Github link: https://github.com/KatrineWik/Assignment1/tree/main

Assignment 1

Task 1a)

Using SpotifyFeatures.csv file and report the number of samples (songs) as well as the number of features (song properties) in the dataset. I will solve this by using the hint using panda and the read_csv function.

The dataset used for this assignment is the "SpotifyFeatures.csv," which contains 232,725 samples (songs) and 18 features (song properties). By using the Pandas read_csv function, we can efficiently load and analyze the dataset.

From the code I got the result:

```
Number of samples: 232725
Number of features: 18
```

```
Amunt of classical songs: 9256
Amunt of pop songs: 9386
        liveness  loudness  label
104022    0.0762   -21.356      0
104023    0.1060   -34.255      0
104024    0.0916   -28.215      0
104025    0.1730   -37.264      0
104026    0.0858   -35.213      0
...          ...       ...    ...
167297    0.0776   -25.477      0
167298    0.2450   -28.192      0
167299    0.0816   -25.843      0
167300    0.1050   -20.238      0
167301    0.0953   -29.223      0

[18642 rows x 3 columns]
```

Task 1b)

The dataset was filtered to include only two genres, "Pop" and "Classical." The code assigns a label of 0 to Classical songs and 1 to Pop songs. The features selected for classification are “Liveness” and “Loudness”. Sorting the songs(samples) into two genres (pop and classical) by using labels.

Task 1c)

The dataset was divided into two key components, matrix X and vector Y.
Feature Matrix X: This matrix contains the selected features 'liveness' and 'loudness' for each song, with songs represented as rows. While label vector Y, contains the corresponding genre labels for each song (0 for Classical, 1 for Pop as in 1b)).

The dataset was split into a training set (80% of the data) and a test set (20% of the data) using the train_test_split function, ensuring that the class distribution is provided in both sets by stratifying the labels.

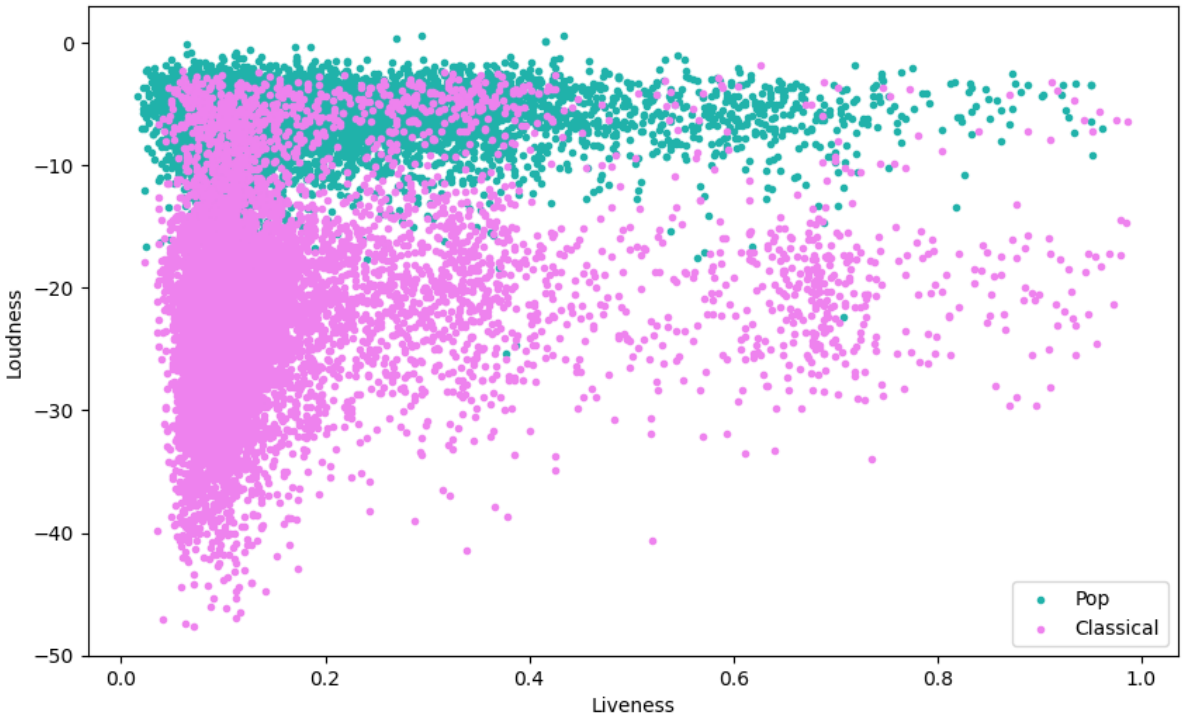
```python
#X Matrix with songs as rows, lables as features
X = pop_classic_data[['liveness', 'loudness']].values  # feature matrix

# y, contains vector with songs
y = pop_classic_data['label'].values  # Target array (labels)

#Training set with 80%training set 20%test set split + the data is shuffeled by using random_state
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

Task 1d)

The scatter plot visualizes the data distribution between the 'liveness' and 'loudness' features for Pop (green) and Classical (purple) songs. From the plot, there is a significant overlap between the two genres, particularly in the lower ranges of 'liveness'. While some separation can be seen in certain regions, the overlapping data points makes the classification task somewhat challenging. Distinguishing between the two genres based on these two features alone may not yield perfectly accurate results, as the features do not separate the genres.



2a)

In this section, we implemented a logistic regression classifier using stochastic gradient descent (SGD). The logistic regression model is based on the sigmoid function, which helps convert the linear combination of features into probabilities. We initialized weights and bias to zero, and during each epoch, we updated the weights and bias by calculating the gradients to minimize the cost function.

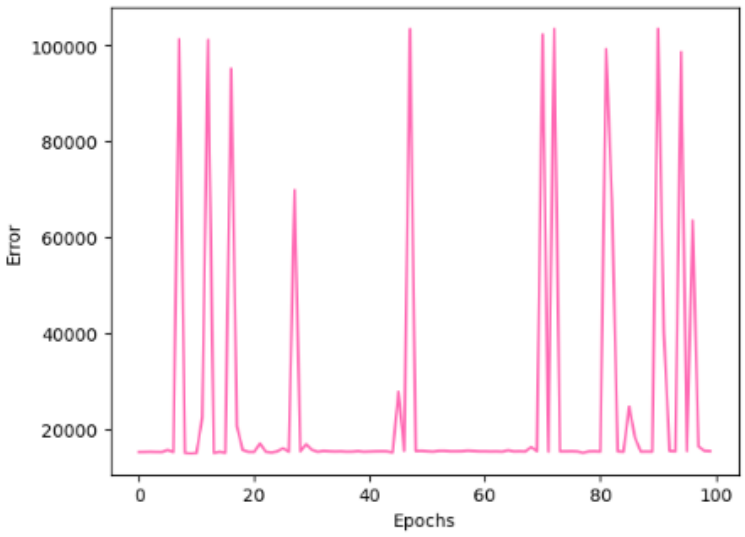


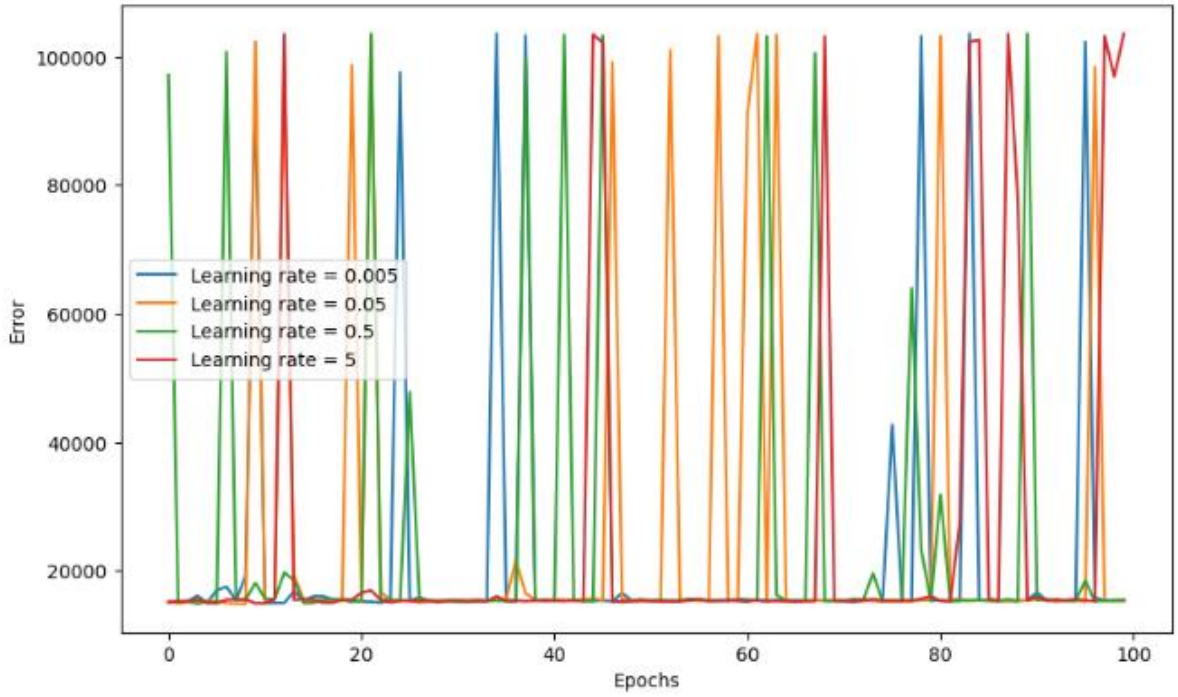
Figure 1 Error w epochs

The training error as a function of epochs:

The graph (Fig1) illustrates the fluctuation of the training error over 100 epochs during the training of a logistic regression model using stochastic gradient descent (SGD). The error has significant spikes throughout the training process with

some stable solution for some parts of the epochs, exp between epochs 50 and 70. These sharp increases and decreases might come from high learning rate. The model may be overshooting the optimal solution, causing the error to increase and decrease dramatically. Or it can have gradient instability. If the gradient values are large, it can cause drastic changes in weights, leading to erratic behavior in the error over epochs. Also why we use an weight list in the code.

Different learning rates:



The graph compares the training error over 100 epochs for multiple learning rates (0.005, 0.05, 0.5, and 5) during the training of a logistic regression model using stochastic gradient descent (SGD).

Low learning rate 0.005: This rate results in a relatively stable training process, with the error fluctuating minimally across epochs. It shows smoother convergence but at the cost of slower learning.

Moderate Learning Rate 0.05: This rate provides a balance between learning speed and stability. It converges faster than the lower rate but still maintains reasonable stability, making it likely the best choice among the tested rates

```
Learning rate 0.005 gives train accuracy 92.46%
Learning rate 0.05 gives train accuracy 92.56%
Learning rate 0.5 gives train accuracy 92.52%
Learning rate 5 gives train accuracy 49.74%
```

Figure 2 Results from the code

High Learning Rate (0.5): Converges more quickly, the error oscillates significantly showing instability in learning. The model jumps across the solution space, that could prevent proper convergence.

Very High Learning Rate (5): The error fluctuates wildly throughout the epochs, indicating that the model is overshooting the optimal solution. This rate is too high for the model to learn effectively, resulting in erratic behavior.

As conclusion from figure 2. Lower learning rates provide more stable learning but at the expense of longer training times. A learning rate of 0.05 seems to strike a good balance between speed and stability. Higher learning rates lead to severe oscillations and poor convergence.

2b)

The test accuracy was found to be 92.20% with a learning rate of 0.005. This is a strong indication that the model performs well, although not perfect. Lack of significant difference between training and test accuracy indicates that the model has learned the features effectively and is not overfitting to the training data.

3a)

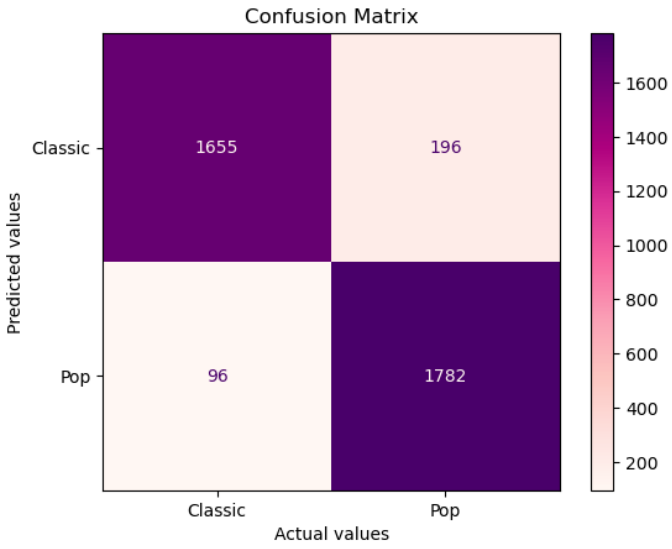
The confusion matrix provides analysis of the classifier's performance on the test set. It compares the actual labels with the predicted labels, offering insights into true positives, true negatives, false positives, and false negatives.

True Positives = Pop correctly classified: 1,782

True Negatives = Classical correctly classified: 1,655

False Positives = Classical misclassified as Pop: 196

False Negatives = Pop misclassified as Classical: 96



3b)

Accuracy vs. Confusion Matrix.
The accuracy score provides a single value representing the overall correctness of the model. It is the proportion of correctly classified samples out of the total samples. Accuracy alone doesn't give an detailed information about the types of errors made by the classifier.

In contrast, the confusion matrix offers a more comprehensive view by breaking down the performance into four categories This allows us to understand specific weaknesses of the model:

True Positives (TP): Correctly classified positive samples

True Negatives (TN): Correctly classified negative samples

False Positives (FP): Misclassified negative samples as positive

False Negatives (FN): Misclassified positive samples as negative

It shows whether the model struggles more with one class than the other. It highlights where misclassifications happen, helping to identify potential areas for improvement.

In summary, while accuracy provides a high-level overview, the confusion matrix gives deeper insights into the model's classification behavior and potential misclassification patterns.

3c)

In this task, the goal was to recommend Classical songs that could appeal to a Pop music fan. The recommendation was based on misclassifications made by the logistic regression model, specifically when Classical songs were misclassified as Pop. These misclassifications suggest that these Classical songs share certain similarities with Pop songs in terms of the selected features ('liveness' and 'loudness'). This is the result I got when asked for 3 recommendations:

```
Classic songs for Pop fan:
                  track_name     artist_name  liveness  loudness
24   Quand je monte chez toi  Henri Salvador     0.143    -7.287
34  Ambarsare Diyan Warhiyan          Chorus     0.222   -10.732
44         Dancing with Gene        Ken Page     0.158    -8.904
```

Sources:

Rahman, M. S., & Islam, M. R. (2022). Effect of feature selection on the accuracy of music popularity classification using machine learning algorithms. Retrieved from ResearchGate: https://www.researchgate.net/publication/364970799