FYS-2021: Machine Learning.
Assignment 1

Artemii Gribkov

1 Introduction

For our first assignment in Machine Learning we will work with the spotify-dataset, where a lot of different songs are presented. They have different genres and every features includes different numbers. In this assignment we are focusing on the two genres- Pop and Classical. Our goal is to create a model that will classify our data by its features To do it, we will use our knowledge from the lectures and previous experience from studying, for examples experience with programming in Python-language. GitHub repository can be found in the reference part.

2 Technical Background

In this assignment I will use several concepts that will help me to solve given in assignment problems:

- Logistic regression - technique in statistics used to classify data into one of two categories by estimating the probability of each category.
- Sigmoid function - function that is used in logistic regression and outputs value between 0 and 1 (convert predictions into probabilities)

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Stochatic gradient descent - optimization method that using gradients from data points or small batches, and helping with training
- Accuracy - proportion of how many samples been correctly classified out of the total number of them
- Confusion matrix - table that giving insight into the performance of a classification mode.
- There are useful formulas in terms of True Positive, True Negative, False Positive and False Negative that gives better understanding with the confusion matrix.

Metric	**Formula**
Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP+FN}}$
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN+FP}}$
Precision	$\frac{\mathrm{TP}}{\mathrm{TP+FP}}$
Accuracy	$\frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$

Tabell 1: Evaluation Metrics

3 Design

In this assignment we need to use Stochastic Gradient Descent model for classification of our dataset. Also it is important that sklearnpackage is not fully permitted because it will oversimplify the work and will ruin the main goal - understand how classification work and how it can be modelled.

For the first par of the assignment I am planning to use my previous experience with the Python, since this

part requires only to work with the raw data. Reading and presenting of the data can be done with pandasfunction and plotting can be done with "matplotlib". For the training and testing arrays I will present two methods that can be used. First one is manual by separating Pop and Classical music from other genres and take liveness and loudness features. Second method is by using function from sklearnthat is allowed to use for this assignment. The important part for creation of the sets is to use 80 prosent of all samples.

For the second part of the assignment I plan to create three functions - Sigmoid function which is part of linear regression, prediction function to predict whenever it is Pop or Classiscal music, and the SGD function which is our model for classification.

For the third and last part of the assignment there are two possible was to solve the problem. We can use manual method, namely formulas that we learned from the lectures or again sklearn". Here I will use only manual method, since it gives better understanding since we see what is happening. Also I plan to show the Confusion matrix as a plot, so I can work with it.

4 Implementation

a) Problem 1

As mentioned in Design part and problem explanation, I use pandas to read the csv file. Unfortunately, the spotify-dataset is huge and therefore we cannot see if the programm print all the data - samples and its features. Therefore I print how many samples and features do we have in our dataset. The result is:

There are 232725 samples in the dataset.
There are 18 features in the dataset.

In the sub-problem 1b, I filter out Pop and Classical music separately because it is easier to me to work with the separately. The same procedure I did with the labeling and extracting two features from the dataset. Afterwards I combined variables that are for Pop music and Classical music together. This method doesn't look perfect, since I am breaking the order in which songs are listed in original dataset, but it helped me to follow the procedure of splitting and I got this result:

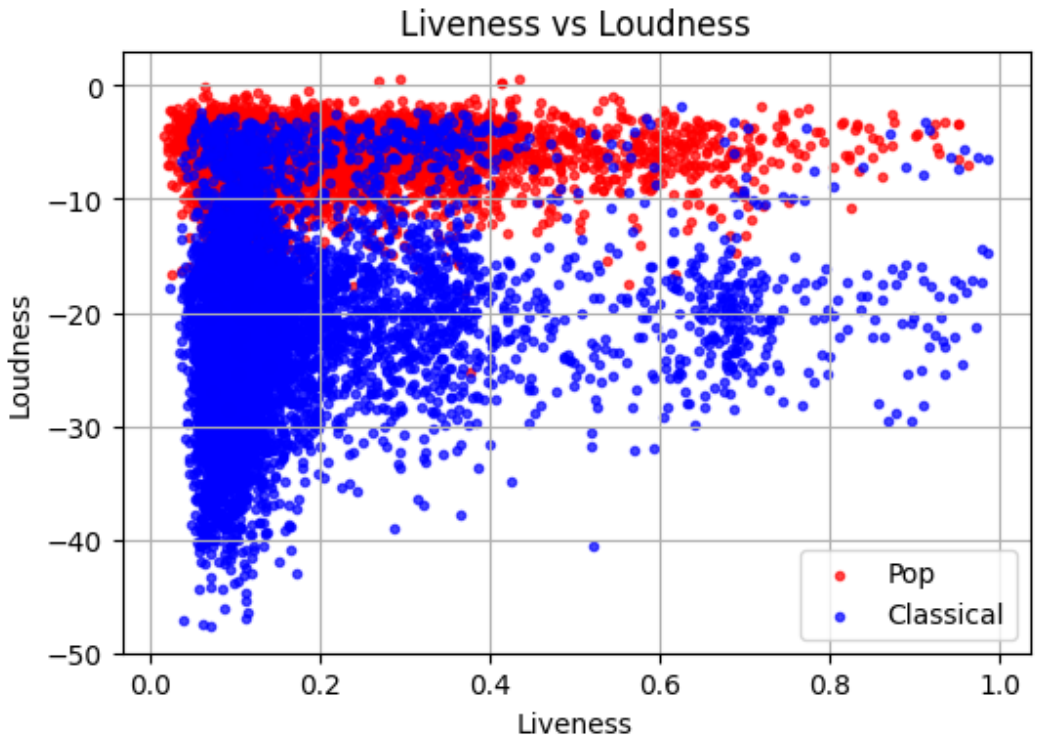
There are 9386 Pop music songs.
There are 9256 Classical music songs.

In sub-problem 1c, I introduce new variable that represents the ratio of samples we want to use for training. I multiply this ration with total number of samples and get desired number, which I name as training samples. For better training and testing, I got advice to shuffle the data. Firstly, I create reproducibility, so we will get the same output every time the indices are randomised. Secondly, I randomise indices for our samples. My next step is to introduce 4 variables, where two of them are for training and two of the are for testing. To check if these two arrays contain approximate (I didn't calculate it by hand) correct amount of samples, I print the result how many samples in each array. X represents features and y represents labels.

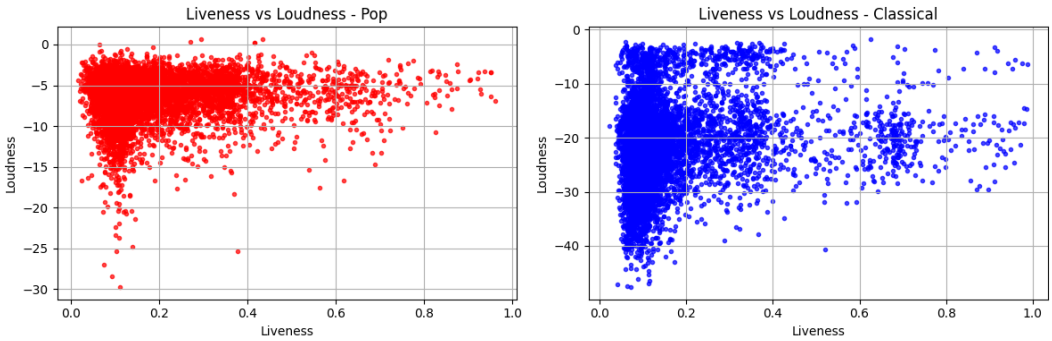
Training set has 14913 samples. Test set has 3729 samples.

For the interest and to compare methods, I used function from sklearn", and gladly got the same result. Also it took only one code line instead of several.

For the bonus sub-problem 1d, I used new training variables. Also I decided to create three plots. One for both Pop and Classical, and one for each separately.



Figur 1: Pop and Classical



Figur 2: Separate Pop and Classical

From what I see, it will be uneasy task for us and classification. As we can see a lot of Pop samples are overlapped with Classical samples. Basically we can say that data is mostly on the left side of the plot. Also there are some samples that are a bit far from gathering of bigger group of samples. It is still possible to do, since Pop samples are mostly located on top, while Classical on the middle, but the regression line will be far from being perfect in my opinion. To solve this problem, we might need a bit more complex solution. More features for example.

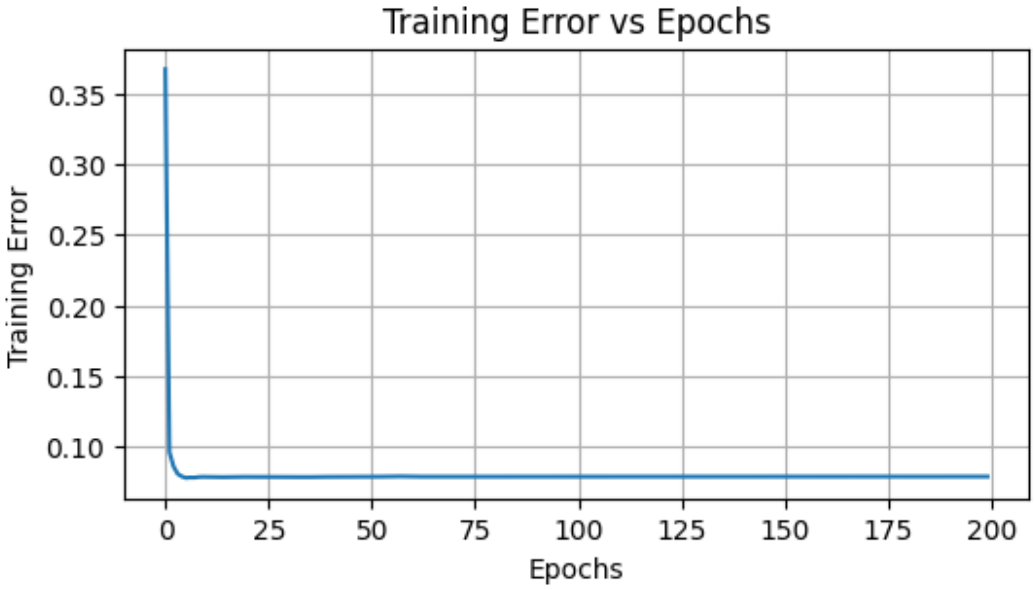
b) Problem 2

This part was the hardest and most important part of the assignment. It contains the model with its needed functions. I start with the easiest to understand and create - Sigmoid function to convert data to binary format (probabilities). Input of it is the weighted sum of inputs

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The second fucntion is prediction, where I use the Sigmoid function as was mentioned to convert prediction. It takes two imputs - data (x) and weights. The important part of this function is what it returns. There I use binary classification. So we get for all number higher thatn 0,5 to be True, otherwise False. Stochastic gradient descent function takes four inputs - data (x), labels (y), number epochs and training rate. With this function I experienced the most troubles in the assignment. First of all it is how to make the function. After each attempt I had a feeling that I forget something. Nevertheless, I got some help and inspiration from other sources, so the function is working but most likely not ideally. Since our plot should include training errors, outputs of the function are optimized weight of data and training errors that we be noted inside the function. First part of function we can say is preparation. I create an empty array for errors and initialize taken weights randomly. Afterwards for every epoch and number of samples I calculate errors and use prediction function - get prediction probability by Sigmoid, calculate error, adjust weight and lastly predict outcomes for the entire dataset. After I am done with functions, I introduce two variables that we need - training rate and epochs. Training rate determines the size of the steps taken during the weight update process and epochs which are the number of times the entire training dataset is processed through the model during training. At this point I encountered new problem. For better training we need shuffling. There are two possible places where we can do it - inside the function and outside the function. When I used shuffling of data inside, I got less accuracy than if I do it outside. Unfortunately I don't have good explanation for it. I would say, from side it requires more studying and observation. Nevertheless, with some advice, I stand by the outside shuffling. With the training rate 0.0001 and epochs 200 I got these results:

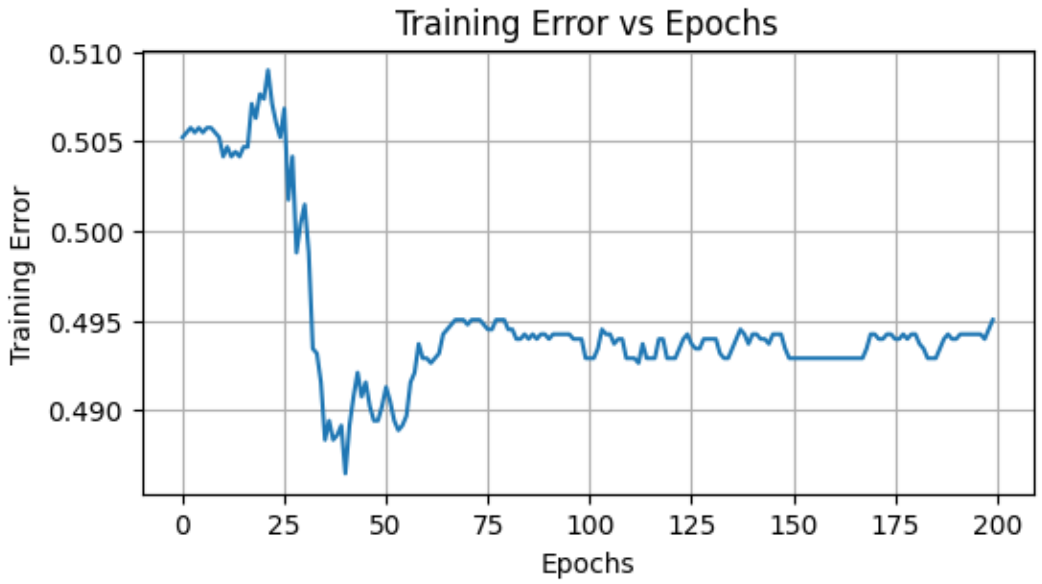
Training Accuracy: 92.15
Test Accuracy: 91.74



Figur 3: SGD

And this is what I got with shuffling inside:

Training Accuracy: 70.80
Test Accuracy: 72.35



Figur 4: SGD2

After I choice how I want to use shuffling, I start to play with epochs and training rate numbers. With constant training rate, decrease in epochs - increase the training accuracy and decrease in testing accuracy. With constant epochs, I observed changes only with big increase of training rate, which leads to decrease in training and testing accuracy.

In the bonus sub-problem 2c, I was not able to do regression line properly, because line was or lower than all samples or higher. I got some advice that I might use scale.

At this state I would like to to pay more attention to it. Because I used scale of data also for SGD but didn't mention. It is mostly because I start to experience doubts and problems later. Without scaling of data my accuracy was around 50-60 which is very small. By introducing the scaling function and implementing it later in code, increased the accuracy and solved (I don't fully understand why) the regression line. One more positive thing of scaling is that it solves the problem of significant difference in scaling between liveness and loudness. For regression line I used following equation (desicion boundary equation):

$$w_1 \cdot x_1 + w_2 \cdot x_2 + b = 0$$

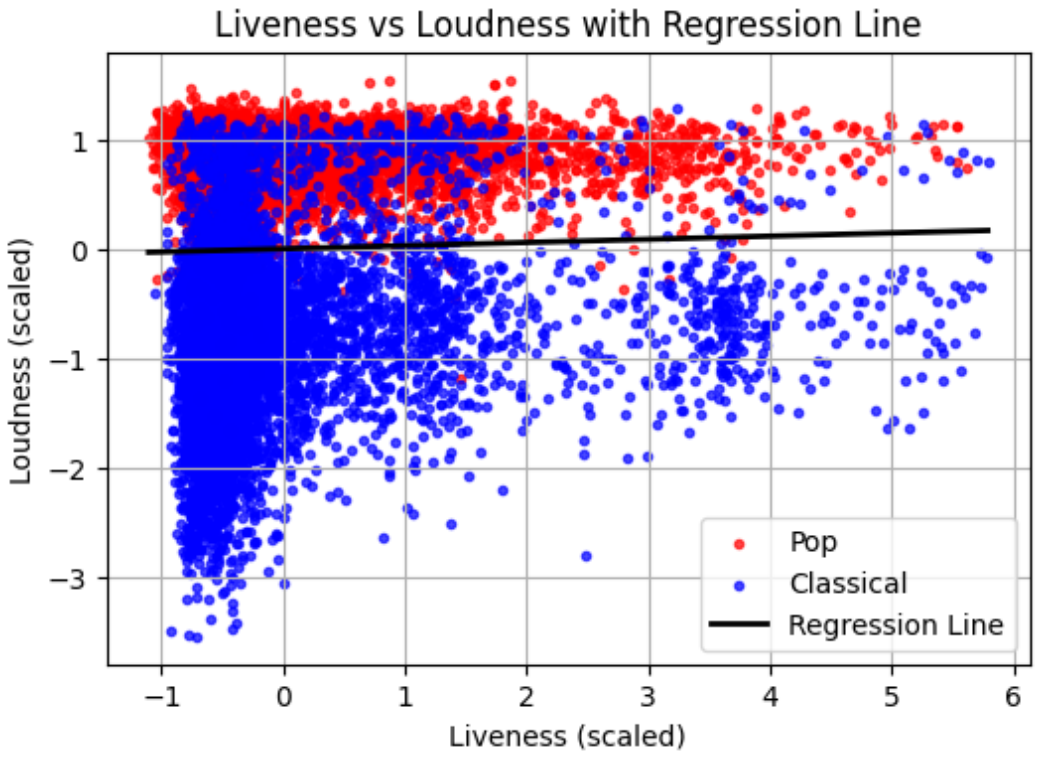
Take that the bias term b is zero, the equation simplifies to:

$$w_1 \cdot x_1 + w_2 \cdot x_2 = 0$$

Solving for x_2 in terms of x_1:

$$x_2 = -\frac{w_1 \cdot x_1}{w_2}$$

By using this formula, where w1 is liveness, w2 is loudness, I got this plot:



Figur 5: Regression line

As I mentioned previously, it will be hard to classify these two genres, but the result is surprisingly better than expected. As was predicted it is separating them between the middle and upper parts of the plot. So, I would say that classification will not be easy, but have high chance to be done quite good.

c) Problem 3

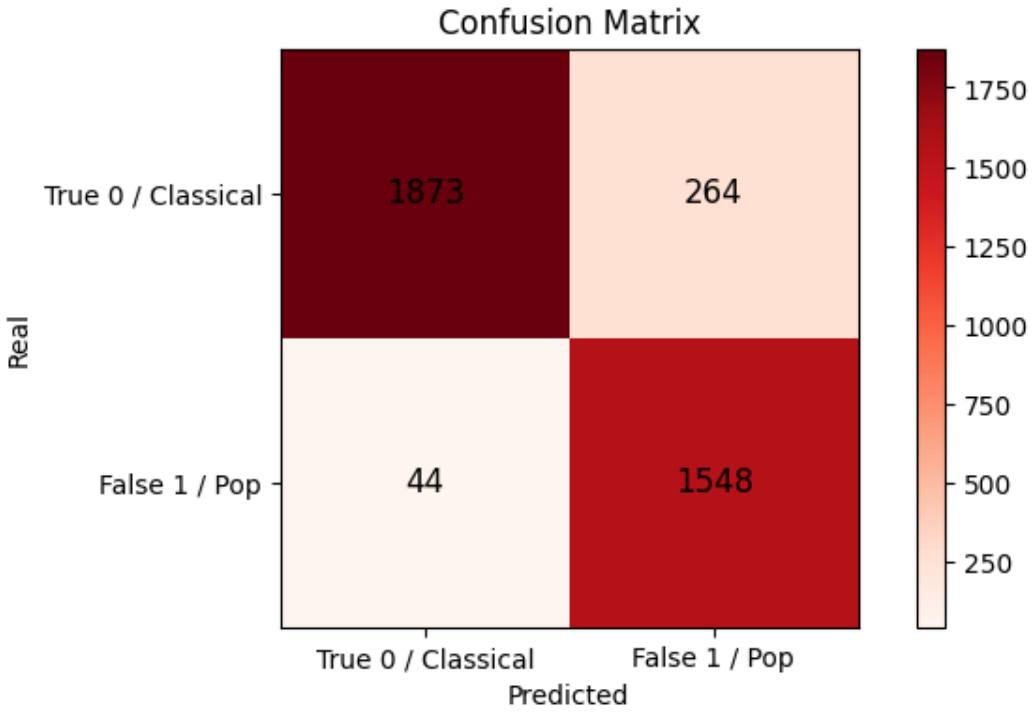
In the third part of the assignment, I decided to create confusion matrix manually. Most likely with sklearn it is faster but again understanding will be better when you do it fully. I used formula. First thing that I did, was to create a function to calculate every TP, TN, FP, FN and make an array of them how they supposed to look in the matrix ([[TP, FP], [FN, TN]]). Afterwards I created the matrix (four squares with showing True and False) and then put the numbers inside. AS for report of confusion matrix, I printed out TP, TN, FP, FN:

The confusion matrix is given by:

$$\begin{bmatrix} 1873 & 264 \\ 44 & 1548 \end{bmatrix}$$

Where:

- **True Positives (TP)**: 1873
- **False Positives (FP)**: 264
- **False Negatives (FN)**: 44
- **True Negatives (TN)**: 1548



Figur 6: Confusion Matrix

Confusion matrix provides us the better understanding of classification. We can see how many samples from each class were misclassified as another class. Also as was mentioned in the Technical Background section, we have formulas to calculate Precision, Specificity, Sensitivity which can provide us more information of classification and help with the further understanding and observation. Accuracy tells us only the overall proportion of correct classifications, Therefore it vulnerable for Class imbalance and lacking details about errors.

Unfortunately, I was not able to come up with the proper idea for the bonus sub-problem 3c. I am thinking that it one of the possible solution to it is the samples from Classical songs that shares values for features like loudness with Pop songs. The second option might be one of the problems of accuracy that matrix shows. Classical songs that have been classified as Pop songs can be a good proportion for Pop song lovers.

5 Discussion

From this assignment, I want to point out some problems and possible improvements I found out later or in process.

Works pretty good:

- I would say that the whole Problem 1 overall works fine.
- Sigmoid function works pretty well
- Confusion matrix looks fine and correct.

Have error or minor errors:

- I would like to change is to separate and classify samples together (Pop and Classical) rather than separately
- Optimization of the code can be better
- I should add some more explanation line to the code and steps

Require improvements or changes:

- SGD function requires some changes or improvements in general
- Scaling function requires more studying from me, it might be unnecessary function that makes code more confusing
- After finishing the code and lacking time I found out that I most likely missing the entropy function
- Should give some attention to bias (relevant for 2c)
- Some parts require a better/proper explanation and better structure
- Report itself can be structured better (speaking about the answers and procedures)

6 Conclusion

It was an interesting experience to work with this assignment. I struggled a lot with Problem 2 and I have a good feeling that I made plenty of mistakes there because of lacking some experience and understanding. Throughout the whole process of solving this assignment I got good advice from fellow students and found some inspiration of ways how to solve some problems. Also this assignment helped me with understanding how GitHub works, which is in my opinion good.

7 Reference

Gribkov, Artemii (2024). *FYS-2021 Machine Learning. Assignment 1.* Github repository. URL (link to the repository).
Ricaud, Benjamin (2024). *FYS-2024 Machine Learning. Lecture Notes.*
Størdal, Magnus Oterhals (2024). *FYS-2024 Machine Learning.Solution for Exercise Set 2*