

Spotify Track Features

Author One

Author Two

Abstract

ITMAL - Machine Learning Slut-Journal

Contents

1 Problem Description	2
1.1 End-to-End Machine Learning study process	2
1.2 Supervised learning explained	3
2 The Spotify Tracks dataset	5
2.0.1 Identification and general features:	5
2.0.2 Sound and feel:	5
2.1 Visualising the data	5
2.2 Cleaning the data	8
2.3 Feature scaling and PCA	10
3 Genre Classification	12
3.0.1 Algorithm Selection	12
3.0.2 Data processing and structure	12
3.1 K-Nearest Neighbors	12
3.1.1 Hyperparameter Search	12
3.1.2 Results	13
3.1.3 Discussion	14

1 Problem Description

This project is applying the process for developing a Machine Learning (ML) Model described in the chapter “End-to-End Machine Learning Project”¹ with the goal of creating a ML model to predict genres and popularity of tracks on Spotify based on other features defined for each of these songs.

The core idea is to produce 2 pipelines in parallel, working independant of each other. One pipeline should end with a model to be used to predict popularity of future songs from the given features, and the other to try and predict the genre of a given song. As such, this project will work both with regression and classification. It must be stated that in order for this process to work, for either of the prediction of a quantity or category, there must be some correlation/distinction between the features. Thus, the underlying assumption for the project, and decision to use the dataset, is that the features generated by spotify to describe their songs is connected in some way - and our goal is find an algorithm that can use that connection for prediction, if it is there.

Since the data set include the target value of our predictions these features will be seperated from the data set and vil be used as the labels in training the model using supervised learning, both the classification and regression problem fits this training strategy.

The “End-to-End” process and and supervised learning concepts will be further explained in the following two sections.

1.1 End-to-End Machine Learning study process

Genneraly the chapter outlines the following key steps in developing such a model:

- **Getting the Data and looking at the Big Picture:** Defining the problem, we wish to apply a ML model to solve, finding an analyzing the dataset to get an overview and initial insight into any underlying patterns and relationships between features in the data set.
- **Prepare the Data for Machine Learning Algorithms:** From the data analysis necessary data cleaning and feature scaling are planned and performed to prepare it for the selected model. Most ML models perform poorly on accidental null-data and features with a high variance. To this end we can drop bad samples or features and apply normalization or standardization to sanitize the data.
- **Selecting, Training and Fine-Tuning a Model:** When the data has been prepared it time to train a appropiate model and validate the result. This is done utilizing a iterative process where the model is optimize according to a cost function and then its validated using a score metric.
- **Launching, Monitoring and Maintaining the System:** This is eventual goal of the entire process. When the model has been optimized then it time to launch it. Hopefully it will perform well and make accuarate predictions. However it is important to keep monitoring maintaining the model as it generally tends to rot over time as the input data might change and therefore retraining on fresh data might be necessary.

¹Hands-On Machine Learning with Scikit-Learn andTensorFlow, Aurélien Géron, O'Reilly, 2017

1.2 Supervised learning explained

In the case of this project a supervised learning approach is used to train the selected models, see Figure 1 for a detailed map-view of this approach.

The Map

Supervised Classification and Regression

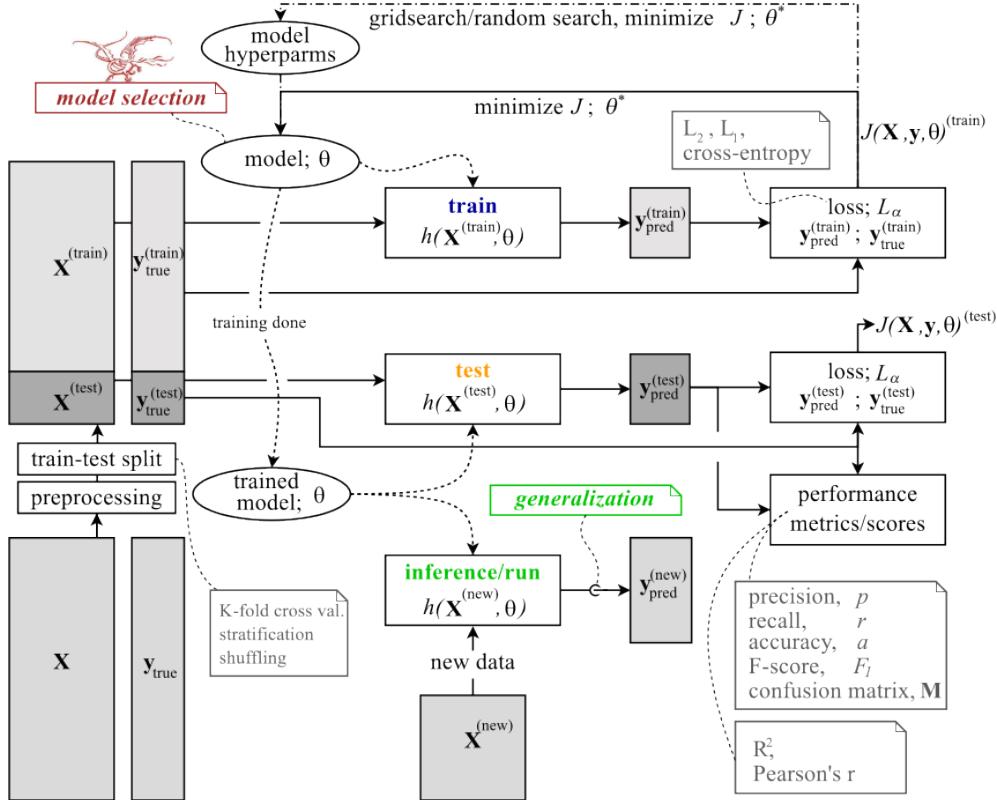


Figure 1: Map-overview of model-fitting using supervised learning

The collection data samples containing all the information on different features denoted \mathbf{X} is one of the components the entire set. The other denoted \mathbf{y}_{true} is a list of true values, one corresponding to each sample in \mathbf{X} . \mathbf{y}_{true} can either be a label or a numerical value depending on whether problem is classification or regression.

When the data have been acquired and preprocessed then its time to split into two sets one for training the model and another for performance validation. The components in these sets are denoted $\mathbf{X}_{\text{train}}$ and $\mathbf{y}_{\text{train}}$ and \mathbf{X}_{test} and \mathbf{y}_{true} for the training and validation (test) set respectively. Splitting the data into two sets are an important step because for the validation to be meaningful the model has to be applied to data that it has not parsed before. This is the only way its ability to generalize and make predictions can be analyzed.

The next step after splitting the data is running the training loop. Here the model (\mathbf{h}) is fed the data and then is tasked to make predictions (\mathbf{y}_{pred}) on each of the training samples, it then evaluates this result compared to the samples \mathbf{y}_{true} using a cost function (\mathbf{J}) often this function is chosen to be the least square solution. This is an iterative process where the goal is to minimize the cost function for the entire set of training samples. In each iteration of the training loop the model parameters (θ) are tweaked. When the cost function reaches its minimum the ideal model parameters are found and the training process of the model is complete.

When the training process of the model has been completed the model is tasked with making predictions on the test set (unseen samples). Again the predictions are compared y_{true} but instead of applying a cost function, a score metric is applied instead as the cost function would not give any meaningful representation of how well the model performs. The goal is to reach as high a score as possible in contrast to the finding a minimum for the cost function.

2 The Spotify Tracks dataset

The dataset for this report is found on kaggle, where the original data has been gathered directly from spotify via their API. The dataset includes 16 features with around ~233 thousand songs. The features included in the dataset can be divided into 2 main categories, namely identification/general features and sound/feel features. The features included in the dataset are as follows:

2.0.1 Identification and general features:

- Genre
- Artist name
- Track name
- Track ID
- Popularity

2.0.2 Sound and feel:

- Acousticness
- Danceability
- Duration in ms
- Energy
- Instrumentalness
- Key
- Liveness
- Loudness
- Mode
- Speechiness
- Tempo
- Time signature
- Valence

It should be noted here that the data has already been preprocessed by spotify before we attempt to do anything with it. This can easily be seen from the popularity feature being a scalar, and thus, must be generated from some algorithm translating the yearly/monthly/daily listens into some entity. With that said, some processing for the models must be performed, be that cleaning or scaling as will be seen from the next parts of the report.

It should be noted that for the classifier there is a fear of the dataset being too small, as with 26 unique genres, 233 thousand samples could in reality be a too small training set for the model to learn.

2.1 Visualising the data

As a first step in the preprocessing of the data, a sufficient knowledge of the dataset should be established. The data should be analyzed in regards to both popularity and genre, since those are the features to be predicted from our models. Starting off with the popularity, a histogram is made using matplotlib to gain insight of it's distribution.

It can be seen that the distribution resembles a normal distribution, but deviates significantly in some places. The most notable difference being the big spike in the first bin(a popularity rating of 0), which could have several explanations. It might be that there are simply just a lot of unpopular songs on the platform, as the data would seem to indicate. Another explanation could be that the popularity score is not calculated by spotify before some criteria is met, e.g. a track has been on the platform for period of time. It depends on how and when the score is computed, when the tracks were put on the platform, all information that the group does not currently have.

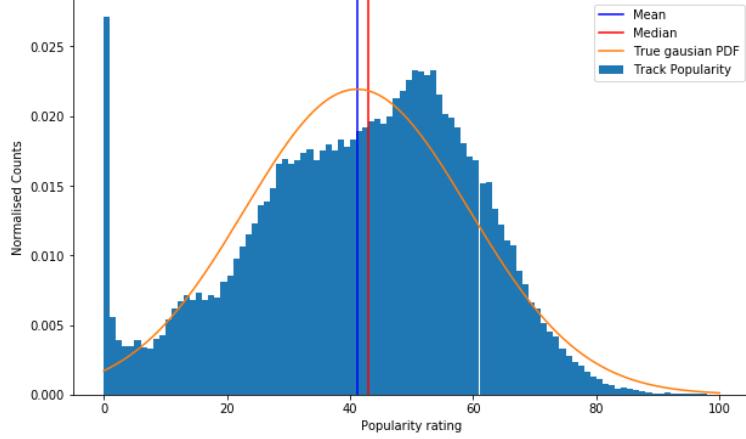


Figure 2: Histogram of the popularity feature, taken from whole dataset

Continuing, since the dataset included multiple genres, an assumption of each genre having it's own distribution seems somewhat probable, and as such, the popularity across each genre is plotted, yielding the result as seen below:

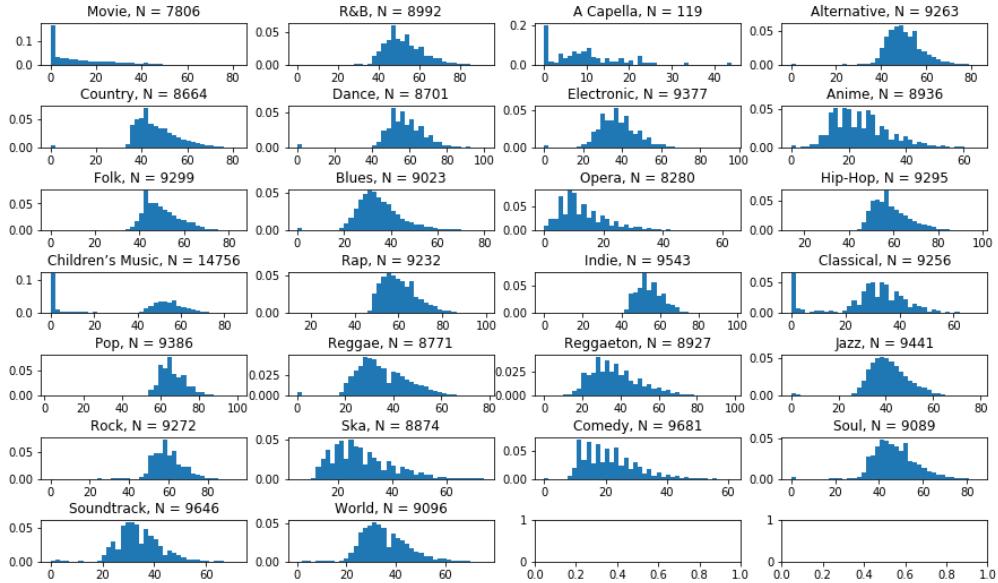


Figure 3: Histograms of popularity for each genre

From the above, it is seen that the assumption holds somewhat true. Furthermore, a few key insights is gained, namely that the “A Capella” genre only has 119 samples, and only a few genres contains the outliers with a popularity rating of 0, which was seen in the generel histogram. Indirectly, seeing that the feature distribution is somewhat different across genres, this could be examined further. Here, one could examine the means/variances across genres of different features and plotting these against each other, with an example below of the mean of the popularity for each genre.

This process however, would get rather tiresome to do for each feature, and as such, a radar plot is used instead for each genre, with each plot containing multiple feature means. It should be noted here, that for this to work, the data has been scaled to value between 0 and 1, which is also done in order to improve our models efficiency later. The radar plots yielded the results as seen below:

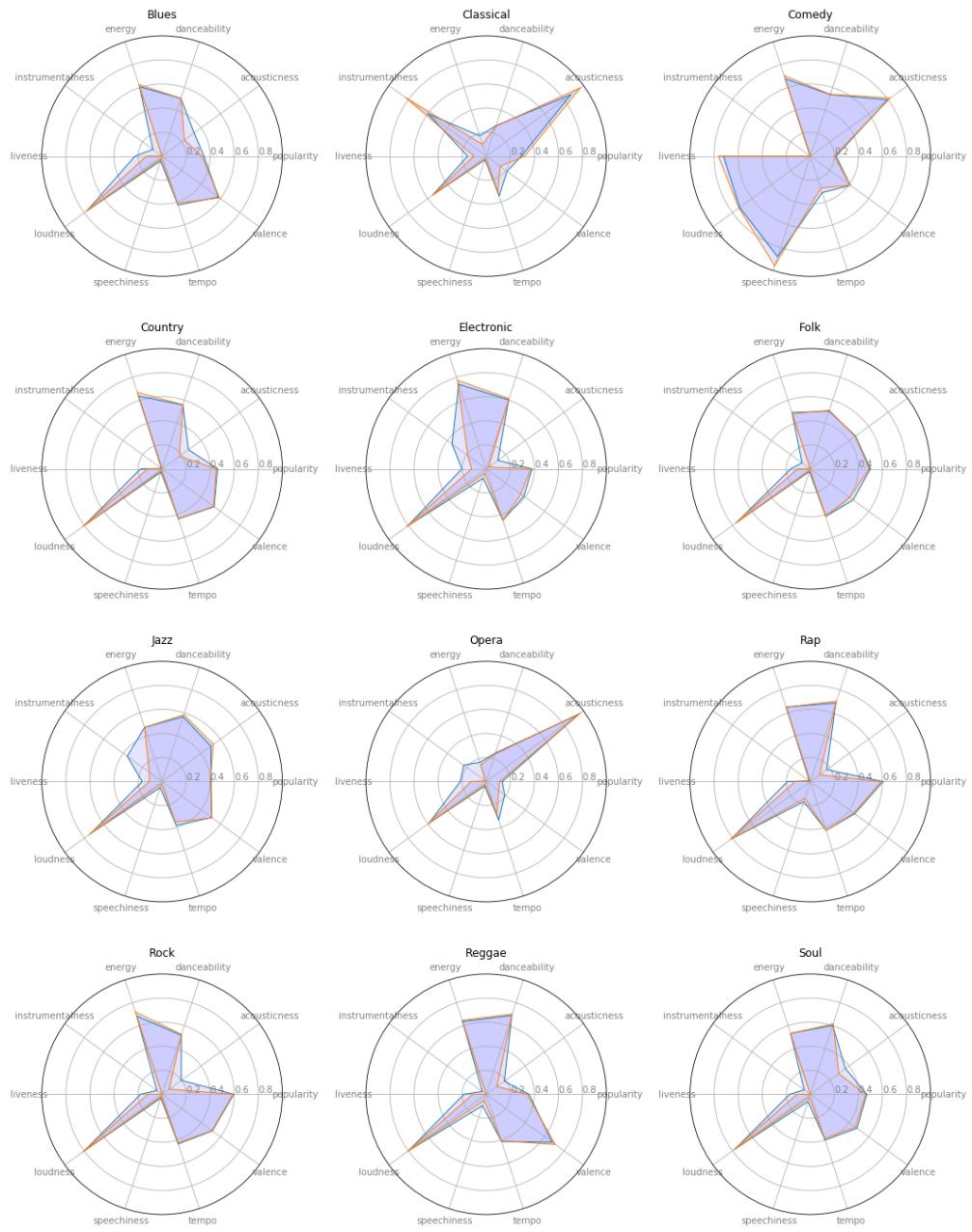


Figure 4: Mean/Median value for each feature, pr genre visualised on radar plots

Now, from the above, a distinction between the shapes of the radarplots can be seen, which is good, because in order for classifier to distinguish between genres, their features should differ in some way. There is however some issues, which can be seen looking if comparing for example Rock, Raggae and Country. The shape in the radar plot of these genres is very similar, and as such, it must be expected that trying to classify tracks included in these genres will be harder. This thought will be expanded upon later in the report, after the data has been processed, after which a PCA would be beneficial to further continue this analysis.

Now, in moving back to the regressor which was to be used to predict popularity, the correlations between popularity and other features should be analysed. Turning the beforementioned analysis around, where popularity was argued to be somewhat related to genre, the genre feature can be seen as an important feature to predict the popularity from - this leaves the rest of the features to be analysed. To easily visualise the results of these correlations, scikik learns *scatter_matrix()* is used. Using only some of the features here as an example, the results of running the *scatter_matrix()* is seen below:

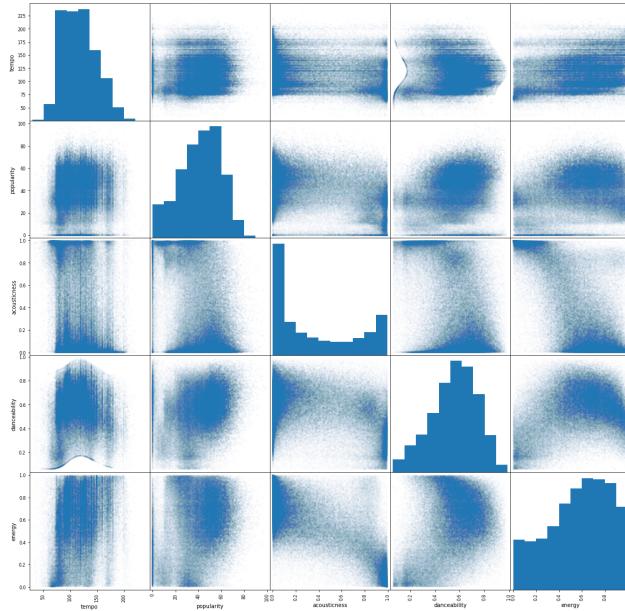


Figure 5: *scatter_matrix()* for the popularity, tempo, acousticness, danceability and energy feature

Looking above, the popularity does not seem to be correlated much with the other features, which is unfortunate. This result alone leaves the expectation for the regressor rather low, as if there is no correlation, no predictions can be made, e.g if there is no correlation, no mathematical connection between the two can be made.

Lastly, before moving on, the “*duration_ms*” feature contained significant outliers, while the feature itself had a huge range.

From the above it is seen that each feature had atleast 1 significant outlier for their duration feature, while all being distributed around a somewhat similar mean value.

2.2 Cleaning the data

From the visualisation of the data, a few key points was seen, in regards to the data:

- The popularity feature contains a significant amount of samples with a rating of 0.
- The “A Capella” genre only contains 119 samples
- The duration feature is in a too high range, and contains significant outliers
- Both the genre, key, time signature and mode are all labeled traits

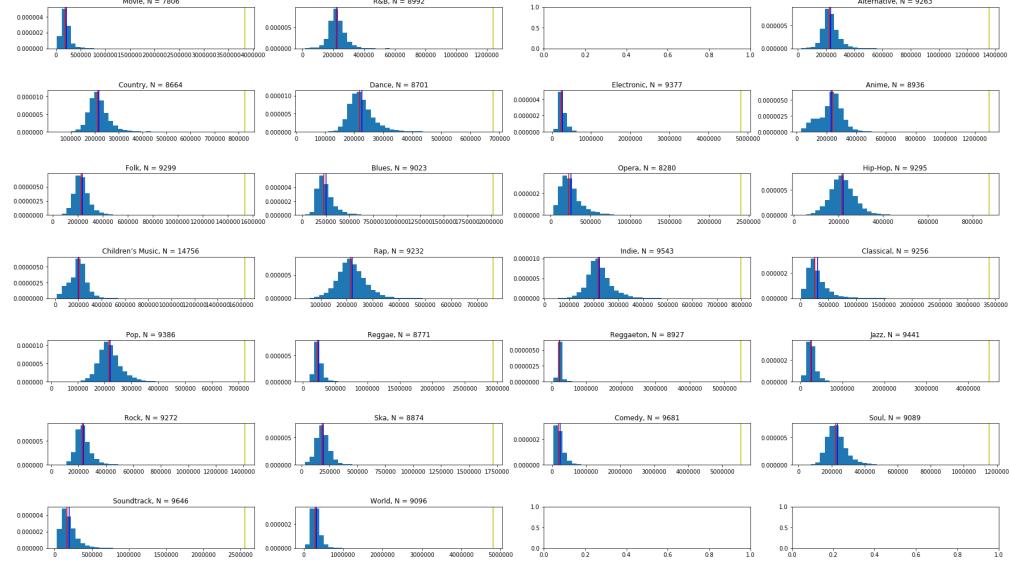


Figure 6: Showcasing the max value of outlier (yellow) of the duration feature pr genre

Since the group has no idea what causes a popularity rating of 0 - is it a new song without any data yet? Is there a threshold of listens/checks that causes the popularity rating? Is it just unpopuler with no interrest? - it has been decided to drop the samples with this rating from the dataset. This does however cause some implications, as our models ability to predict unpopular songs will be severely hindered. Futhermore, dropping samples from the groups already somewhat small dataset is not desireable as it also hinders the models ability to train and gather information. The “A Capella” genre does not contain enough samples to provide valuable information about the genre itself, no model will be able to learn from 119 samples, and certainly not distinguish between other genres from it(outliers will have too much effect as no real distribution can be made).

The duration feature was difficult to tackle, as one could no simply scale it to a range of 0 - 1, since the outliers would cause too much of a skew in the range and thus, the feature would face truncation issues. One possible solution the group though of would be to scale the unit, either to second and minutes as this would leviate the issue of the huge range found in the feature. This, however, did not help the outlier issue. To overcome the outlier issue, a manual boundry should be set in place, which would discard the outliers, and remove the from the dataset. The boundry could be found from analysis of the distribution plots, but again, this solution would cause further reduction in the dataset. Therefore, since the correlation to popularity was already low, and the distribution mean and medians across genres was so similar, the feature is removed.

For the labeled features, some encoding must be performed. For this, it has been decided to use one hot encoding. This again does not come without a price, as this effectively increases the amount of features used in the models, and thus, it must be expected that the size of the dataset should be sufficient to still be able to learn properly. Thus, looking at just the genre, we are looking to expand the dataset with an addititional 24 features - this is problematic. The mode exapnds to 2 features, and time_signature to 3(the time signatures found in the dataset was 0/4, 1/4, 3/4, 4/4, 5/4 - 0/4 and 1/4 was decided to add to 4/4, as there really is no *music* in either of those). Key would expand to an additonal 12, and there was no found any real correlation, thus the feature was decided to be dropped. The rest was encoded. **OBS:** The genre has only been one hot encoded for the regressor, as alot of scikit learns classifiers contains *labelBinarizer()*² inside them - thus, they do not expect their y-values to encoded beforehand. In the case of them expecting it to, *labelBinarizer()* should be used for in order to satisfy the models of scikit learn.

Performing the above cleaning, the popularity across the genres can be plotted again to verify that the 0 rated samples has indeed been filtered out:

²<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html#sklearn.preprocessing.LabelBinarizer>

2.3 Feature scaling and PCA

After the data cleaning, the features should be scaled in order to improve the models efficiency. The improvement is done as the models who use the Euclidean Distance between points in the feature space of our datamatrix, the distance is governed by range of our features. As such, a feature with high range will contribute more than those with low range, which will skew the final distance, as the features will no longer contribute equally(which is what we want). This is especially seen if we are to use the K-nearest Neighbours model, as this is exactly based on the Euclidean Distance.

Scaling should be performed carefully, as stated before with the duration feature, truncation issues and outlier issues must be taken into account. If scaling is done properly, there should not be any downsides in form of performance of the algorithms, as those who works on non-scaled data does not generally suffer from receiving scaled input. Furthermore, in order to further investigate the the tendencies noted from the radarplots, a PCA decomposition into a 2D or 3D plane would be beneficial.

Now, there are several feature scaling methods to choose from, but for the groups project, only standardized scaling has been used:

$$z = \frac{x - \mu}{\sigma}$$

The group decided on this scaling method due to the desire to use PCA on the dataset. The advantage for using standarized scaling in comparisson to just normalizing the data is that the PCA maximises the variance of the feature projections onto it's component axes. In order to ensure an equal weight of each feature, min max scaling(normalization) should not be used, as the outlier effect of this scaling method would degrade the performance of the PCA, in the sense that min max scaling ensures no guarantee that the variances of the features is dimensioned equally.

With the scaling done, the relation tendencies from the radarplots can be further investigated using PCA, plotting both in 2 dimensions and 3:

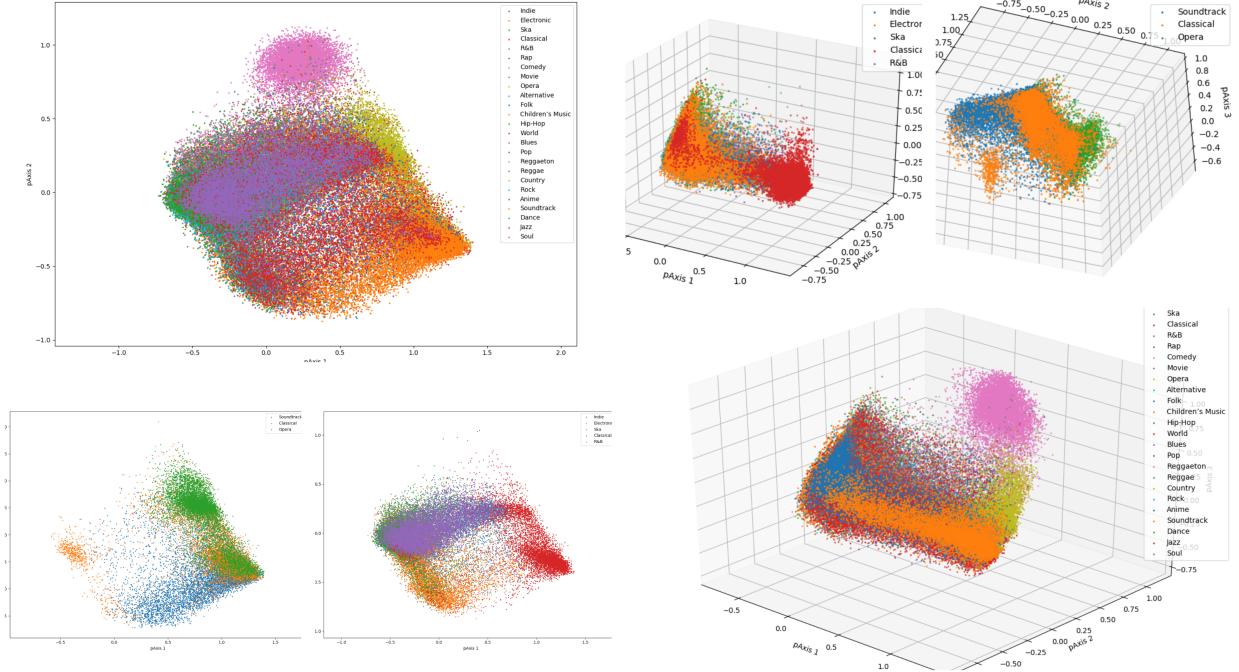


Figure 7: PCA plot using matplotlibs scatter, both in 2D and 3D. Only a sample of the results used is shown here. Both 2D and 3D contains the same samples, but reduced differently(Plots cropped to keep size at acceptable level). *Left:* 2D Plots, *Right:* 3D Plots.

From the PCA plots, the theories of some genres overlapping more than others is verified, and as such, it must be expected that the classifier will have varying results, being better at some genres than others. Some interesting things to note from the plots is the fact that a lot of the more “*mainstream*” genres, such as *Rock*, *Pop*, *Rap*, *Hip-Hop* are all grouped together. Likewise, both *Classical*, *Opera*, *soundtrack* and especially *Comedy* are somewhat distinguishable. *Classical*, *Opera*, *Soundtrack* are grouped somewhat together, as they all overlap in one place, which from intuition makes somewhat sense, as *Soundtracks* often uses *Classic* pieces, and *Opera* and *Classical* is often from the same group (Mozart's pieces are all classified as classical on Spotify, but “*Die Zauberföte*” is a mix of opera and classic pieces).

3 Genre Classification

One of the main goals of setting up the ML-pipeline with this dataset, is to investigate the prospect of predicting a genre, based on the audiofeatures of a track. This is a supervised classification task, as the desired output is one distinct category of a predefined set of possible options.

3.0.1 Algorithm Selection

Before training can begin, an algorithm, or set of algorithms, must be selected for the job. The choice of algorithm has been affected by which algorithms the group has had previous experience with, and the primary basis of comparison has been their success at performing similar classification tasks. Due to the very good results with the K-Nearest Neighbors (KNN) algorithm in the MNIST search quest exercise for O3, this was the initial choice for the genre-classification task.

In addition, the group has chosen a second type of model, Fully Connected Neural Networks, for comparison. This algorithm seems like a good complementary alternative to the K-nearest Neighbors algorithm, primarily due to its flexibility and scalability. Since there are more ways to configure a Neural Network, it could also prove more difficult and time-consuming to find the optimal initial configuration of the network. But this also means that it has a good chance of being able to cover any areas where the K-Nearest Neighbors is insufficient, making it a good fall-back candidate.

3.0.2 Data processing and structure

The input for the classification model consists of all the scaled audiofeatures, along with one-hot encoded “mode” and “time_signature” features. This gives the model a total of 15 features to use a basis for assigning a genre to each input.

The classes themselves are created from the “genre” column of the dataset, which is encoded through a LabelEncoder, assigning an integer to each possible value, replacing the original string. This means that the classes will be represented by integer values ranging from 0-24, and this data will be the y-values for our model.

Before the model is trained, the data is split into training and validation sets. The model is fitted to the training-set, and afterwards the validation-set is used to test how the model responds to previously unknown samples. This is an important part of the process, as it can be used to identify whether or not the models ability to generalize has been compromised by over-fitting to the training data. The proportion of the dataset assigned to the testset was selected to be ≈ 0.3 .

3.1 K-Nearest Neighbors

As mentioned, the KNN classifier has been selected as the primary candidate for genre-prediction. The configuration of the model is described below, followed by the results.

3.1.1 Hyperparameter Search

In order to get the best possible performance, the model is initialized multiple times with different combinations of hyperparameters, so that the best values for each hyperparameter can be found by evaluating the performance of each combination. The first search performed on the KNN model considers the hyperparameters shown in Table 1 :

Table 1: KNN hyperparameter-space to search

Name	Value
n_neighbors	[3, 4, 5]
weights	['uniform', 'distance']
algorithm	['ball_tree', 'kd_tree', 'brute']
p	[2, 3, 4]

As the `n_neighbors` was set relatively low in this first search, larger values for this parameter were used in a later search, where the values 10, 20, 50 and 100 were tested for `n_neighbors`. After searching through the selected parameter space, the best hyperparameters were found to be:

- `n_neighbors=100`
- `weights='uniform'`
- `algorithm='ball_tree'`
- and `p=2`

These are the final parameters given to the K-Nearest Neigbor model.

3.1.2 Results

Initializing a model with the found hyperparameters and fitting it to the training data, has yielded the results shown below:

3.1.2.1 Performance metrics These performance metrics for the KNN model are taken from sklearn's classification report, which contains the following four metrics:

- *Precision* - The fraction of positives that were true positive, defined as $\frac{tp}{tp+fp}$
- *Recall* - The fraction of positive samples the model was able to find, defined as $\frac{tp}{tp+fn}$
- *F1-score* - The weighted mean of *recall* and *precision* (both are equally weighted in this case)
- *Support* - The number of samples in the validation set

In combination, these four metrics give a good picture of the models overall performance.

Table 2: Classification results (summary)

	Precision	Recall	F1-score	Support
Accuracy			0.38	66778
Macro avg.	0.38	0.38	0.37	66778
Weighted avg.	0.37	0.38	0.37	66778

Table 3: Classification results (pr. class/genre)

Class/Genre	Precision	Recall	F1-score	Support
Alternative	0.23	0.20	0.21	2687
Anime	0.51	0.38	0.44	2660
Blues	0.36	0.30	0.32	2640
Children's Music	0.26	0.19	0.22	3289
Classical	0.53	0.51	0.52	2423
Comedy	0.95	0.93	0.94	2708
Country	0.25	0.42	0.31	2563
Dance	0.20	0.17	0.18	2595
Electronic	0.48	0.48	0.48	2736
Folk	0.24	0.31	0.27	2692
Hip-Hop	0.27	0.40	0.32	2783
Indie	0.16	0.11	0.13	2835
Jazz	0.36	0.33	0.34	2813
Movie	0.57	0.27	0.37	1609
Opera	0.67	0.86	0.75	2389
Pop	0.30	0.40	0.34	2861
R&B	0.20	0.17	0.18	2718

Class/Genre	Precision	Recall	F1-score	Support
Rap	0.24	0.18	0.21	2793
Reggae	0.40	0.38	0.39	2588
Reggaeton	0.45	0.58	0.51	2636
Rock	0.25	0.32	0.28	2744
Ska	0.52	0.45	0.48	2762
Soul	0.21	0.11	0.14	2739
Soundtrack	0.53	0.69	0.60	2821
World	0.37	0.36	0.36	2694

3.1.2.2 Distribution of predictions pr. class The models performance is examined further by plotting how the models predictions for the test data is distributed for each genre (true and false positives).

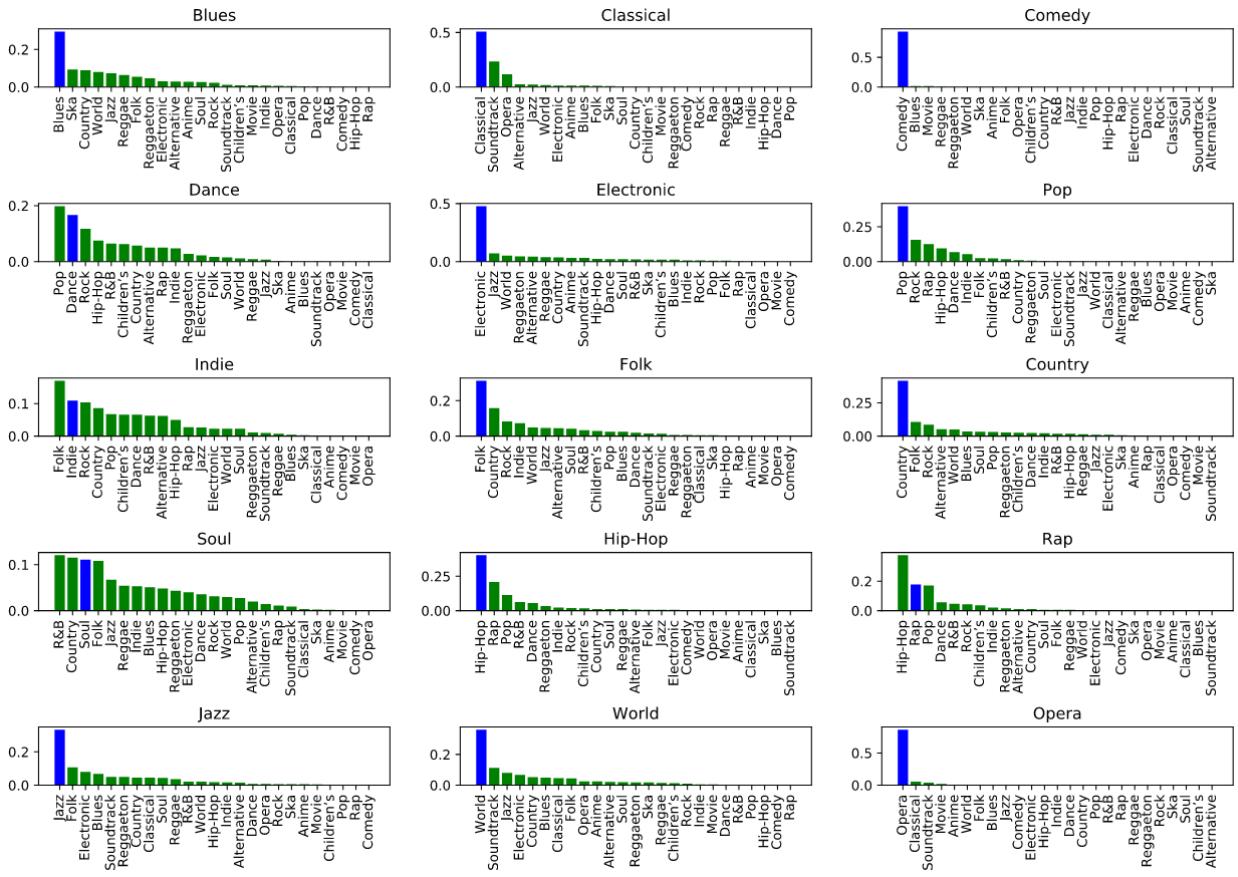


Figure 8: Distribution of correct and incorrect predictions pr genre

3.1.3 Discussion