Predicting Music Genres with Spotify Data

1. Selecting Data set

There are plenty of articles online in which people attempt to predict the popularity of a song based on a range of metrics. We thought it would be interesting to try and predict the genre of a song based on these metrics. One would expect that songs from a genre share similar characteristics. This could help to categorise songs from unknown artists with those characteristics. The dataset that we used is provided by Spotify, it contains ~40.000 tracks from 1921 to 2020 with 20 columns that include information about musical compositions. The genres the songs come from are Pop, Rock, R&B, Latin, Rap and Edm.

2. Preparing Data Set

2.1 Exploratory Analysis

When preparing the dataset we first had a look at the variables, their meaning, their distributions as well as their distributions over the genres. For the variable names and their description see Appendix 1, for the variable distribution Appendix 2 and for their distributions over the genre see Appendix 3.

The dataset contains three binary variables: popularity, mode and instrumentalness. While population and mode are quite balanced, almost all songs were classified as not instrumental. Furthermore we have four categorical variables: track name, artist name, decade and genre. Lastly there are a range of numeric variables. Some of them are more normally distributed (danceability, energy, valence, normal) and others more skewed (speechiness, accoustcness, instrumentalness, liveness).

After having a good overview of our explanatory variables, we had a look on how they are distributed over the genres (Appendix 3) to already get a feeling on how different the genres were. We found that especially Rock and Latin songs behaved unsimilar to the others. For rock danceability and valence were much lower and speechiness and instrumentalism higher, while for Latin energy was much higher and accousticness not represented.

2.2 Feature Engineering

As the dataset was provided by Spotify, it is already very clean and so there was no need to handle missing data or outliers. We only had to remove one duplicate song.

We decided to create new features based on the square of all features. Squaring features allows to represent non-linear relationships. In regards to the categorical variables, we decided to drop track and artist name as an encoding would have not made sense. The decade we converted into a numeric variable as year imply ordinality. The genre is our target and we decided to encode it in two ways, first based on its representation in the dataset based on counts and second just with numbers from 0-5. Later we decided to only go with the quantitative count encoding.

As some of the models will be sensitive to distributions we decided to have a copy of scaled and not scaled data for training and testing to observe the effect of scaling on model performance. We were choosing between Min Max, which preserves the shape of the distribution and only changes the range and the standard scaler which standardises to a zero mean and unit variance. As outliers are no problem, and we were interested in standardisation we decided on the standard scaler.

Then we removed features with low correlation, we dropped all features that showed a small absolute correlation <0.1

Lastly, we completed a random train test split with 80% of the data as a training set.

3. Selecting ML techniques

Since our task is a supervised classification we decided to use the most common models and add in a multilayer perceptron.

3.0 Baseline with Logistic Regression

In order to evaluate the models later in a better fashion we have decided to form a Logistic Regression as a baseline. This allows us to not compare the results against random choice but against a very basic model.

3.1 KNN

The first model we choose is the k nearest Neighbour classifier, it has the intuition that similar samples are close to each other meaning that if music in a genre is similar, those music should have a small distance. The mode is a simple model to begin classification problems with and can do well in practice with large samples of data. However with large datasets the storage of data becomes an issue as all points are compared to each other.

3.2 Support Vector Classifier

The second model we choose is the support vector classifier; it has the intuition that observations can be separated into classes finding an optimal hyperplane. Basically, the hyperplane wants to be as far as possible from every class, in other words, maximising the margin. All in all, it's an intuitive classifier that works effectively when there's a clear separation margin, in numerous dimension spaces and even when the number of samples is not too large as we do not have to compare each data point with each other.

3.3 Random Forest

The third model we chose is a random forest, it has the intuition that a large number of relatively uncorrelated trees operating as a committee will outperform any of the individual decision trees. So, the fundamental concept behind random forest is the wisdom of crowds. Random forest works well with both categorical and numerical data and no scaling or transformation of variables is usually necessary. Further it can handle linear and non-linear relationships well. What is interesting, is that Random Forests generally provide high accuracy and balance the bias-variance trade-off well. Since the model's principle is to average the results across the multiple decision trees it builds, it averages the variance as well. Negative sides are the hard interpretation (black box algorithm) as well as the intensive computation with large data sets

3.4 Neural Network

The last model we choose is a Multilayered perceptron, it is a neural network that takes the brain as an inspiration using networks of neurons that fire based on activation functions. The pros are that the model trains itself throzúgh forward and backward propagation and can take advantage of information from higher levels. However, whats negative is that neural networks are black boxes, meaning the results are not explainable and the training is very computationally expensive

4. Running techniques

4.1 K Means Grid Search + Scaling

We run the KNN tuned and not tuned, and although we are aware that the model should be used with scaled data we tried it with and without to compare later. The not tuned model is run with the default parameters. We will only describe the tuning that we have done with Grid Search. When tuning we looked at the k neighbours, the distance measure and distance voting. For the KNN we need to define the number of neighbours we will consider. Larger values of K will have smoother decision boundaries which mean lower variance but increased bias. One should try and keep the value of k odd in order to avoid confusion between two classes of data. In general, practice, choosing the value of k is k = sqrt(N) where N stands for the number of samples in your training dataset. In our dataset this is sqrt(32879) = 181, so we searched for k in 5:241. The distance can be calculated based on euclidean (the square root of the sum of differences between the components of x and y), Manhattan (sum of the absolute values of differences between the components of x and y coordinate) or Minkowski (generalisation of the other two techniques). The majority voting of the class can be done uniformly, meaning all neighbours are weighted equally or based on distance.

4.2 SVC Grid Search + Standardising

We run the SVC with both scaled and not scaled data, and then we run it again with the parameters tuned. The following parameters can be tuned: C with options between 0.1 (SVM optimizer low tolerance) and 10 (SVM optimizer high tolerance); and kernel (function types to separate the datapoints) with options linear (should always try this one) and RBF (similar to KNN but works better with high dimensions spaces). We could also try other kernel modes such as polynomial or sigmoid.

4.3 Random Forest Grid Search + Scaling

Like the other three models we run the random forest with scaled and not scaled data as well as tuned and not tuned. As the random forest however has more parameters we utilised the random grid search to minimise the amount of fits we run. When tuning we changed the number of trees in the forest (no of estimators 200-2000), the maximum number of features considered for splitting at each node (max feature), the number of levels in each tree (max depth 10-110), the minimum number of data points in a node before the node is split (min sample split 2 , 5 or 10), the minimum number of data points in a lead node (min sample leaf 1, 2 or 4) and the method for sampling data (bootstrap replacement or no replacement). The larger the trees are, considering only small data points per leaf the more overfit a model can get.

4.4 Neural Networks

We used a Multi-layer Perceptron (MPL) as a Neural Network classifier. We ran this model as well with scaled and unscaled data, both with default settings. Further, this classifier makes use of a cross-entropy loss function (log loss function), which is commonly used for classifiers that give a probability between 0 and 1 as output. In regards to tuning, we manually tested how many neurons were best in the hidden layer After that we ran a gridsearch to find the best combination for the other parameters. We tried to use three kinds of solver: lbfgs, sgb and adam (stochastic gradient-based optimizer that works better with relatively large datasets). The learning rates we tried were constant and adaptive, meaning that the learning rate either stays constant or it stays the same as long as training loss keeps decreasing. Each time two consecutive epochs fail to decrease the learning rate will be divided by 5. Lastly, for activation we tried identity, logistic, tanh and relu, to make sure that we have a broad range of different activators

5. Compare Solution

The models will be evaluated based on their accuracy score. All models were trained on the training data and tested against the test data. The best **KNN** accuracy is: 0.895 and was achieved with scaled data Grid Search using the following parameters: {'metric': 'manhattan', 'n_neighbors': 21, 'weights': 'distance'}. The best **SVC** accuracy tuned with Grid Search and using scaled data is: 0.987, with the following parameters {C: 10, kernel: linear}. The best **Random Forest** accuracy is 0.9997 and was achieved with unscaled data and using Grid search with the following parameters {'n_estimators': 600, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 80, 'bootstrap': True}
The best **Neural Network** accuracy was 0.999 using scaled data with the following settings {neuron hidden layer: 100, solver: adam, learning rate : adaptive, unit function: rectified linear}

	Logistic Regression	KNN	SVC	Random Forest	MLP
Accuracy non Standardised	0.596	0.661	0.488	0.9998	0.45
Accuracy Standardised		0.851	0.968	0.98978	0.99
Best Accuracy Tuned		0.895 (scaled)	0.987 (scaled)	0.9997 (not scaled)	0.99 (scaled)

The best performing model has been the random forest without any scaling of data.

6. Reflection

It was an interesting experiment to run a range of different classifiers on the same problem. Seeing how models behave differently with scaling has been the most impressive remarkt. We learned that since the random forest is a rule based algorithm that evaluates based on the range of each feature individually and not as a group, scaling is not needed. Even further, the information loss that comes with normalising the distributions of the variables can even negatively impact the results. Knowing this can be very helpful when dealing with outliers or when the datas distribution is very unique for different features (as in our case with Rock and Latin music).
In comparison to the random forest, since the kNN is a distance based algorithm, it was very much impacted by scaling as it gives higher weight to variables which have higher magnitude. The SVC is also very much impacted by feature scaling, since SVM are based on distance and margins. The parameter tuning did improve some by a small margin, however, it didn't affect as much as the scaling did. It came as a surprise to some of us when the kernel mode linear turned out to be a better classifier than rbf. Lastly, running the neural network with unscaled vs scaled data showed us also immediately its sensitivity for feature scaling. The unscaled data gave us an accuracy of 0.45 and the scaled data 0.99.

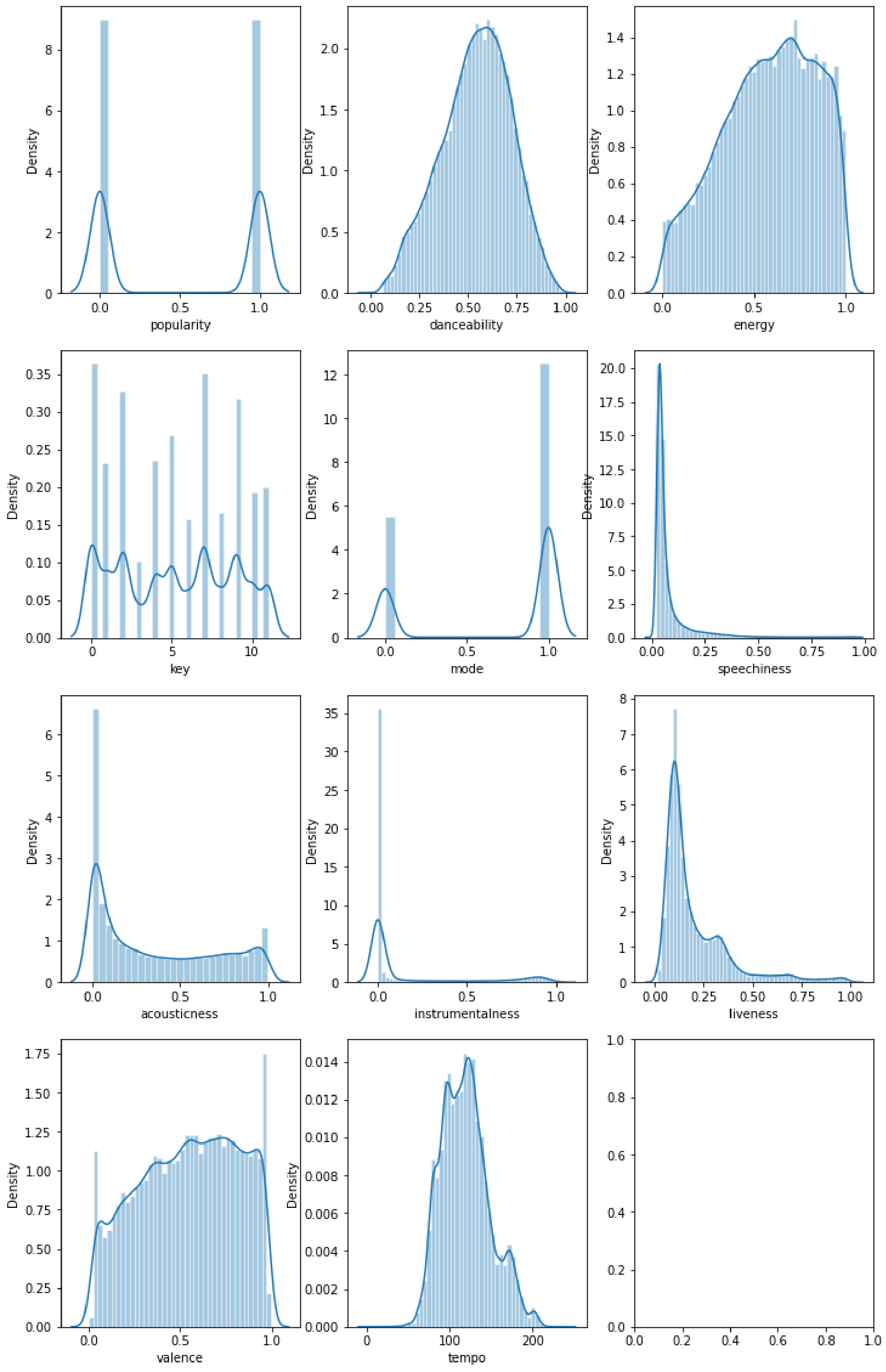
When repeating the prediction we decided that it would be interesting to also add logged features, not only squared ones, as they are useful with skewed data, This would have maybe decreased the need to scale the data, and give additional information to the random forest and Neural Network.
Furthermore we would have a look at the individual correlations of the variables to each other, to create a more strict feature selection by dropping highly correlated features. In

this way we could avoid overfitting through too many features. Or we could have used a recursive feature selection algorithm to help us with that.
Lastly, it would be interesting to have a look at the balance of the dataset. As not all genres are equally distributed it might make sense to sample more underrepresented genres. This could improve the general prediction.

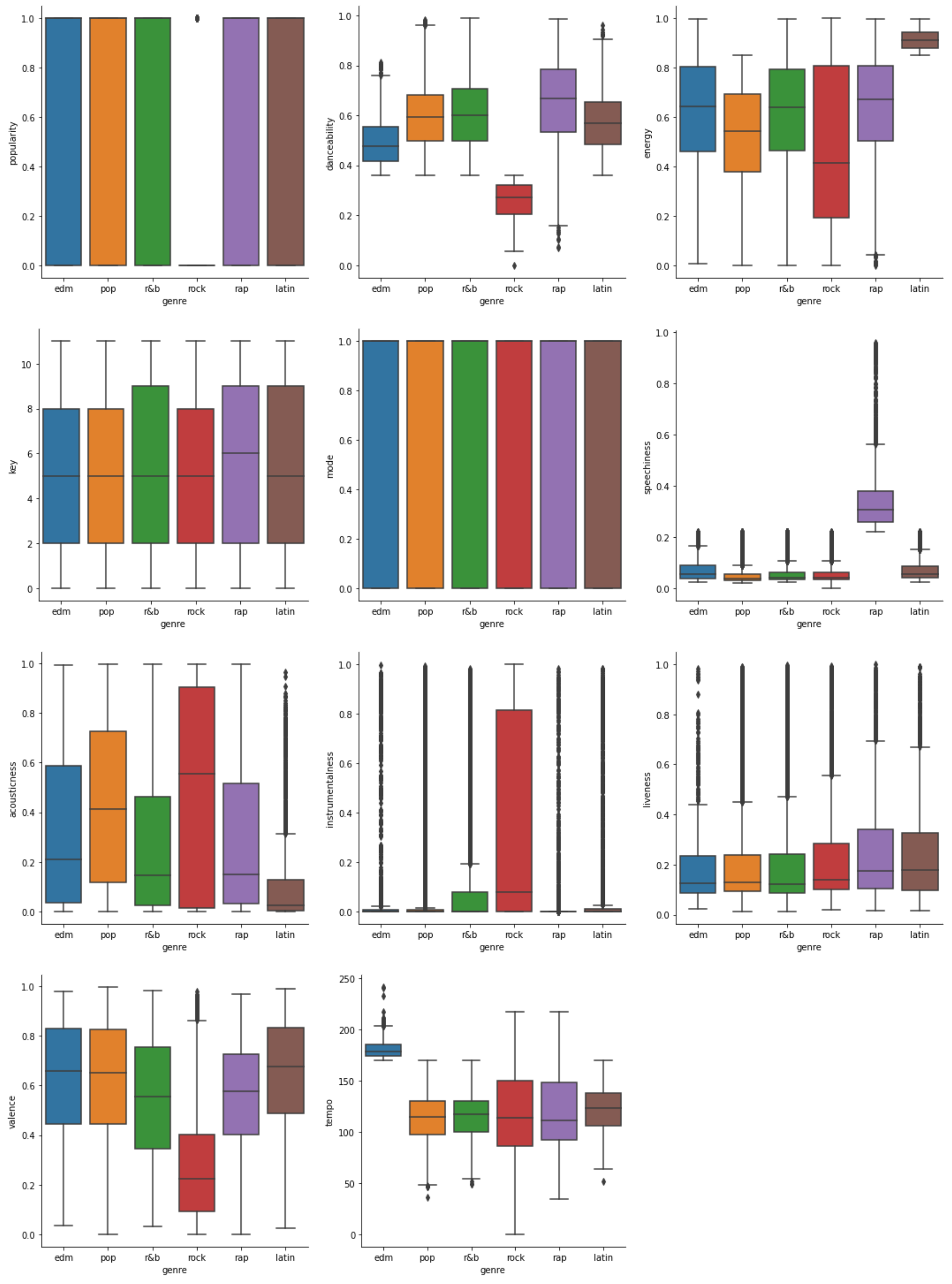
Appendix 1 - Variables

- **artists**: artist's name
- **name**: name of song
- **acousticness**: 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**: 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**: 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.
- **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. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- **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).
- **popularity**: calculated by algorithm and is based, in the most part, on the total number of plays the track has had and how recent those plays are.
- **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.
- **liveness**: 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**: 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 typically 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. talk show, audio book, poetry), the closer to 1.0 the attribute value.
- **year**: release year. ranges from 1921 to 2020
- **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.
- **key**: The key the track is in. 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..
- **duration**:: duration of track in milliseconds, integer ranging from 200000 to 30000

Appendix 2 – Distribution plots



Appendix 3 - Distribution of Variables over the genres


popularity
danceability
energy
key
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
genre
edm
pop
r&b
rock
rap
latin