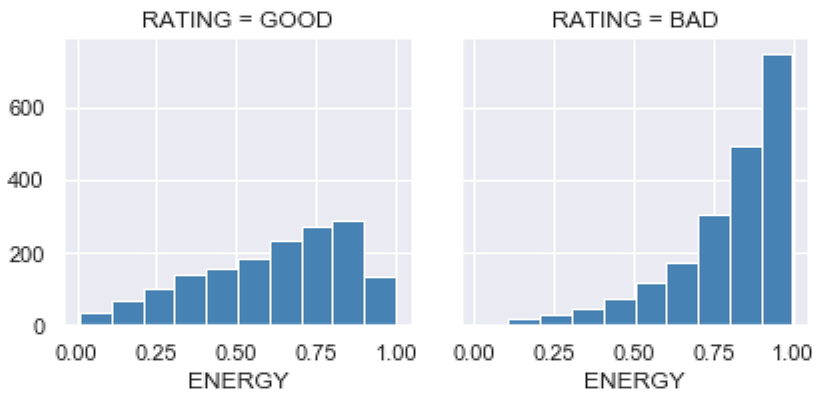
3. Methodology

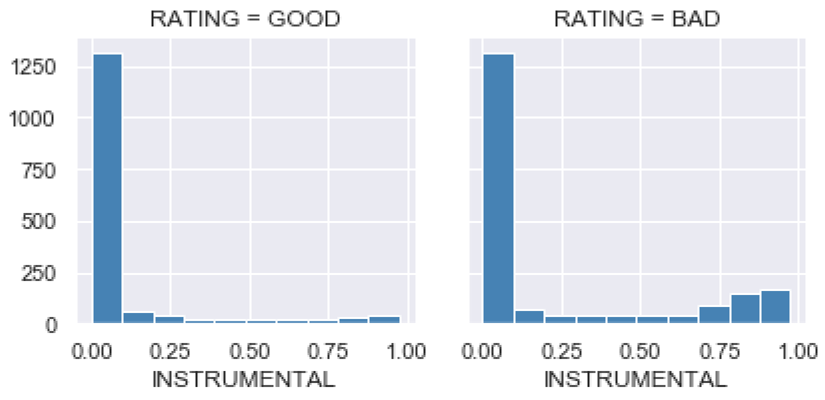
3.1. Exploratory Data Analysis

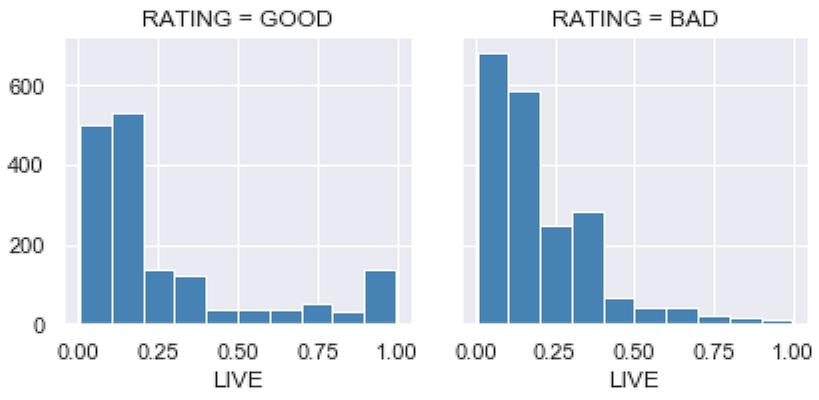
Exploratory Data Analysis is a good way to get more insight how data features are related to each other and which one could have more influence on the modeling result. I have chosen the part of graphical representation in EDA-s analysis techniques to compare how liked and disliked songs differ depending on user's preferences.

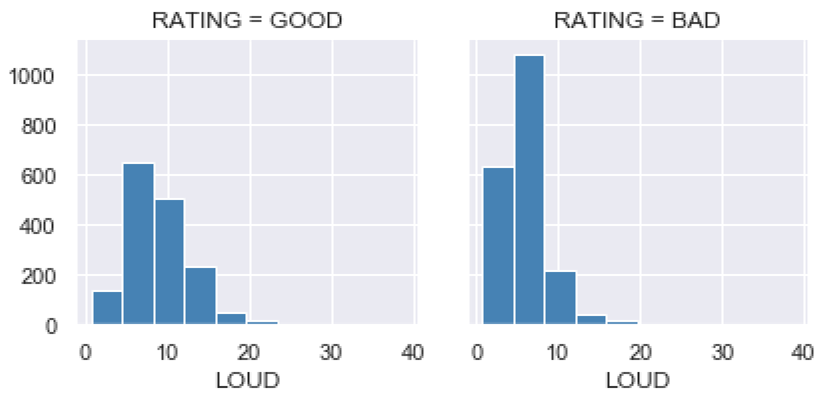


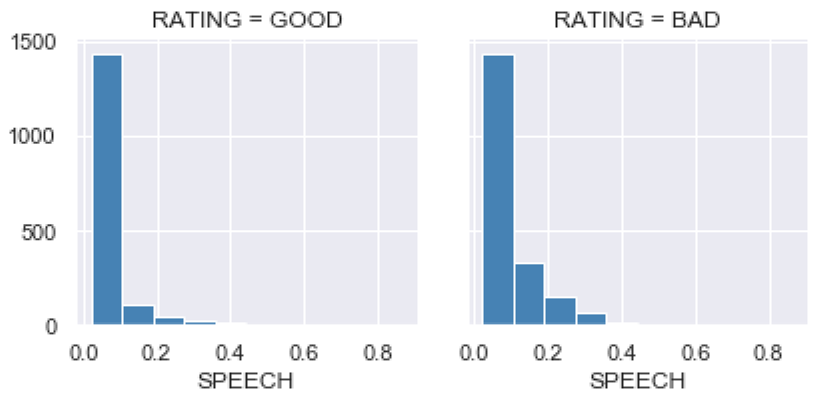

RATING = GOOD
RATING = BAD
1500
1000
500
0
0
500000
1000000
DURATION

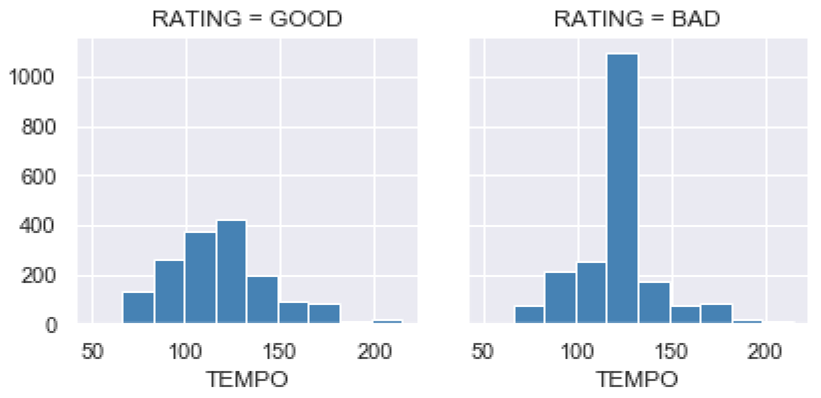

RATING = GOOD
RATING = BAD
600
400
200
0
0.00
0.25
0.50
0.75
1.00
ENERGY

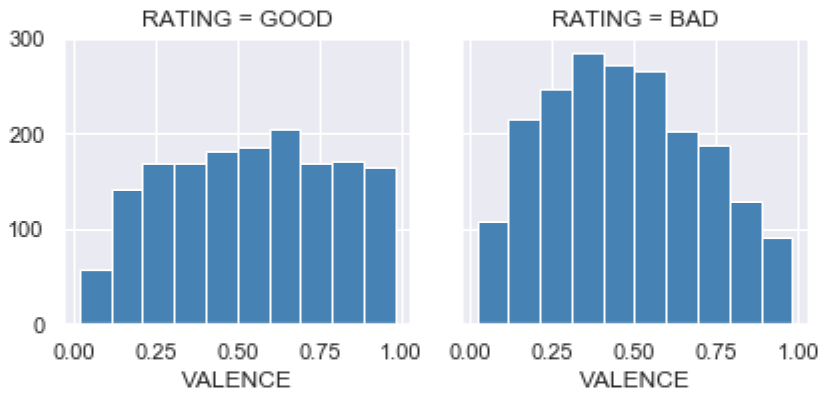

RATING = GOOD
RATING = BAD
1250
1000
750
500
250
0
0.00
0.25
0.50
0.75
1.00
INSTRUMENTAL


RATING = GOOD
RATING = BAD
600
400
200
0
0.00
0.25
0.50
0.75
1.00
LIVE


RATING = GOOD
RATING = BAD
LOUD
LOUD


RATING = GOOD
RATING = BAD
SPEECH
SPEECH


RATING = GOOD
RATING = BAD
TEMPO
TEMPO


RATING = GOOD
RATING = BAD
VALENCE
VALENCE

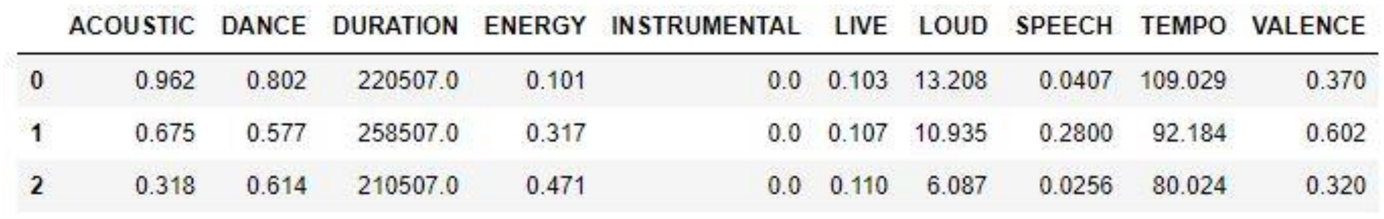
Looking at the distributions of each feature, there are clear distinctions between my liked and disliked songs, especially in the ENERGY, DANCE, LOUD, and ACOUSTIC features.

- **Acoustic:** I don't like songs that are not acoustic at all.
- **Dance:** I prefer songs that are moderately danceable.
- **Energy, Loud, Tempo:** I prefer songs that are less energetic, fast and loud.
- **Valence:** I don't like songs that have little valence but the distribution of values in valence rate is quite equal.

3.2. Feature Selection

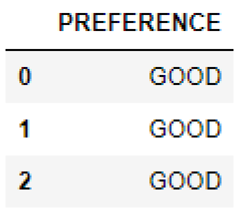
Feature selection is the process of selecting a subset of relevant features for use in model construction. As it turned out in previous chapter some audio features differed in greater extent over preferences that others. For this reason, only these audio features should be selected for later modeling. But as it turned out in later testing it didn't have any remarkable impact for prediction accuracy. Therefore, all audio features have been included in later modeling.

After writing and executing the feature selection code, the following dataframe was created. Let's display the first three rows of **independent** data features:



	ACOUSTIC	DANCE	DURATION	ENERGY	INSTRUMENTAL	LIVE	LOUD	SPEECH	TEMPO	VALENCE
0	0.962	0.802	220507.0	0.101	0.0	0.103	13.208	0.0407	109.029	0.370
1	0.675	0.577	258507.0	0.317	0.0	0.107	10.935	0.2800	92.184	0.602
2	0.318	0.614	210507.0	0.471	0.0	0.110	6.087	0.0256	80.024	0.320

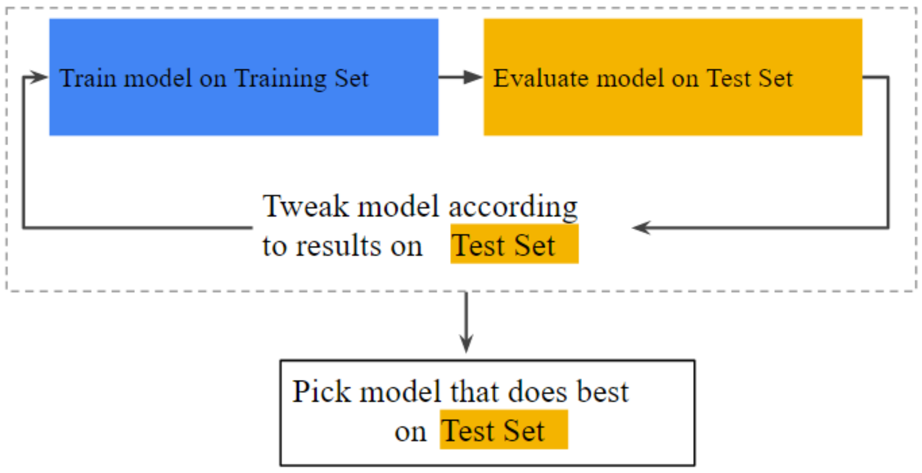
And the same for **dependent** data features:



	PREFERENCE
0	GOOD
1	GOOD
2	GOOD

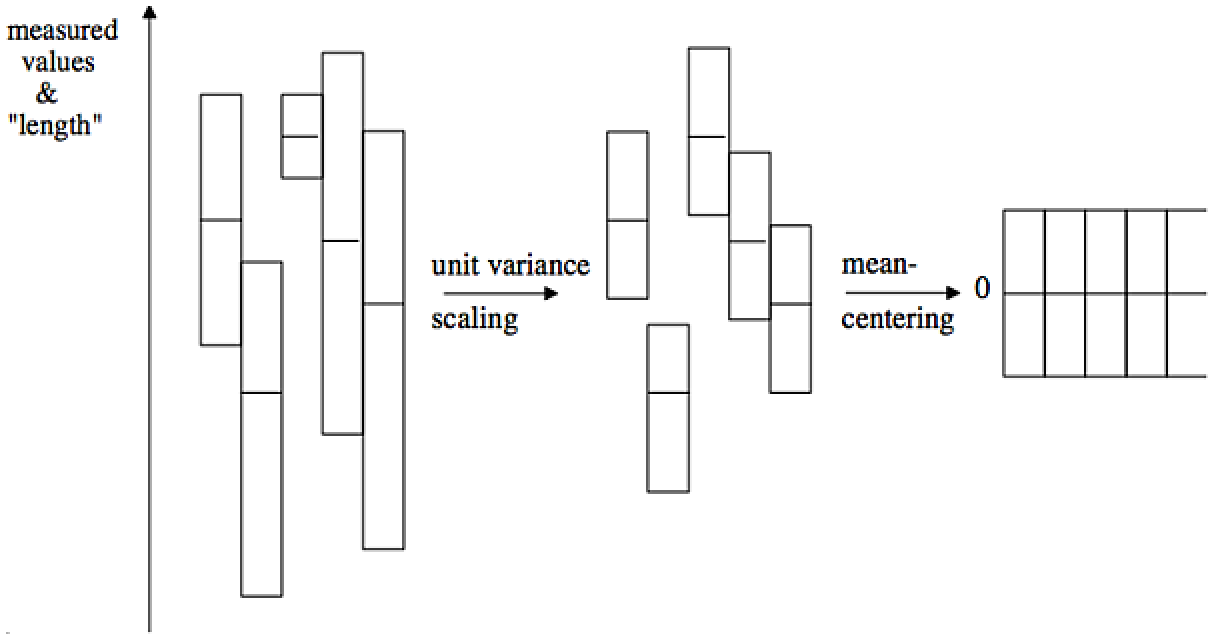
3.3. Data Splitting and Normalization

Splitting the data into a training and test set helps to judge whether a given model will generalize well to new data.


Train model on Training Set
Evaluate model on Test Set
Tweak model according to results on Test Set
Pick model that does best on Test Set

After splitting the data set in proportions where testing data has **20%** of all the data, the train set got **2896** rows and testing set **724** rows.

Data Normalization is a process of transforming values of several variables **into a similar range**. It is needed for mathematical-based algorithms to interpret variables with different magnitudes and distributions equally. Normalization gives the data **zero mean** and **unit variance** and is calculated by subtracting from each datapoint their mean value and then dividing by the variance.



To avoid **Data Leakage**, it is important to remember that normalization should be carried out after splitting the data into training and testing sets. Data leakage means that information from outside the training data is used to create the model. Testing data should be unseen and accessible at the training stage. If the mean and variance is calculated from the whole data set, then the future information is introduced to the training variables.

In normalization, the sklearn.preprocessing.StandardScaler package was used. The formula by which the standard score of a sample x is calculated is the following:

$z = \frac{\mathrm{x-u}}{s}$, where

u - the mean of the samples
s - the standard deviation of the samples

After normalizing data, the following training data dataframe was created. Let's display the first three rows of it:

	ACOUSTIC	DANCE	DURATION	ENERGY	INSTRUMENTAL	LIVE	LOUD	SPEECH	TEMPO	VALENCE
0	-0.705768	-1.983439	1.104880	0.949050	-0.393894	2.925918	-0.791645	-0.506682	-1.467003	-0.027716
1	0.281826	0.813724	-0.304712	-0.285565	-0.559277	-0.789725	0.790116	-0.661744	-0.104507	1.904277
2	2.207462	0.049768	-1.011505	-1.965888	-0.562119	0.425301	1.115622	-0.393671	1.384146	0.873340

And the same for the testing data set:

	ACOUSTIC	DANCE	DURATION	ENERGY	INSTRUMENTAL	LIVE	LOUD	SPEECH	TEMPO	VALENCE
0	-0.725145	1.009868	-0.363918	0.651478	0.506258	-0.390273	-0.398017	-0.207204	0.160924	1.896967
1	-0.614260	-0.172687	-1.228097	0.743661	-0.519411	-0.163615	-0.497802	0.013100	-0.038882	1.590061
2	-0.675446	0.873419	0.811672	0.862183	-0.520250	2.455540	-0.819532	-0.287550	0.283417	1.144024

3.4. Prediction Metrics

There are many prediction metrics that can be used in measuring machine learning model accuracy. In this report the **Accuracy, Precision** and **Recall** as three well known metrics were used.

Define the positive event that is the basis of predictions: **"Is the track listened labeled as 'GOOD'?"**

Define four types of predictions that are the basis of formulas:

- **True Positives (TP)** — model predicts the event is positive and that was correct.
- **False Positives (FP)** — model predicts the event is positive but that was incorrect.
- **True Negatives (TN)** — model predicts the event is negative and that was correct.
- **False Negatives (FN)** — model predicts the event is negative but that was incorrect.

- **Accuracy** - the proportion of total number of predictions that were correctly identified

$$\text{accuracy} = \frac{\text{correct}}{\text{correct} + \text{incorrect}} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{TN}) + (\text{FP} + \text{FN})}$$

- **Precision** - the proportion of positive cases that were correctly identified

$$\text{precision} = \frac{\text{correctly predicted positive events}}{\text{all positive event predictions}} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall/Sensitivity** - the proportion of actual positive cases that were correctly identified

$$\text{recall} = \frac{\text{correctly predicted positive events}}{\text{total number of positive events}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

3.5. Model Selection

To conduct a modeling, I have chosen three well known classification algorithms - **Logistic Regression (LR), K-Nearest Neighbors (KNN) and Support Vector Machines (SVM)**. After building a model and making predictions, the following accuracy metrics were achieved:

	ACCURACY	PRECISION	RECALL
Train set (LR)	0.811809	0.812130	0.805372
Test set (LR)	0.809392	0.809279	0.804241

	ACCURACY	PRECISION	RECALL
Train set (KNN)	0.854282	0.862081	0.845072
Test set (KNN)	0.827348	0.832282	0.819391

	ACCURACY	PRECISION	RECALL
Train set (SVM)	0.866367	0.868480	0.860691
Test set (SVM)	0.835635	0.834543	0.832616

3.6. Model Optimization

In Model Selection chapter model building was mostly done with default parameters. Traditional way in improving model’s performance is **Hyperparameter Optimization** that is done via **Grid Search** cross-validation. Grid Search works by searching exhaustively through a manually specified subset of hyperparameters. A **hyperparameter** is a parameter whose value is set in algorithm initialization before the learning process begins.

In previous work the best prediction accuracy was received by SVM algorithm. After optimizing data, the following results were achieved:

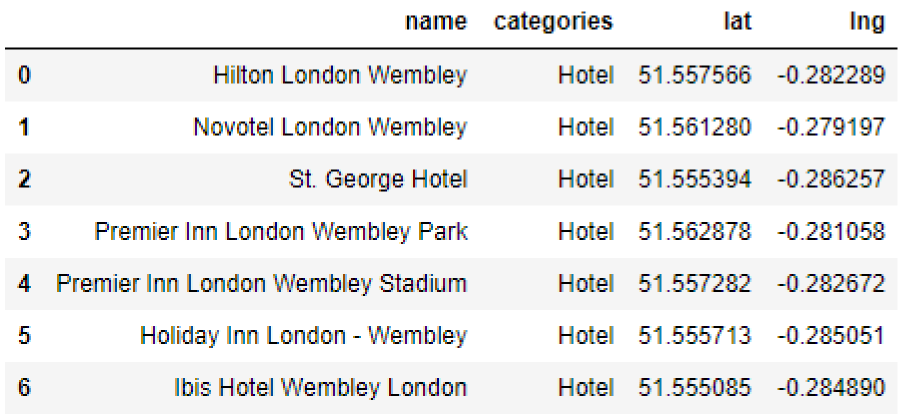
- **Best parameters:** {'C': 1, 'class_weight': None, 'gamma': 'scale', 'kernel': 'rbf', 'shrinking': True}
- **Best score:** 0.848083
- **Prediction accuracy after GridSearch:** 0.827348

As seen from the result the default settings seem to be good enough and hyperparameter tuning didn't make it better.

3.7. Clustering Neighborhoods

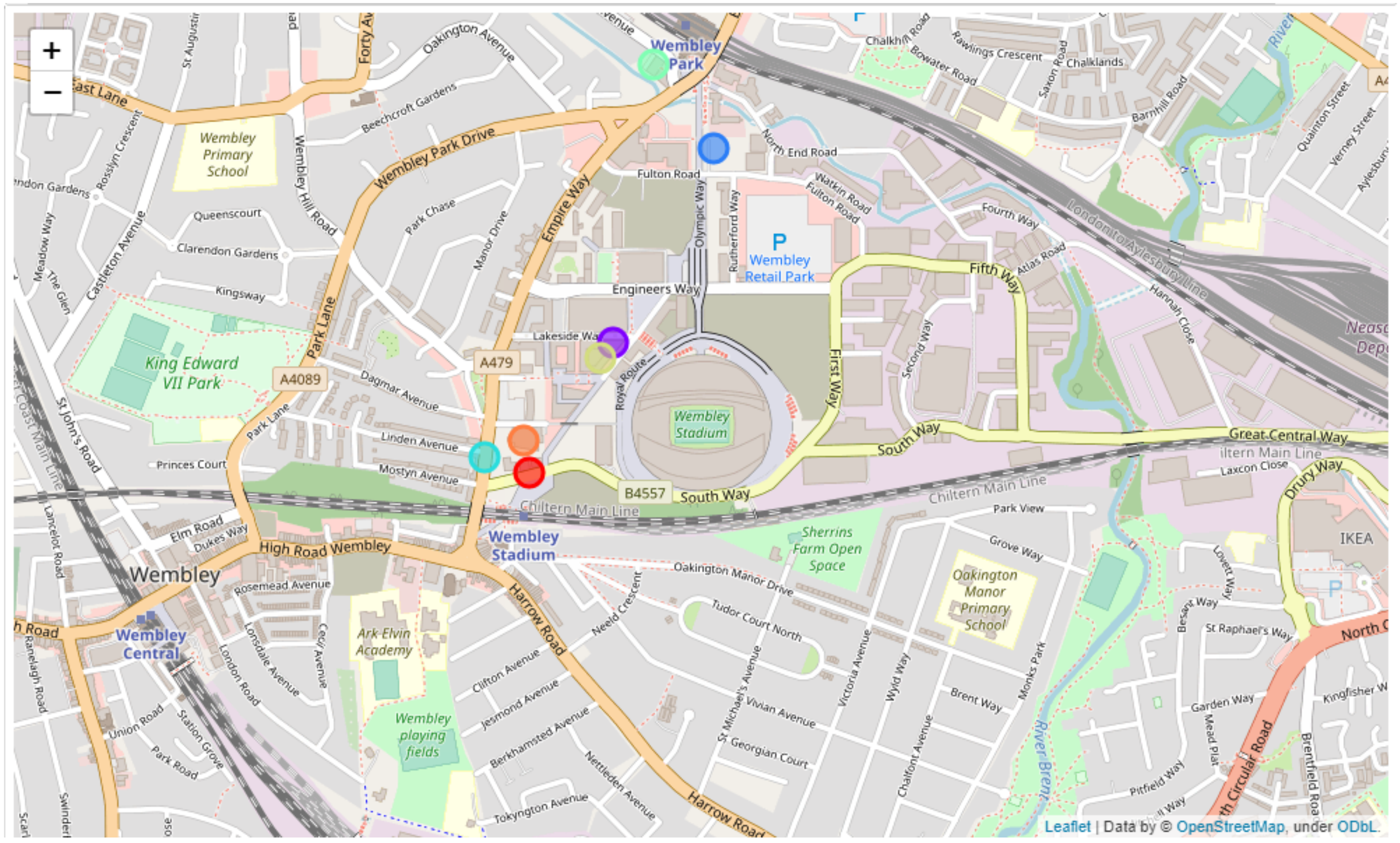
One condition in performing this certificate specialization was the usage of Foursquare API. Therefore, let's suppose we want to attend the Ed Sheeran concert at Wembley Stadium in London. In this case any suggestions about the nearby hotels would be welcome. Let's first find out the geographical coordinates of Wembley Stadium and then plot the colored circles for each hotel found in a request.

After making Foursquare request and filtering the result, the following list of hotels were received:



	name	categories	lat	lng
0	Hilton London Wembley	Hotel	51.557566	-0.282289
1	Novotel London Wembley	Hotel	51.561280	-0.279197
2	St. George Hotel	Hotel	51.555394	-0.286257
3	Premier Inn London Wembley Park	Hotel	51.562878	-0.281058
4	Premier Inn London Wembley Stadium	Hotel	51.557282	-0.282672
5	Holiday Inn London - Wembley	Hotel	51.555713	-0.285051
6	Ibis Hotel Wembley London	Hotel	51.555085	-0.284890

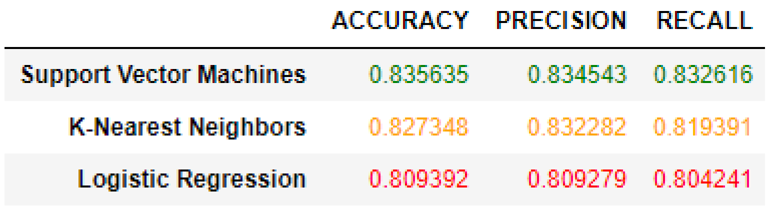
Using Folium visualization package, the following map was created with seven hotels located nearby Wembley Stadium:



4. Results

4.1. Algorithms Comparison

After summarizing prediction results of three different algorithms, the following dataframe can be constructed. The best classifier is **Support Vector Machines**, but there is very little difference with K-Nearest Neighbors.

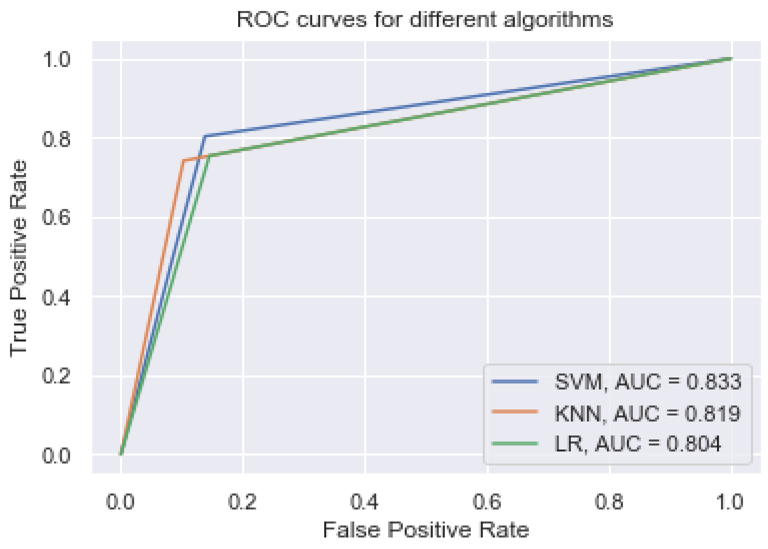


	ACCURACY	PRECISION	RECALL
Support Vector Machines	0.835635	0.834543	0.832616
K-Nearest Neighbors	0.827348	0.832282	0.819391
Logistic Regression	0.809392	0.809279	0.804241

4.2. ROC AUC Curve

In order to compare one model to another, the AUC ROC curve needs to be constructed. **AUC** (Area Under the Curve) **ROC** (Receiver Operating Characteristics) **curve** is a performance measure at various thresholds settings. ROC is a probability curve and AUC degree of separability. ROC curves are obtained by plotting the

True Positive rate against the **False Positive** rate and the performance is usually measured by the area under the ROC curve. Thus, it shows the model's capability to distinguish the difference between classes.

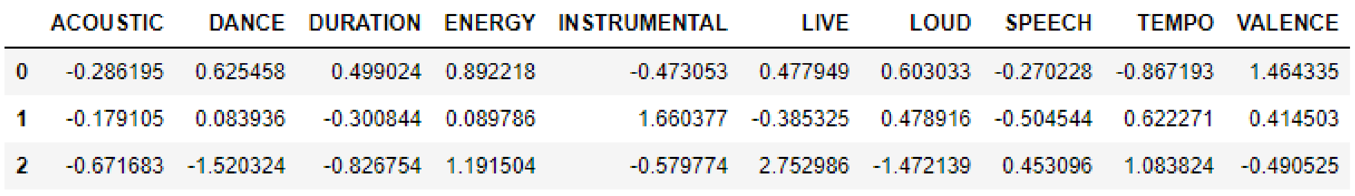


In this case the **best AUC by SVM was 83.3%**, indicating a good level of prediction accuracy to distinguish a good musical track from bad one.

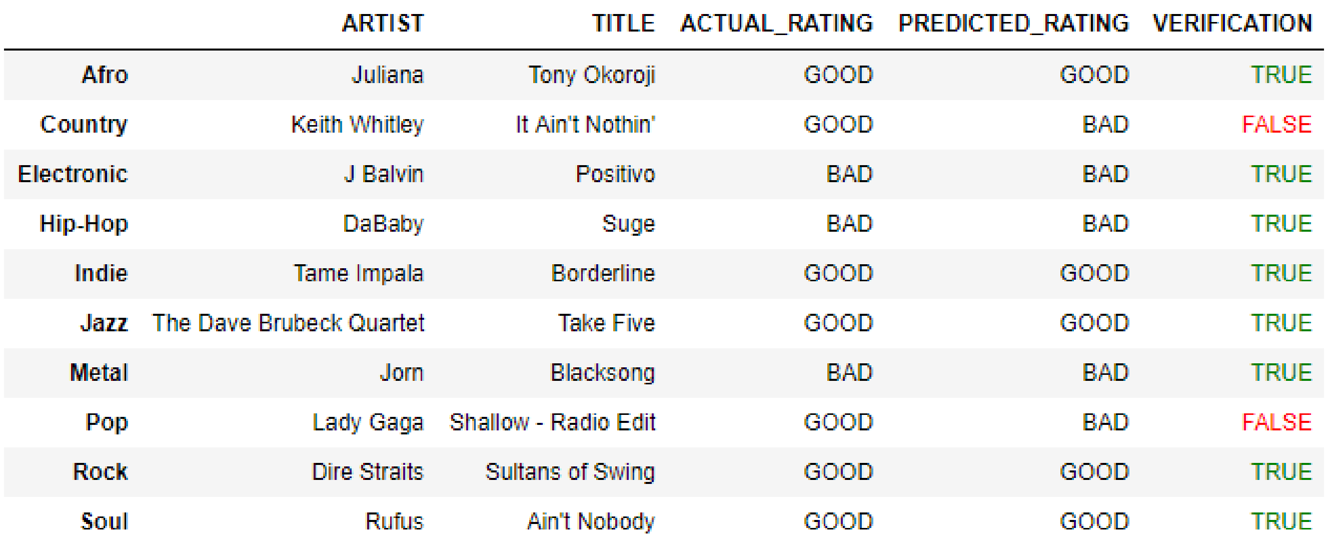
4.3. Model Evaluation

Partitioning data into the training and test sets may be insufficient when doing many rounds of testing and hyperparameter tuning. Doing many iterations where model is trained, tested, metrics observed and parameters adjusted until the best possible model is found, may result to overfit to the peculiarities of test data.

One solution to handle this, is to create a third data set out of previous two partitions. In this way created classifier is evaluated only on this data set. I have chosen a way where data for the final test contains ten musical genres from playlists not used in training. One track from each of them is included for evaluation, a dataframe with appropriate audio features constructed and then predictions made. For data labeling, I listened to all ten tracks and rated them as a 'GOOD' or 'BAD' one.



	ACOUSTIC	DANCE	DURATION	ENERGY	INSTRUMENTAL	LIVE	LOUD	SPEECH	TEMPO	VALENCE
0	-0.286195	0.625458	0.499024	0.892218	-0.473053	0.477949	0.603033	-0.270228	-0.867193	1.464335
1	-0.179105	0.083936	-0.300844	0.089786	1.660377	-0.385325	0.478916	-0.504544	0.622271	0.414503
2	-0.671683	-1.520324	-0.826754	1.191504	-0.579774	2.752986	-1.472139	0.453096	1.083824	-0.490525



	ARTIST	TITLE	ACTUAL_RATING	PREDICTED_RATING	VERIFICATION
Afro	Juliana	Tony Okoroji	GOOD	GOOD	TRUE
Country	Keith Whitley	It Ain't Nothin'	GOOD	BAD	FALSE
Electronic	J Balvin	Positivo	BAD	BAD	TRUE
Hip-Hop	DaBaby	Suge	BAD	BAD	TRUE
Indie	Tame Impala	Borderline	GOOD	GOOD	TRUE
Jazz	The Dave Brubeck Quartet	Take Five	GOOD	GOOD	TRUE
Metal	Jorn	Blacksong	BAD	BAD	TRUE
Pop	Lady Gaga	Shallow - Radio Edit	GOOD	BAD	FALSE
Rock	Dire Straits	Sultans of Swing	GOOD	GOOD	TRUE
Soul	Rufus	Ain't Nobody	GOOD	GOOD	TRUE

ACCURACY: 0.8, PRECISION: 0.8, RECALL: 0.857143

Since the model fits to the training set slightly better than to the evaluation set, some overfitting may exist in the model. But as the cap is quite small and the accuracy score is still good, **we can consider the project successful**.

5. Discussion

Doing a real data science project was very helpful in practicing topics learned in the course and also finding new tips & tricks for many complicated tasks. Most time-consuming part in this project was definitely data acquiring and cleaning. And before that also finding the project subject that could from one side be interesting to yourself and on the other hand be supported by the necessary data. The fact if the data has enough meaning at all for later modeling will often come out after a lot of work has been already done in data preparation.

Although the model's prediction capability was good the next steps could be considered in further developments if there is a desire to improve the results:

- Increase the data set
- Try different proportions between liked and disliked playlists
- Try different normalization methods
- Try different modeling algorithms

6. Conclusion

The key factors that I can conclude from this project are:

- Most important audio features that determine my musical taste are Acousticness, Danceability, Energy and Valence.
- I prefer Country, Indie, Jazz, Pop, Rock, Soul and Afro music but Electronic, Hip-Hop and Metal are not so much in my favor.
- The best classifier was Support Vector Machines and the model optimization didn’t give any better results.
- Always normalize data before modeling. The difference in prediction accuracy with normalized and not normalized data was depending on the algorithm up to a twenty percentage point.
- Despite the quite small dataset the prediction model I created achieved a good level of prediction accuracy.