FYS-2021: Machine Learning

Assignment 1

Sera Madeleine Elstad

September 8, 2024

`GitHub Repository for assignment 1`

Contents

1 Introduction

This report describes the completion of the first mandatory assignment for the Machine Learning course (FYS-2021). The assignment required classifying music tracks into "Pop" and "Classical" genres using a Spotify dataset that among other things includes genre, artist, liveliness, and loudness. The assignment is organized into three primary parts: data preprocessing, machine learning, and results. In the reports implementation part the solution of each problem, together with its subproblems, is explained.

2 Methods

2.1 Logistic Regression

Logistic Regression is used for binary classification tasks where the goal is to predict discrete, binary outcomes. In this project, it is used to identify songs as 'Pop' or 'Classical'. Logistic Regression predicts the likelihood of the default class (e.g., 'Pop') using the logistic function, also known as the sigmoid function.

2.1.1 Logistic Function Sigmoid Function

The sigmoid function ($\sigma(x)$) maps any real-valued input to the interval (0, 1), which is ideal for binary classification as it interprets the output as probabilities [1]:

$$F(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

2.1.2 Decision Boundary

Logistic regression divides the data into two classes using a decision boundary. This barrier is set at the point where the chance of belonging to a positive class equals 0.5. Beyond this point, instances are classified into the positive class; otherwise, they are classified into the negative class [2].

2.2 Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an optimization algorithm used to minimize the objective function, represented as a sum of individual losses associated with each dataset point. This method updates model parameters w iteratively using:

$$Q(w) = \frac{1}{n} \sum_{i=1}^{n} Q_i(w) \tag{2}$$

where ($Q(w)$) is the objective function, (n) is the number of data points, and ($Q_i(w)$) is the loss at the (i)-th point. SGD modifies (w) by applying the update:

$$w := w - \eta \nabla Q_i(w) \tag{3}$$

where, $\nabla Q_i(w)$ is the gradient of the loss at a randomly selected data point, and η is the learning rate [3].

2.3 Cross-Entropy Loss Function

The Cross-Entropy Loss Function, or log loss, evaluates the accuracy of a classification model. It compares the actual labels of data points to the model's predictions, with lower values indicating better model performance. The cross-entropy loss is calculated using this formula:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{4}$$

where N is the number of data points, y_i is the actual label , and $\hat{y}_i$ is the predicted probability for each data point[4].

2.4 Confusion Matrix

A confusion matrix is a tool that summarizes the performance of a classification model by showing the counts of true positive, true negative, false positive, and false negative predictions [5]. Figure 1 shows a generall example of who a 2 times 2 confusion matrix look.

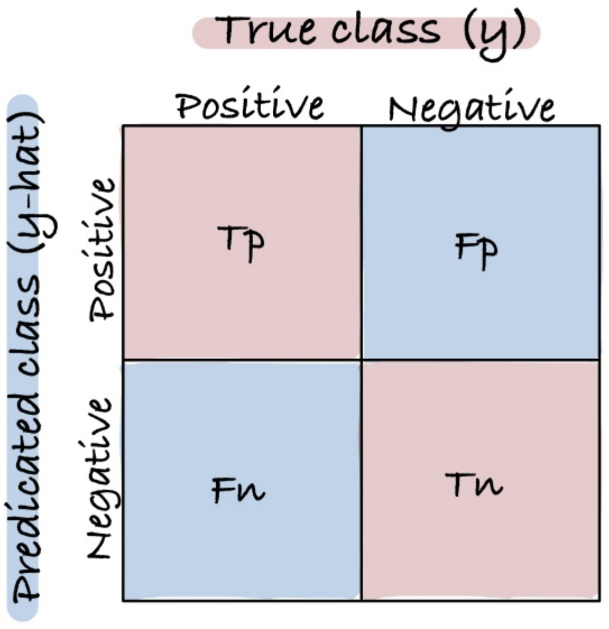


Figure 1: A general 2x2 confusion matrix illustrating the classification outcomes.

3 Problem solution

3.1 Problem 1

For the first task, the goal is to classify songs into either the ”Pop” or ”Classical” genres using the provided Spotify dataset. To achieve this, the raw data must be processed, with relevant samples selected and labeled according to their genre classification.

3.1.1 Problem 1a)

Data loading is managed by the *DataProcessing* class. The class takes a filepath as an argument, initializing the data attribute to None. It employs the `load_data()` method to read the CSV file, using the pandas `read_csv` function, and reports the number of samples and features:

- Number of samples (excluding header): 232724
- Number of features (columns): 18

The try-except blocks will capture and log any load errors.

3.1.2 Problem 1b)

In this step, the Spotify dataset is filtered to focus exclusively on the genres "Pop" and "Classical." The filtering and labeling process is managed by the *process_data* function within the *DataProcessing* class. This function chooses only entries that belong to these genres and applies the following labled labels: "Pop" as 1 and "Classical" as 0. After filtering, the dataset distribution is as follows.

- Tracks labeled as "Pop": 9386
- Tracks labeled as "Classical": 9256

The *filter_features* function is used for further organizing the dataset, isolating only the most relevant features for our research, especially "liveness," "loudness," and "genre."

3.1.3 Problem 1c)

To prepare the data for the machine learning model, the *split_data* function, within the *DataProcessing* class. This fucntion uses the `train_test_split` function from sklern to divide the dataset into training and testing sets. The data is shuffled to ensure randomness and split in an 80-20 ratio. The resulting dataset sizes are:

- Training set size: 14913
- Testing set size: 3729

3.1.4 Problem 1d) Bonus

Figure 2 shows a scatter plot visualizing the relationship between "liveness" and "loudness" for the "Pop" and "Classical" genres, with each genre represented by a different color. As shown in the figure, there seems to be some overlap between the two genres, notably in the lower "liveness" range (0 to 0.4), where the majority of the data points are located. Despite the overlap, one can observe that the "Classical" tracks cluster more densely at lower "loudness" values.

Despite the overlap, the general scattering of "Pop" records along loudness implies that focusing on this attribute may still allow for a good classification performance. While the overlap might prevent perfect accuracy, the gap in loudness suggests that a classification model could perform reasonably well in discriminating between the two genres.

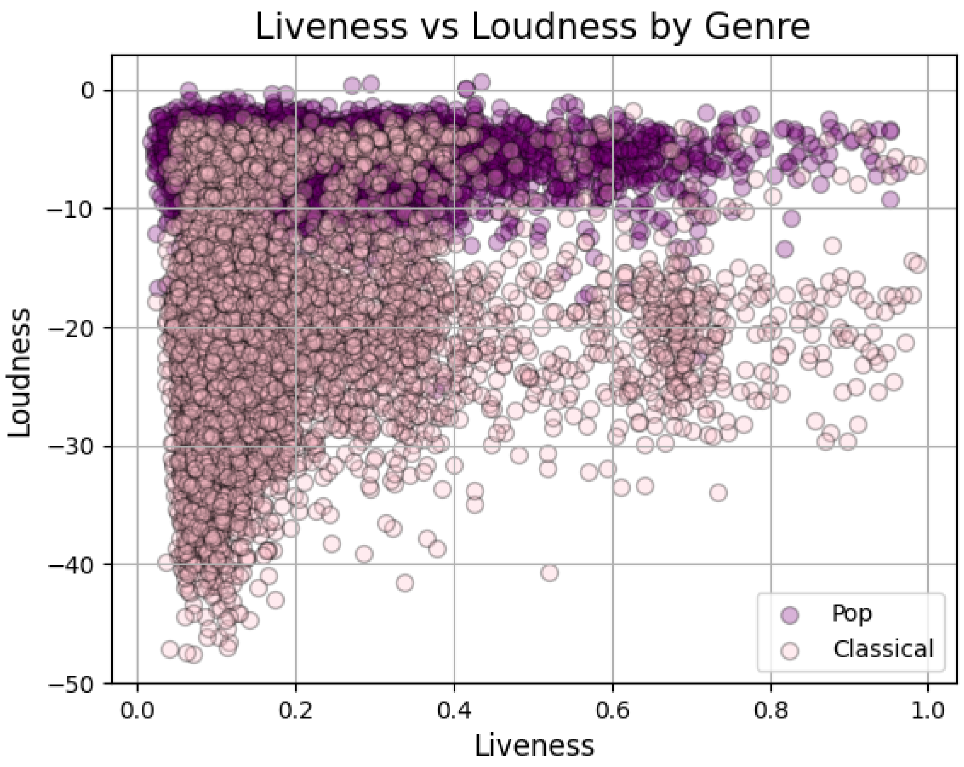


Figure 2: Liveness vs Loudness

3.2 Problem 2

The goal of this task is to develop and evaluate a logistic regression model, also known as a logistic discrimination classifier, to distinguish between "Pop" and "Classical" genres based on the parameters "liveness" and "loudness".

A class called *Perceptron* is created for this purpose. This class initializes the weights, bias, and learning rate in preparation for the Stochastic Gradient Descent (SGD) algorithm to optimize these parameters.

Problem 2a)

The first part of Problem 2 involved creating our own logistic classifier and use the data from Problem 1 to train it. This classifier uses the Stochastic Gradient Descent (SGD) to learn to classify songs as either "Pop" or "Classical" based on features: "liveness" and "loudness." SGD updates the model little by little with random parts of the data, which might make the learning process look a bit uneven as shown in the graph below 3.

The classifier uses a sigmoid function 1 to turn the input features ("liveness" and "loudness") into probabilities. These probabilities show how confident the model is that a song belongs to either "Pop" or "Classical." A probability close to 1 means the model is confident the song is "Pop," while a probability close to 0 means the model thinks it's "Classical."

These probabilities are then compared to a threshold, usually 0.5. If the probability is higher than 0.5, the song is classified as "Pop"; if it's lower, the song is classified as "Classical." The model calculates these probabilities during its forward pass and uses the cross-entropy loss function (4) to measure the error between the predicted probabilities and the actual labels.

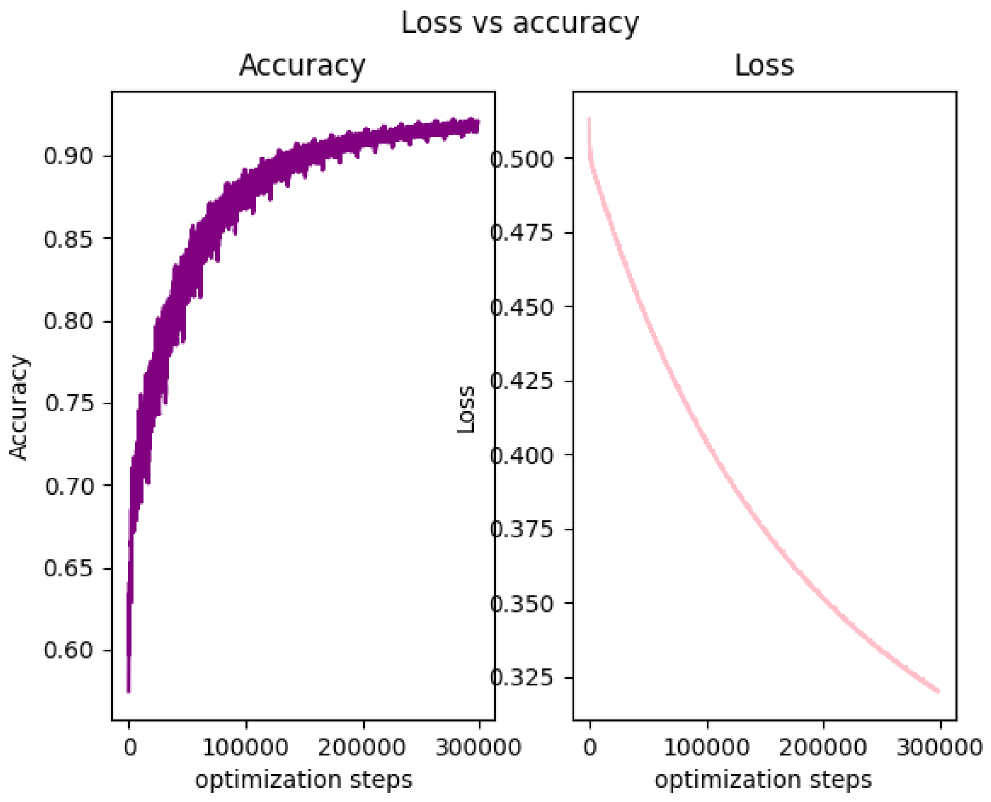


Figure 3: Loss vs accuracy for the training set, 50 epochs

3.2.1 Problem 2b)

After training, the logistic regression classifier was evaluated using a test set that had not been used during the training phase. This approach helps determine how well the classifier performs on new, unseen data. Both the training and test sets include the same attributes—liveness and loudness.

The accuracy of the classifier was assessed by comparing the predicted labels with the actual labels from the test set. The results, that also are displayed in figure 3 showed:

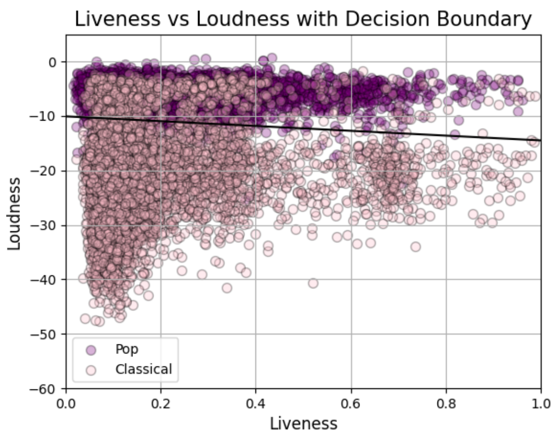
- Training set accuracy: approximately 92% (0.9207)
- Test set accuracy: approximately 91% (0.9150)

These results were achieved after running the model for 50 epochs, involving about 300,000 optimization steps. This high number of steps is due to each of the 14,913 samples being processed repeatedly across all epochs, which enhances the model's ability to learn from the data.

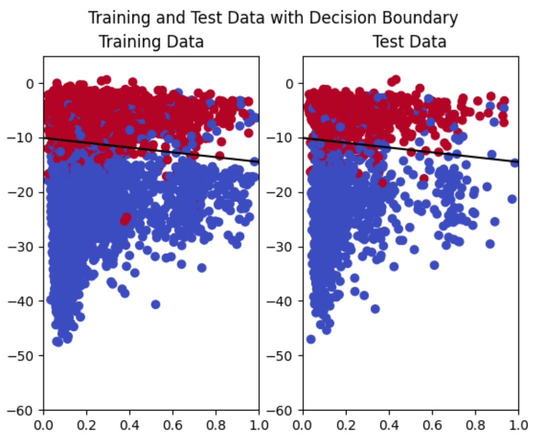
3.2.2 Problem 2c) Bonus

As a bonus task the trained model's parameters (weights and bias) from the logistic regression model was used to plot the decision boundary on the scatter plot of "liveness" vs. "loudness" that was created in Problem 1, see . Figure 4b shows a scatter plot for the test and training data separately with the decision boundary.

As a bonus task the trained model's parameters were used to visualize the decision boundary between "Pop" and "Classical" genres. This is depicted in the scatter plots below, see figure 4a, which show how well the model can separate the genres based on ”liveness” and ”loudness”. Figure 4 illustrate the decision boundary for combined and split sets of training and testing data:



(a) Combined Sets



(b) Split Sets

Figure 4: Comparison of Combined and Split Sets with Decision Boundaries

3.3 Problem 3

The goal of this problem is to evaluate the reliability of the classifier developed in previous problems, ensuring the results can be trusted.

3.3.1 Problem 3a)

For this problem, classification results from the test set used in Problem 2 were analyzed using a confusion matrix. The matrix helps visualize the classifier’s performance, distinguishing between correct classifications and types of errors made. The results of the confusion matrix are illustrated in Figure 5.

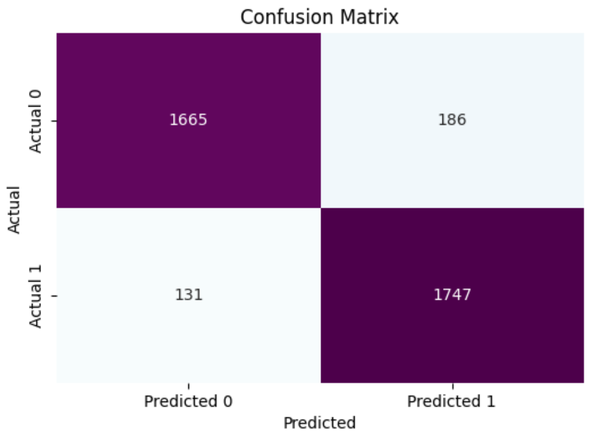


Figure 5: confusion matrix

3.3.2 Problem 3b)

The confusion matrix provides a detailed view of the classifier's performance, by revealing exact counts of true positives, false positives, true negatives, and false negatives. This breakdown is more informative than a accuracy percentage because it shows precisely how well the classifier is identifying each class.

From the matrix: - True Positives (Classical correctly identified): 1747 - True Negatives (Pop correctly rejected): 1665 - False Positives (Pop incorrectly identified as Classical): 186 - False Negatives (Classical incorrectly rejected): 131

These results allow the calculation of precision and recall:

- **Precision**: The precision of the classifier for predicting Pop music is calculated as the ratio of correctly predicted Pop songs to the total predicted as Pop. It is given by:

$$Precision = \frac{1747}{1747 + 186} \approx 90.39\%$$

- **Recall**: The recall for Classical music shows how many of the actual Classical songs were correctly identified. It is calculated as:

$$Recall = \frac{1747}{1747 + 131} \approx 93.03\%$$

4 Conclusion

This report covers the completion of the first assignment for the Machine Learning course, focusing on the implementation of a logistic regression classifier to differentiate between "Pop" and "Classical" music genres. With the developed code, one can upload a data file and categorize songs into "Pop" or "Classical" based on liveness and loudness features. The classifier demonstrated robust performance with an accuracy of 91%, and the confusion matrix further illustrated strong precision at 90% and recall at 93%, confirming the classifier's effectiveness.

References

[1] GeeksforGeeks. Derivative of the Sigmoid Function. *GeeksforGeeks*, June 2024.

[2] GeeksforGeeks. Decision Boundary of Label Propagation Vs SVM on the Iris Dataset. *GeeksforGeeks*, December 2023.

[3] Contributors to Wikimedia projects. Stochastic gradient descent - Wikipedia, September 2024. [Online; accessed 8. Sep. 2024].

[4] GeeksforGeeks. What Is CrossEntropy Loss Function? *GeeksforGeeks*, January 2024.

[5] GeeksforGeeks. Understanding the Confusion Matrix in Machine Learning. *GeeksforGeeks*, July 2024.