Assignment machine learning 2021-2022
Machineleertechnieken

(B-KUL-JLI209)

Gilles Peeters

December 9, 2021

Date:	December 4, 2021
Class:	mELICTgs
Instructor:	Tony Wauters

Contents

List of Listings

1 Introduction

For the course 'Machineleertechnieken' I was asked to find two data sets, execute two different learning tasks on these data sets, using (at least) two different models per data set. What follows is the report on my findings and results concerning this assignment.

2 Data set 1: Music genre Classification

For my first data set I chose a data set I found on Kaggle [1]. It contains several features that might be used to determine the music's genre. A column containing the correct genre is also present so we have target prediction data to use in our model later on. A prediction problem like this can typically be solved using either regression or classification models. Since I intend to separate all songs into 10 musical genres with no option for values in between genres, classification will be the learning task of choice here.

2.1 Preprocessing the data

Before we can get to work testing models we have to clean up our data by dropping irrelevant columns and transforming the non-numerical ones.
To process the non-numerical columns, we have to choose between ordinal encoding or one-hot encoding. We will use ordinal encoding for all columns because one-hot encoding seems to be too bulky in this case. For a more in-depth explanation see the python notebook on random tree classification (Music_genre_RTC.ipynb).

2.2 Random Forest Classifier

As a first attempt at solving this classification problem the random forest classifier seemed like a good model to use. This model is simple enough for me to understand and improve my understanding of machine learning as I delve deeper into the matter, and it should be a decent solution to any classification problem. Let's start by going through the process I followed to get to my results.

Note that we will be comparing performance of the model based on the 'model.score' method which gives us a goodness of fit measure for our model. Another option would have been 'accuracy score' which would score our model based on whether the set of labels predicted for a sample *exactly* matches the corresponding set of labels in y true.

2.2.1 Missing value handling

We still have to decide how to handle missing values. There appear to be 3 options: dropping columns, dropping rows or using an imputer. After comparing each method we can see that simply dropping all 'Nan' containing rows gives us the best results.
Having cleaned up our data we can now go ahead and use 'train test split' to split our data into 'train' and 'validation' data, for both the input (X) and output/prediction (y) data.

[1] https://www.kaggle.com/vicsuperman/prediction-of-music-genre

2.2.2 Feature picking

The next step to finding the best model is to determine the importance of every feature, maybe some make the model worse and removing them could improve our model. After running test with one or two features removed, the only improvement occurred when removing the ’energy’ column. We will therefore only remove this column and move on to the next step.

2.2.3 Model parameters

To further improve our model we can try changing some of the default parameters. To see how every parameter was determined see the jupyter notebook.
We could play around with all the different parameters of the random forest classifier model, but since we only have an accuracy score of around 0.6 at this point it is highly unlikely a truly better model will be found this way. In conclusion we can say that even though there were improvements, this model appears to be unable to accurately predict the genre of a song based on the given features. In hindsight this was to be expected since we are using a general purpose model on a task that is known to be very hard and specific. For our next model we will try and find a more advanced model.

2.2.4 Visualisation

Let’s visualize our predictions and their accuracy. This can be easily done using a confusion matrix:

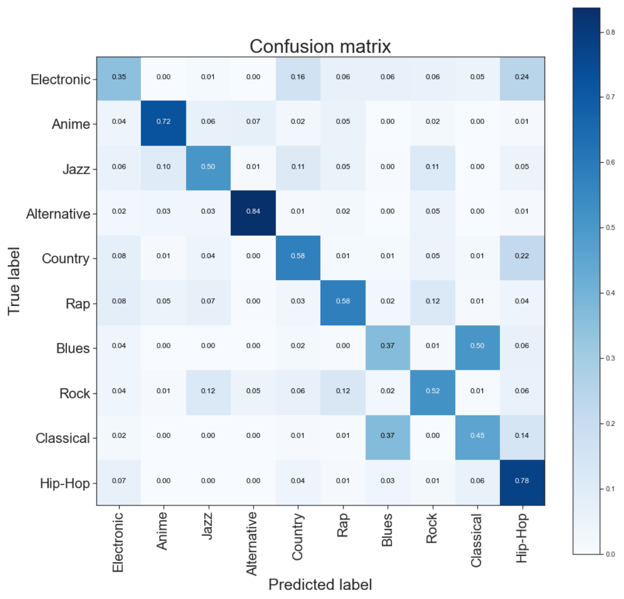

Figure 1: Confusion matrix showing our prediction accuracy

We see that our predictions are quite accurate for some genres, but quite inaccurate for other genres. We see electronic music gets confused with Hip-Hop or even Country music, Blues and Classical music are apparently similar as they often get mistaken for each other. Anime and Hip-Hop get the right prediction 70-80% of the time and Alternative music is predicted with an accuracy of 84% which is starting to look acceptable. The other genres get the right prediction about 50-60% of the time.
This kind of results is to be expected for a problem this complex. Alternative seems to be the best defined genre as it gets accurate predictions and other music genres are rarely mistaken for it. Hip-Hop on the other hand seems to be too similar to other genres, it's predictions are decent but other genres like Electronic and Country get mistaken for Hip-Hop over 20% of the time. A way to improve our model, while at the same time reducing it's usefulness, would be to remove the genres that cause the most confusion and the entries that correspond with them. By removing Hip-Hop, Blues and Classical we may get better results since they seem to cause most confusion.

2.3 Keras Classifier

Instead of just another model, for our second attempt we will use a classification neural network created in Keras. This will take longer to compute but should hopefully give us some better results. Unlike for the random forest classifier we do not have to manually deal with NaN values, the neural network will decide for itself how to handle these.
As to keep this report short enough we will not talk about the steps taken to assemble our neural network. We once more tried changing several parameters and picked the ones that performed best to decide on our final model. For a more in-depth approach see the python notebook on Keras (Music_genre_Keras.ipynb).

To compare this model to the one we got before we will once again generate a confusion matrix based on our predicted data:

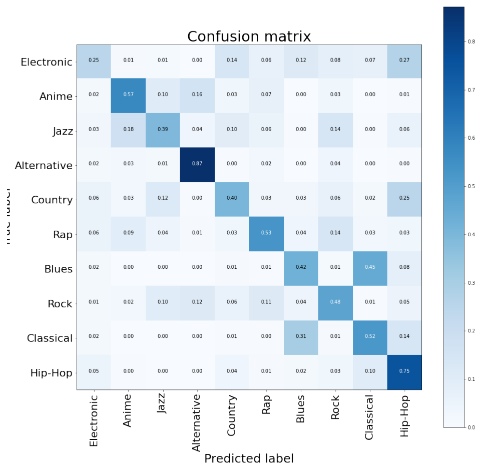

Figure 2: Confusion matrix showing our prediction accuracy

We can see this neural network gives us predictions that are very similar to the ones from the random forest classifier. Same as before applies here, having all genres in the mix seems to make this problem too hard to solve with the models used here, somehow the confusion would have to be improved to improve this model.

3 Data set 2: Twitter mining

We generated our second data set using a script to mine tweets and their data. For our machine learning task we went with logistic regression, using regression models to determine the sentiment of a tweet. Once again, for a more in depth or step by step approach see the python notebook. (sentiment_analysis_regression.ipynb)

3.1 Preprocessing the data

We will be using the 'tweet_text' to determine a tweet's sentiment. In order to be able to train a model for sentiment analysis we have to first generate the sentiment for some of our tweets using an existing library. To do this we first go ahead and clean the tweet text for every tweet. In this case we decided on removing punctuation, https links and hashtags to prevent these from confusing our model.
Next we used the 'Textblob' library to get the 'polarity' for every one of our tweets. This polarity can be seen as a value between -1 and 1 that corresponds with a sentiment between negative and positive. We remove all tweets with a 0 polarity as they are neutral and therefore likely to confuse our models.
A final step we have to take before we can initialize and train our models, is making our tweet text into word vectors. We do this using the 'spacy' library creating a vector for every tweet. This gives us an array of vectors we can work with.

3.2 Models

For our first model we went with Epsilon-Support Vector Regression (or in short: SVR). We will use this model's default parameters to keep this report short enough. Our second model is the RANSAC regressor, which we will also use as is without changing any parameters. We will be using both models to predict the polarity of a tweet based on it's text.

3.3 Comparison

Since we are comparing two regression models we will take a look at 2 visually comparable metrics: the prediction error and the residuals. The prediction error is a measure of how well the model predicts the response variable, it shows us how far off the predictions are visually. The residuals are the difference between the predictions and the response variable, shown in relation to the features the prediction is based on.
Do note that to evaluate our models we used a testing set of tweets that are not related to the data sets used to train our models, to ensure the models are not too specific. Let's have a look at our results:

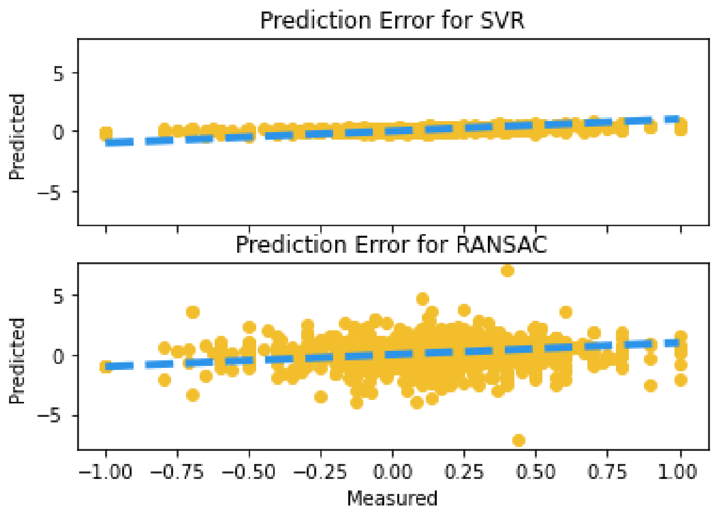

Figure 3: Blue line corresponds with the target values, yellow dots are the predictions

We can see that for SVR the predictions are very close to the target values. For RANSAC we can see the predictions are somewhat clustered around the target, but not nearly as close as for SVR. We even see values above 1 or below -1 which should never occur. For now SVR seems to be a clear winner.

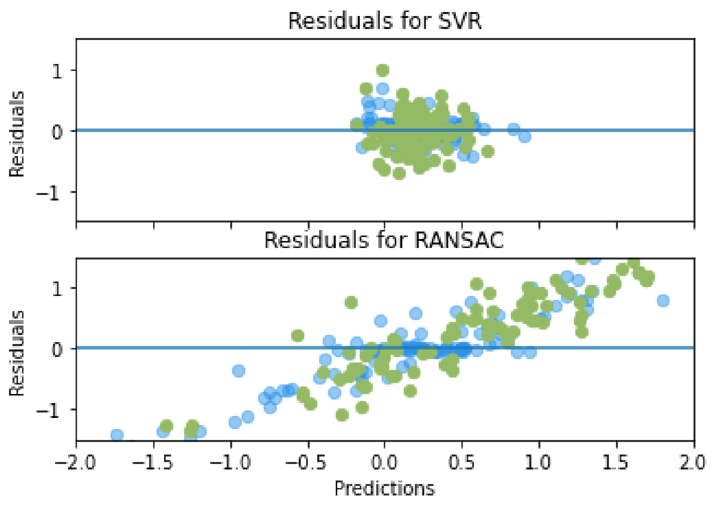

Figure 4: Plotting residuals using training (blue) and test (green) data

Once more we can see that RANSAC gives us many predictions that are out of bounds. The greater the predicted value, the greater the residual. For SVR we see a nice cluster having no residuals above 1 or below -1, indicating this model does a pretty good job at determining a tweet's polarity using the tweet's text.