Assignment 3

Name: Mo Mofatteh **Student Number:** 300551321 **Date:** 10/09/2021

Core:

Data Overview:

The data consists of 18 columns which represent the songs in numerical, categorial, and textual attributes. Of these attributes, the obtained date of the songs is either 4th or 3rd of April and it's not adding much helpful information. Now, if we concatenate these data to build one single dataset, we'll see that the number of instances in each genre are equal, as described in the data description, and the genres are balanced. In addition, there are no missing values in the data. Therefore, we can use different classification metrics efficiently skipping some preprocessing steps to balance the data or remove instances with missing values.

Feature Patterns:

Looking at the correlations heatmap, I noticed that most of the attributes have a very weak relationship based on their correlation scores. For instance, duration of a track has correlation scores of less than 0.2 for all the attributes. This shows that the genre of a music does not affect the duration of it, nor vice versa. The same pattern exists for the liveness attribute. This type of attributes can be neglected and dropped in the next step, when preparing the data for machine learning techniques.

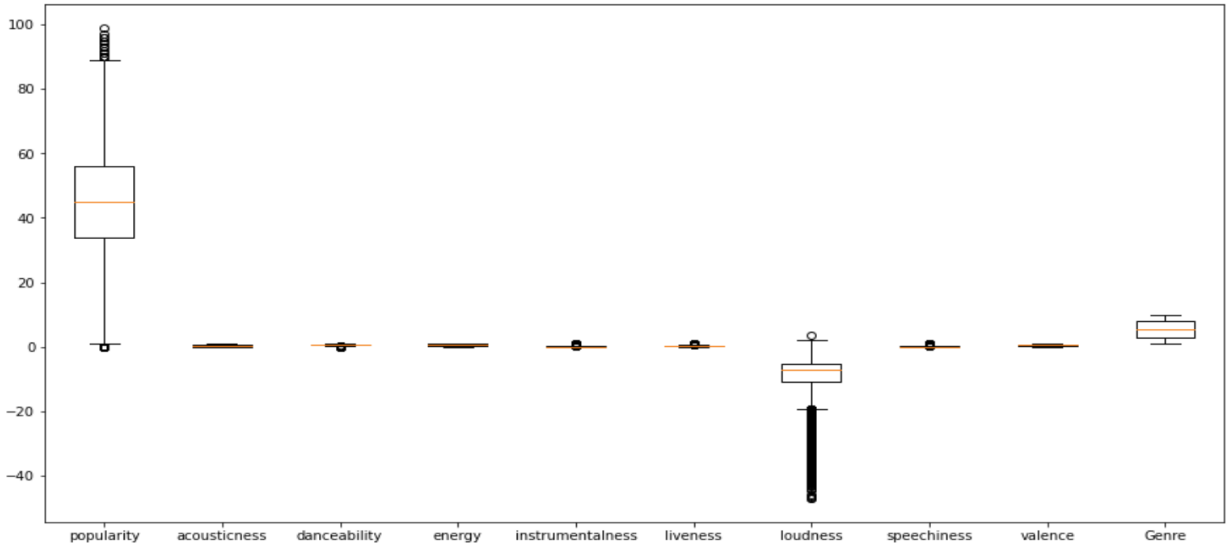
On the other side, there are some high correlation scores. The highest one is for energy and loudness with the score of +0.84. Even though this value is not showing a cause & effect relationship between these attributes, but it is indicating that the louder the song, the more energetic it is, or vice versa. An example could be rock songs, as they are mostly loud with high energy. Two other high correlation scores are associated with acousticness. Acousticness and loudness have a negative correlation, -0.73, indicating the less acoustic a work is, the louder it is, and vice versa. As expected, the same relationship sits with acousticness and energy. Adding the other high value in the heatmap, which is between instrumentalness and loudness with the score of -0.53, we can guess there is a close relationship between these 4 attributes.

Looking at the feature interactions table for these 4 attributes, partly shown below, we can see that the mean value for some of these interactions are pretty low such as instrumentalness and energy (and acousticness). This shows that these variables together are less likely to be helpful in finding the genre of a music track. This value for the combination of loudness and energy (acousticness) is very higher which indicates a possible additional hint on finding the genre of the music because this number shows that when these attributes are combined, they add a noticeable change to the value generated.

	loudness energy	loudness acousticness	loudness instrumentalness	energy acousticness	energy instrumentalness	acousticness instrumentalness	loudness energy acousticness	loudness energy instrumentalness	loudness acousticness instrumentalness
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	-4.111055	-4.334923	-2.719642	0.112302	0.075365	0.098732	-1.072789	-0.721720	-1.960984
std	1.652580	7.157088	6.068026	0.121630	0.168075	0.245904	1.259489	1.526404	5.626687
min	-15.996744	-46.374195	-44.879255	0.000000	0.000000	0.000000	-11.202566	-13.581236	-44.654859
25%	-5.028419	-4.990452	-1.211789	0.012844	0.000000	0.000000	-1.753264	-0.439366	-0.047402
50%	-4.034604	-0.950975	-0.001074	0.068052	0.000088	0.000008	-0.552099	-0.000612	-0.000051
75%	-3.132931	-0.109765	0.000000	0.180073	0.038481	0.005628	-0.079458	0.000000	0.000000
max	3.732768	0.196808	1.748253	0.824992	0.957330	0.977126	0.194053	1.711540	0.012432

Visualization:

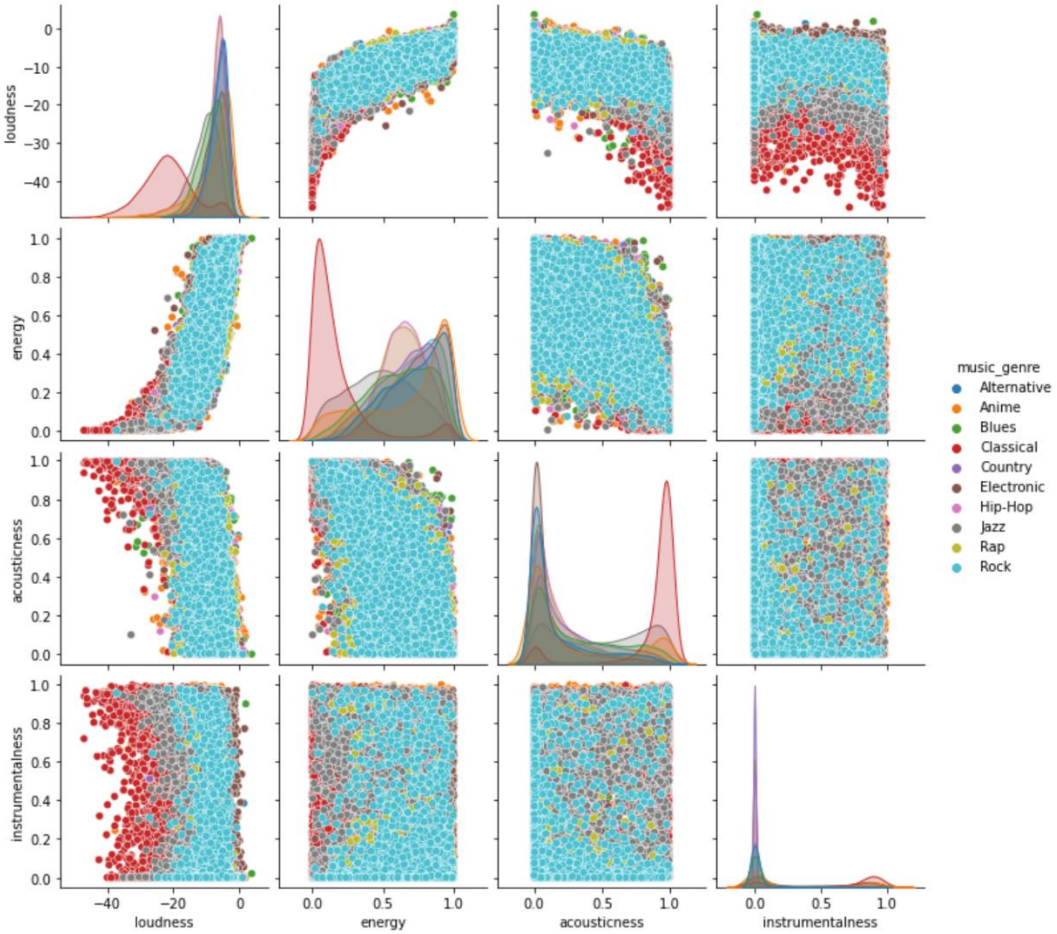
Before going over the features discussed in the previous section, let's look at the distribution of the data on a box plot. The following plot shows, this distribution for the numerical attributes, excluding the duration since they are very large and the distribution for other attributes can't be viewed clearly. As expected, the popularity of a song is a score from 0 to 100 and we have 10 different genres, so that is a number from 1 to 10. As we can see the loudness values are all negative spreading from -50 to 0. Aside from that all the other numerical attributes have value close to zero which means the small decimal values will be decisive in our predictions.



Now, looking at the following pair plot, showing the relationship of the 4 attributes discussed earlier, we can draw some interesting conclusions, listed below:

- Loudness and energy have a noticeable relationship, looking like a 3rd degree polynomial.
- Classical songs have the lowest loudness values while Electronic and Rock songs, in order, have the highest values in the loudness-instrumentalness comparison.
- Classical songs' energy is sitting in the lower half of the available values.
- Loudness and acousticness have a close relationship with loudness decreasing as the acousticness increases.
- Most of the 2-D comparisons have an even distribution showing that more attributes are required to decide on genre of the music.

- From the acousticness univariate diagram, we can see that classical tracks have high acousticness while the other genres mostly lay in the lower values section for this attribute.



Completion:

Initial Design:

With the business goal of "showing music tracks similar to the ones a user listens to", we came up with the datamining goal of "predicting the genre of a music based on the different attributes provided using the machine learning techniques to analyze the data and use the appropriate methods". According to this goal, I decided to use logistic regression as we are predicting a categorical attribute in our problem. It's also easier to implement and efficient to train.

For my initial design, I tried doing some simple pre-processing steps and skipping some of the data preparation intentionally to see the effects of it. So, I started with encoding the categorical attributes and using a unique number for each value and then I normalized the data using the Z-score normalization which sets the mean of the distribution to 0 with a standard deviation of 1. In the next step, I used my data analysis to eliminate those variables that were defined unnecessary for my prediction. Duration because I thought the durations of the songs don't vary much and their values were all high, and deleted the date they were obtained because they were all obtained on April 4th and a few on April 3rd. I also deleted the instance id from my training set, so it's not used in the algorithm. Aside from that, I kept all the remaining attributes unchanged as I wanted to obtain an initial view of how accurate the system would be considering all the provided characteristics. And also, since we have separate data for training and testing, I used the entire file for training and did the testing on the other file.

In my opinion, this prediction would not be as accurate as expected, since there were many attributes in the previous visualizing methods, that did not contribute much to the characteristics of a track and which genre it belongs to. The score and the rank of this prediction on the leader proved my point:

3	Hayden [Tutor]		0.57750	2	9d
4	Conor Foran		0.55926	1	18h
5	GMT+8		0.54750	2	20h
6	Leon Menzies		0.53485	8	1d
7	The Silent Tiger		0.52029	1	~10s
	Your First Entry ↑ Welcome to the leaderboard!				
8	Raphael Dan Gueco		0.50117	1	3d
9	pun2311		0.44955	4	3d
10	Ivan Ivanov		0.43176	3	19h

The interesting thing that I noticed from this score, was how close it was to the score that I received from the "test and score" object on the training set on Orange, shown below. It's interesting because we are using different testing sets for these two procedures, but the results are still pretty similar.

Model	AUC	CA	F1	Precision	Recall
Logistic Regression	0.994	0.524	0.520	0.522	0.524

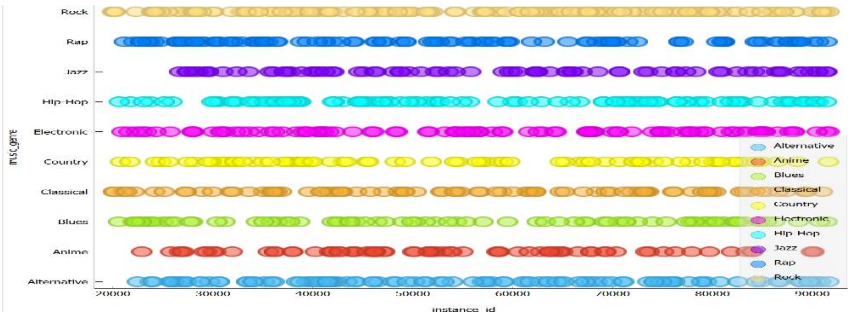
Submissions:

1. Minimum preparation done 0.52
2. PCA added 0.47
3. Outliers removed and C set 0.2 0.48
4. Random Forest with 100 tress 0.53
5. Used Neural Network 0.56
6. Used logistic activation method in Neural Network 0.57
7. PCA and imputation methods applied on Neural Network 0.57

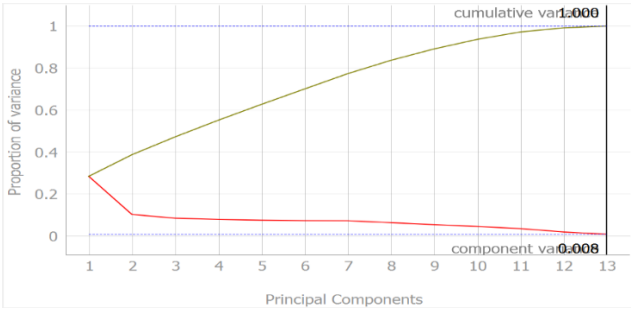
Intermediary Designs:

Neural Network using pca-generated features excluding the outliers:

After my initial design, I tried making variety of changes to my technique and the data that I'm using in those techniques. Starting with the data, I removed the outliers from my data using the local outlier factor with Euclidean distances on Orange. Looking at the scatter plot of the outlier's data, we can see that the outliers are almost evenly distributed among the genres, and it won't make our analysis imbalance.



In addition to the outliers, I once again investigated the original data for any new hints that could possibly enhance the design's power. Looking at the data table after importing data, I found some odd values for two columns of duration and tempo, -1 for duration and '?' for tempo. Therefore, I used the impute object on Orange and gave these instances the average value of tempo in the entire data as their tempo value. For the duration attribute, however, since it was not very effective, I kept it out of my data. Furthermore, I used the PCA object on orange to construct new features with possibly more contribution to the genres. The following diagram shows the effectiveness of the attributes, in terms of variance, with the number of attributes as the altering variable.



For the Machine learning tool, after investigating lots of techniques, such as kNN and random forest, I found none of them as accurate as Neural Network. Using the logistic activation of this algorithm gave the score of around 0.57, as shown below, which was the highest. I passed all the original attributes and the ones generated by PCA to increase the accuracy further.

Model	AUC	CA	F1	Precision	Recall
Neural Network	0.923	0.567	0.565	0.568	0.567

Moving to Jupyter notebook again and adding more feature selection and construction techniques.

After talking to the tutors and a friend, Phil Edie, I found that there were other things that could have been done to increase the score, such as ICA and general programming or some characteristics of the attributes like digits in the name of the tracks, which were hard/impossible to do on orange. Aside from that, testing a model on orange took so long and so few models could have been tested to come up with the best one of all. This prevented me from finding if the model is overfitting the data as it took so long to conduct a different model with different parameter values. Therefore, I decided to start over and use python.

I followed the same steps for pre-processing as before with the imputation of the missing values for duration and tempo using kNN imputer. Also, as I mentioned earlier, I realized that there are some patterns repeated for some genres more often than others, such as having numbers in the track name for classical songs and Japanese alphabet in the artists names of Anime tracks. Therefore, I added new features indicating if those two factors are true and encoded it later using label encoder.

For the classification methods, I used the MLP classifier of the neural network with varying alpha values. By changing the alpha values, I found that the scores increase with the decrease of alpha. However, when we go below 1e-6, the scores decrease again, which could show the data has been overfitted. For the input data, I passed the original features and ICA-generated features separately. This way, I was able to test the effectiveness of ICA which did not come handy and the model with the original features beat it by 4 percent. The following figures, shows the results.

```
In [69]: from sklearn.neural_network import MLPClassifier
         clf = MLPClassifier(alpha=1e-6)
         clf.fit(preprocessed_X_train, y_train)
         print("Accuracy = ", accuracy_score(clf.predict(preprocessed_X_test), y_test))

         Accuracy =  0.588
```

```
In [43]: from sklearn.neural_network import MLPClassifier
         clf = MLPClassifier(alpha=1e-6)
         clf.fit(ica_X_train, y_train)
         print("Accuracy = ", accuracy_score(clf.predict(ica_X_test), y_test))

         Accuracy =  0.5335333333333333
```

Final Design:

For deciding on my final design, I decided to implement the method we used in the first assignment and calculate the score for each input data (the original features, PCA features, ICA features, and GP features) using the classification techniques and their most important parameter used in the first assignment and compare the values. The scores for the best input data is shown below:

kNeighbor
GaussianNB
Logistic R.
Decision Tree
Gradient Boosting
Random Forest
MLP

```
[0.41413333333333335, 0.41733333333333333, 0.44566666666666666, 0.46946666666666664, 0.4846]
[0.37046666666666667, 0.3734, 0.408]
[0.5374, 0.5368666666666667, 0.5371333333333334, 0.5371333333333334, 0.5370666666666667]
[0.3166, 0.3698666666666667, 0.4169333333333333, 0.4716666666666667, 0.49273333333333336, 0.5117333333333334, 0.520266666666
6667, 0.5234, 0.5213333333333333, 0.5181333333333333]
[0.5696666666666667, 0.5811333333333333, 0.5811333333333333, 0.5754, 0.5749333333333333, 0.5647333333333333, 0.5566, 0.55546
66666666667, 0.5542, 0.553]
[0.41313333333333335, 0.4688, 0.5014, 0.5252666666666667, 0.5382, 0.5564666666666667, 0.5612, 0.5653333333333334, 0.5714, 0.
5712666666666667]
[0.5855333333333334, 0.5867333333333333, 0.5876666666666667, 0.5015333333333334]
```

These numbers indicate the scores for different classification models separated by the brackets. As you can the best one belongs to the last set, MLP, with scores decreasing as the alpha decreases. Therefore I decided to test even lower alphas and I figured 1e-10 is the best value.

To finalize this section, I have listed below my choices for each step of the process and my reasonings for them.

- Data Pre-Processing:
  - Imputed the missing values of duration and tempo using the kNN imputer, this way the new values would not affect the original data hugely and might be pretty close to the actual value.
  - Encoded mode and key using label encoder to be able to use them in my analysis and prediction.
  - Normalized the data using standard scaler to avoid the great range of values some attributes, such as duration, have and their unnecessary impacts.
- Feature Selection and feature extraction:
  - Removed the obtained date as it was the same for all of them.
  - Created new featured based on the patterns existing in the string variables and removed the actual variables.
- Feature construction:
  - None because all of them and especially general programming reduced the prediction score.
- Classification technique:
  - MLP from the neural network library because it had the highest score amongst other techniques.
  - With the alpha of 1e-10 since this is the value with the highest value and right before the data is trapped into overfitting.

Challenge:

As Christoph Molnar mentions in his book, called Interpretable Machine Learning[1], interpretability is a very important topic in machine learning and it should be considered in the design of a real-life project, so people can understand the connections between the features. In this case, however, since I implemented my design without considering this characteristic, I have chosen a model that is very difficult, if not impossible, to interpret the model. This is because I'm using a neural network in which according to an article by Eduardo Perez Denadai[2], its weights are a measure of how strong the connections are between each pair of neurons. This means that maybe only the first layer of the neurons can generate a useful which combining it with weights rom other connections makes it very complicated to understand the contribution of the variables.

This could bring consequences with it. For example, when the model is not clearly showing the features in which it predicts the genre based on, the genres with close characteristics might be predicted in place of each other and result in showing the wrong type of music to the user, which after a while will lose the user's trust. In addition, enhancing the power of the technique might take longer and we have to go through the whole process of testing and observing phase again.

A simpler model, such as kNN, in comparison with neural network models, provides a better and more useful analysis on this, with weights showing the contribution and effectiveness of the attributes.

[1] https://christophm.github.io/interpretable-ml-book/index.html
[2] https://towardsdatascience.com/interpretability-of-deep-learning-models-9f52e54d72ab