# Music genre prediction

**Banjac Luka sw-78/2016**
**Metikoš Daniela sw-21/2017**

## 1. Motivation

Both of us are fans of music and we wanted to make software which is going to predict song's genre based on different parameters. The idea is to enable better recommendation for music playlist.

## 2. Research questions

In this project we used a couple of different classifiers, compared them and used the one with the best results. Dataset [1] was found on the Internet. It contains few csv files with more than 160.000 songs collected from Spotify Web API. Data is grouped by artist, genre and year. Every song is described with digital signatures such as:

- acousticness (ranges from 0 to 1)
- danceability (ranges from 0 to 1)
- energy (ranges from 0 to 1)
- duration_ms (integer typically ranging from 200k to 300k)
- instrumentalness (ranges from 0 to 1)
- valence (ranges from 0 to 1)
- popularity (ranges from 0 to 100)
- tempo (float typically ranging from 50 to 150)
- liveness (ranges from 0 to 1)
- loudness (float typically ranging from -60 to 0)
- speechiness (ranges from 0 to 1)
- year (ranges from 1921 to 2020)

## 3. Related work

We have found a lot of projects on this theme. For y it used *Logistic Regression*, classifiers as *RandomForest*, *AdaBoost* and *DecisionTree*, *NaiveBayes* [4], *Convolution Neural Network* (on the spectrogram images, [2]) and they measured *accuracy*. The results were in range from 70 to 80% and they were gained with Logistic Regression.

## 4. Methodology

For data processing is used *MinMaxScaler* [5] to normalize data. This estimator scales and translates each feature in the range from 0 to 1, which is helpful for our dataset as most of our categories are in range from 0 do 1.

Data classification was put through *RandomForestClassifier* [6], *AdaBoostClassifier* [8], *DecisionTreeClassifier* [9] and *KNeighborsClassifier* [10].

For determination of hyperparameters we used *cross validation* from related projects and applied it on our model.

Inequality between classes were balanced with *SMOTE* [12] and *ADASYN* [13].

## 5. Discussion

First of all, we found super genres by going through genres for every artist and selecting one which appears the most. Further, for every song we assigned super genre from the given list. Considering that our dataset has normal distribution, between 0 and 1, we used z-score to remove outliers. So, we got a diffusion of samples in every category and calculated how far the data sample is from mean.

We split the data set to train and test in a ratio of 70 and 30 [6]. On the sets we applied different algorithms with given hyperparameters:

1. SVC
2. GradientBoostingClassifier
3. RandomForestClassifier
4. AdaBoostClassifier
5. KNeighborsClassifier

Out of all algorithms RandomForest with parameters n_estimators=200, criterion='gini', max-features='auto' and max_depth=100 gave the accuracy of 48%, and after applying MinMax() and removing outliers, we got accuracy of 54%.

Considering imbalanced classes for genres in our data we tried to increase accuracy. In order to provide balance, we oversampled the minority classes in the train dataset. We used *SMOTE* and performed, again with oversampled data and got much better results. We tried *ADASYN* too, but *SMOTE* gave slightly better results.

Final result of RandomForest classifier with parameters n_estimators=80, criterion='gini', max-features='auto' and max_depth=40 we got the **accuracy of 85%.**

## 6. References

1. https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks?select=data_by_genres.csv&fbclid=IwAR0HSsOwik5aM3_6Js6Lc1cZvpjMmce3azH5zEz0pUCkPMf0tKSVyDqQn8w
2. https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8
3. https://towardsdatascience.com/how-i-built-an-audio-based-music-genre-predictor-using-python-and-the-spotify-web-api-a957e427a515
4. https://towardsdatascience.com/music-genre-analysis-understanding-emotions-and-topics-in-different-genres-110377d8888c
5. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html?highlight=train%20test%20split#sklearn.model_selection.train_test_split
7. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random%20forest#sklearn.ensemble.RandomForestClassifier
8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html?highlight=adaboost#sklearn.ensemble.AdaBoostClassifier
9. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html?highlight=decisiontreeclassifier#sklearn.tree.DecisionTreeClassifier
10. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html?highlight=kneighborsclassifier#sklearn.neighbors.KNeighborsClassifier
11. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html?highlight=accuracy#sklearn.metrics.accuracy_score
12. https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html?fbclid=IwAR3iiXfPAdozfjhF75xb_xNlvrTjCZmCictXp_qcQ-m4sDcEx5OiXjuvyew
13. https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.ADASYN.html?fbclid=IwAR0oX6ery_OaihnzC7EjaSqJXDrUA1aDIvKxiY1A2HJnKQk03qZZ64s7lT4#imblearn.over_sampling.ADASYN