Assignment 1

Name: Nicodemus Ong

Student ID: 22607943

Note: For this assignment I have opted to scale the data using the StandardScalar() function. The primary motive of doing so is to:

1. Boost performance and computational speed. Pixel values in images can range from 0 to 255. When using machine learning algorithms that compute distances or assume normality, having features on the same scale can help the algorithm converge more quickly and perform more accurately.
2. It ensures that the gradient descent moves smoothly towards the minima and helps the algorithm to converge faster. Without scaling, features with larger ranges could disproportionately influence the gradient updates and cause the optimizer to oscillate.
3. For algorithms like k-Nearest Neighbors (k-NN), which rely on distance calculations, having features on different scales can distort the distance measurements. For example, if one pixel tends to have higher values or greater variance than others, it might disproportionately affect the distance computations.
4. When regularization is used (like L1 or L2 regularization in logistic regression), it applies penalties to the size of the coefficients associated with each feature. If the features are on different scales, the regularization term can unfairly penalize some features more than others, simply due to the scale and not the actual importance of the feature.

For these reasons, I have opted to scale pixel values by using the StandardScalar() library. This helps to normalize the contribution of each pixel to the input features and makes the training process more stable and efficient.

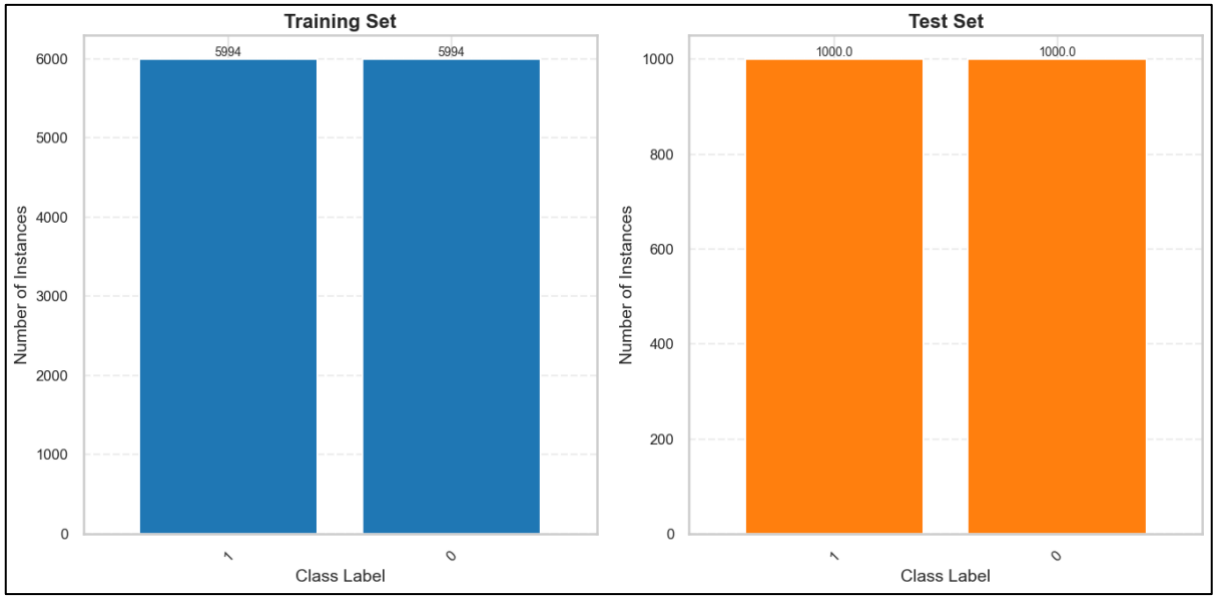
D1:

Instance Description Table:

Instance Description:	
Number of instances in the training set:	11988
Number of instances in the test set:	2000
Total number of instances:	13988

As we can see after the split was done, the training set has a total of 11988 instances and the test set has a total of instances of 2000. The total instance of the dataset for sandals and sneakers has a total of 13988 instances.

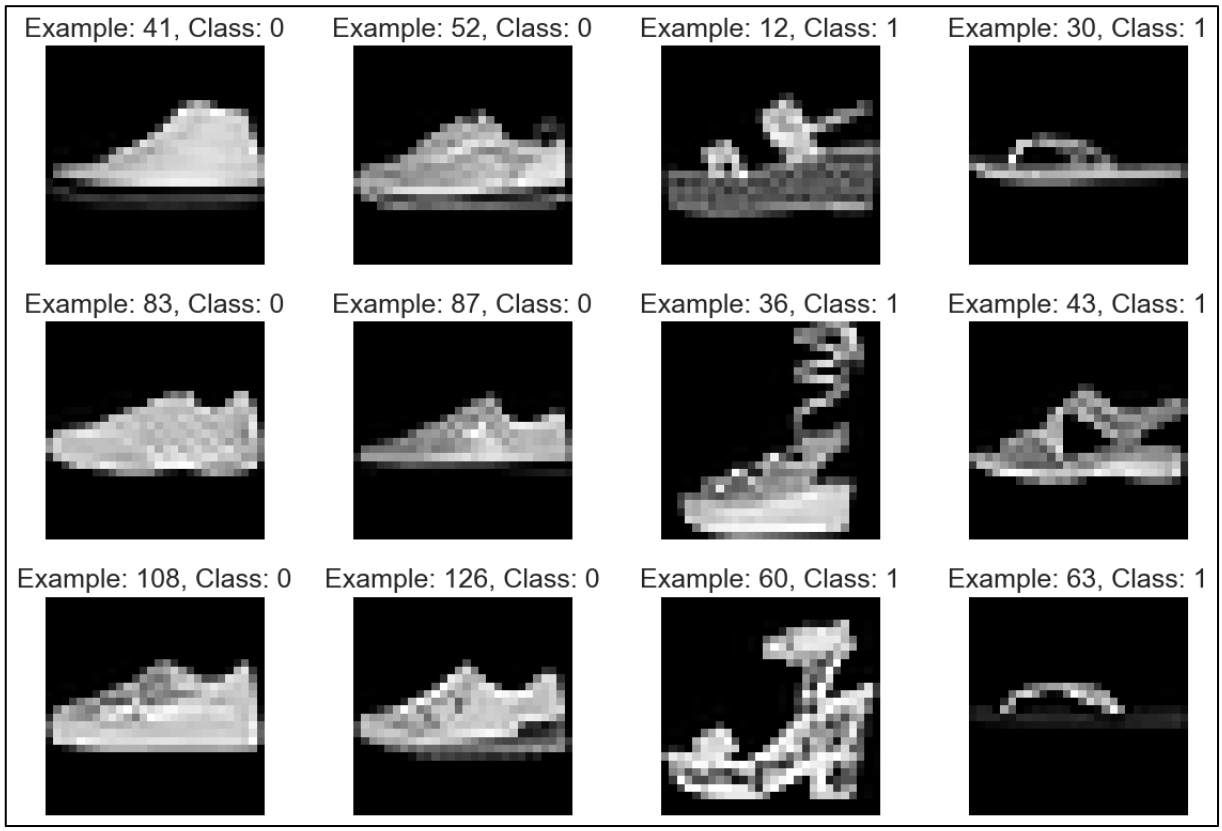
D2:



The balance of the dataset between the classes is equal between both classes (sandals and sneakers) with a random seed of 5508 set.

D3:

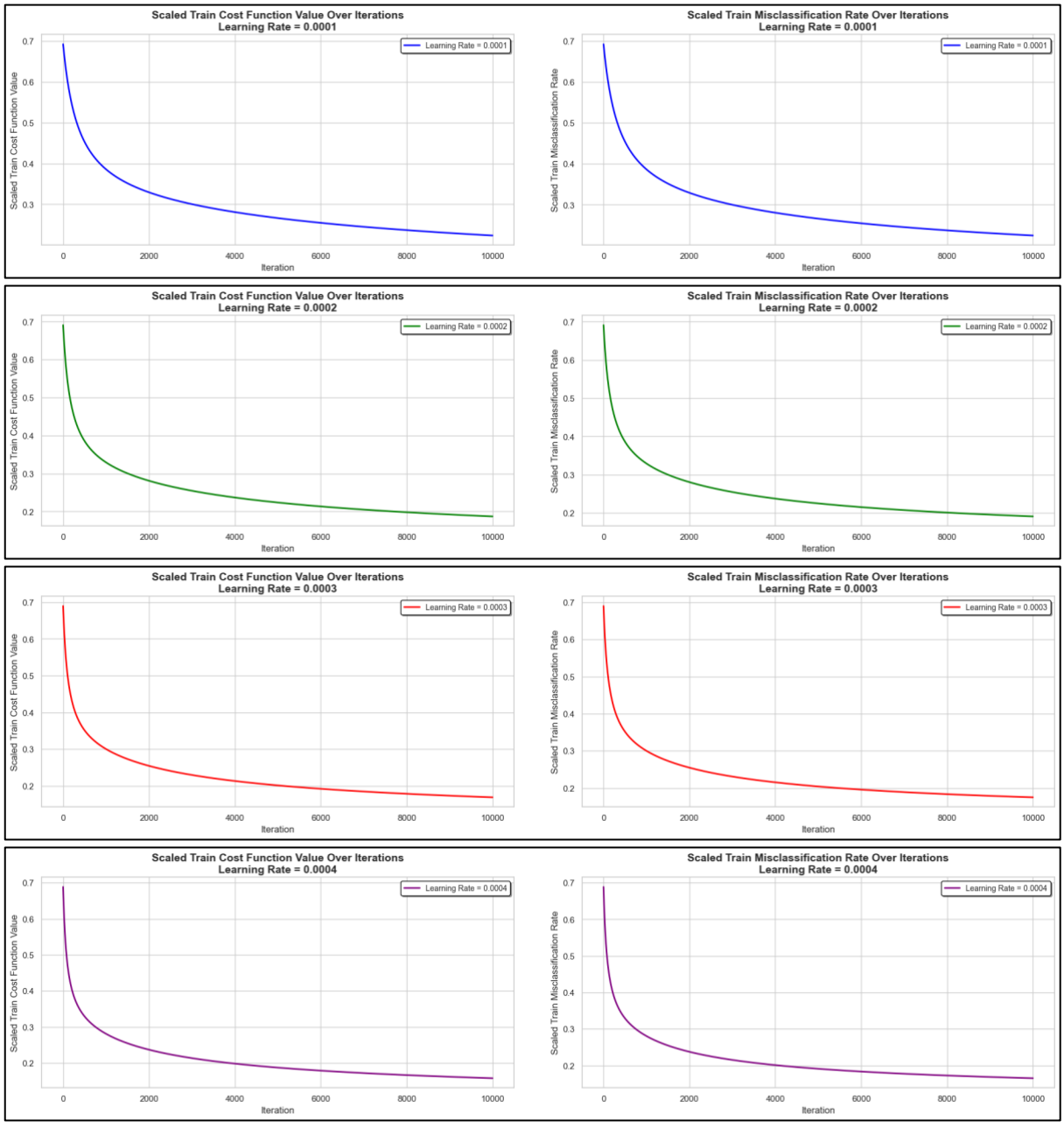
Plotting the first six images/examples from each class with the corresponding *example id* and associated label on the top of the plot.

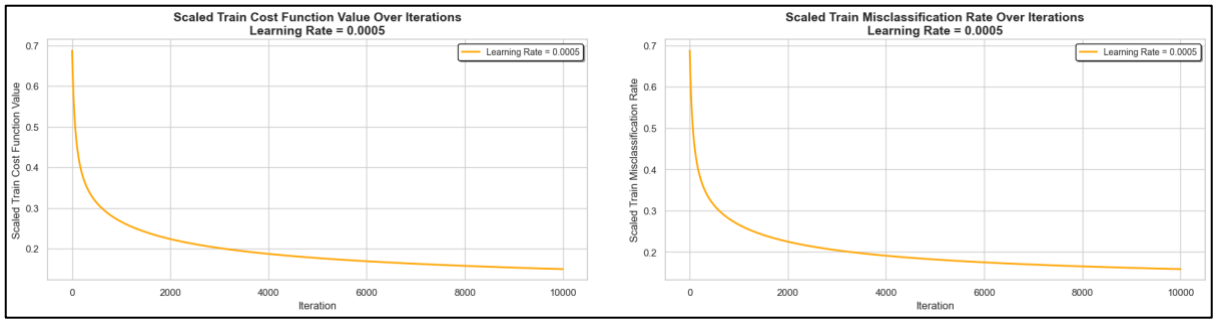


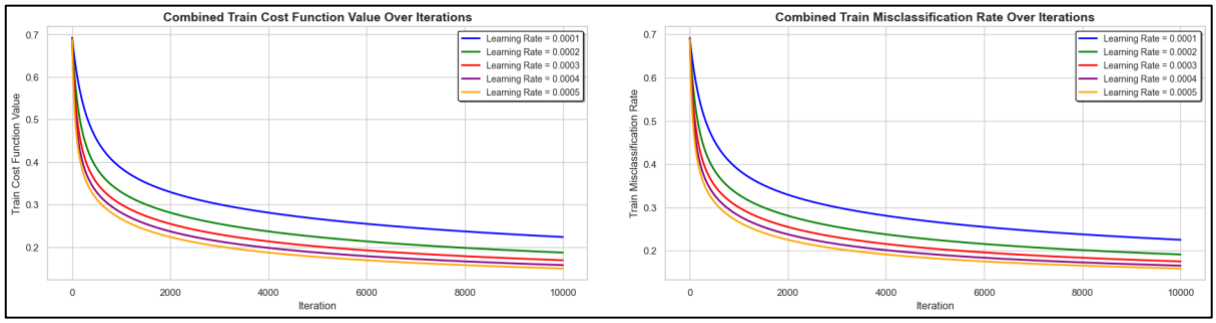
D4:

Based on these observations, a learning rate that balances the speed of convergence with stability and good generalization is preferable. A learning rate of 0.0001 or 0.0002 might be a good choice, as they both converge relatively quickly without showing signs of oscillation or overfitting. However, the decision also depends on computational resources and time constraints. Hence, based on the provided plots, a learning rate of 0.0001 or 0.0002 would be justified as it appears to offer a good trade-off between convergence speed, final cost/misclassification values, and stability during training, leading to a model that generalizes well without overfitting.

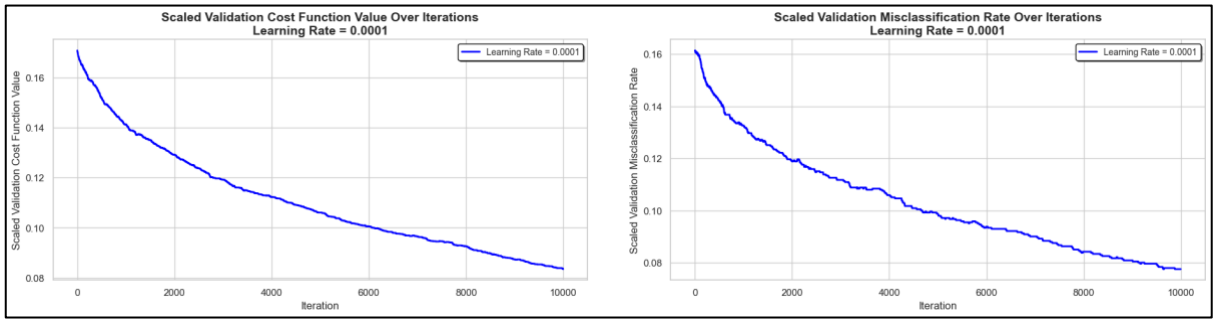
Cost and misclassification across Learning rates [0.0001, 0.0002, 0.0003, 0.0004, 0.0005] (Training data):

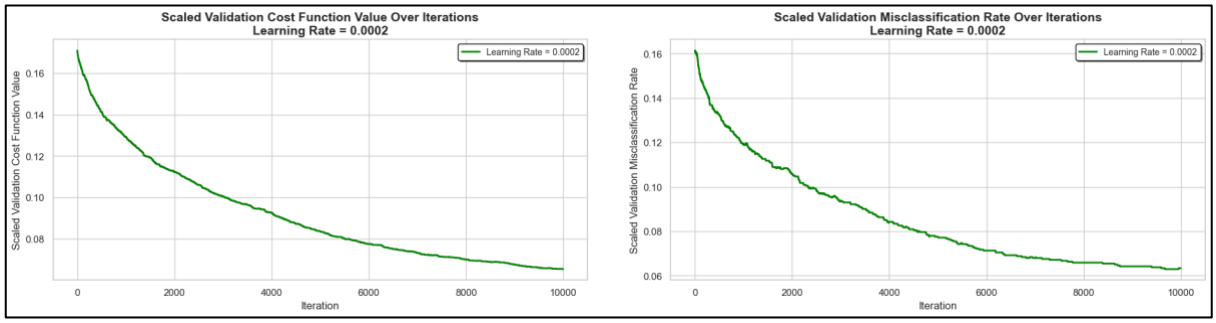


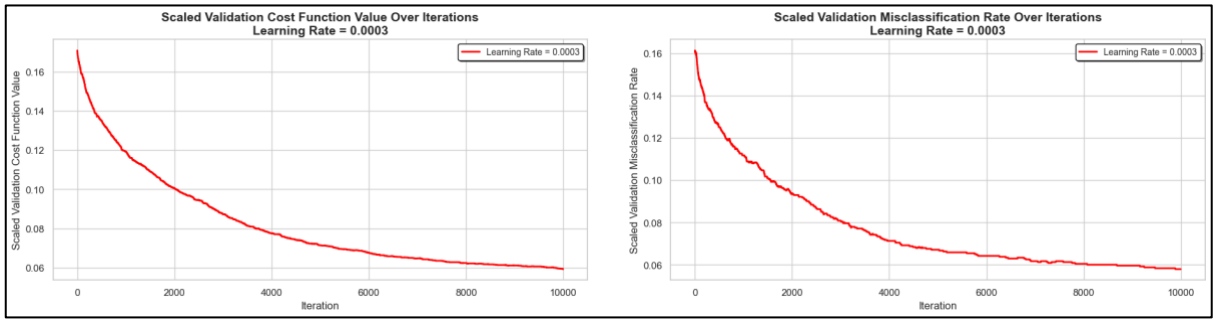


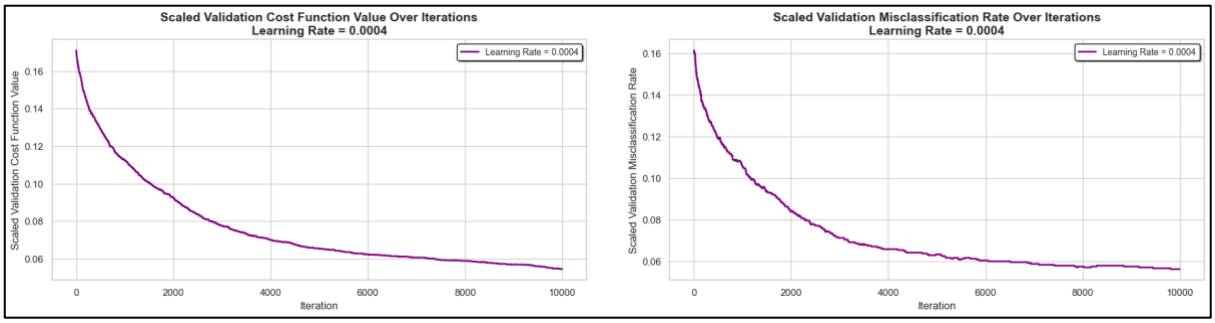


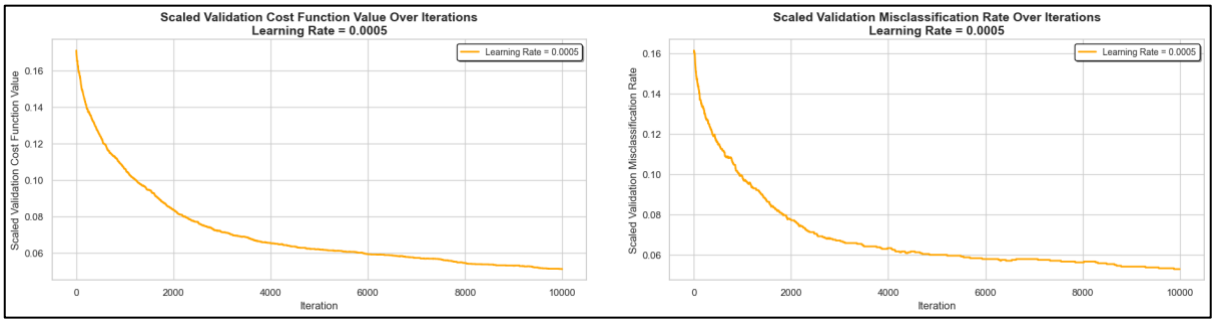
[Cost and misclassification across learning rate [0.01, 0.02, 0.03, 0.04, 0.05] (Testing data):]

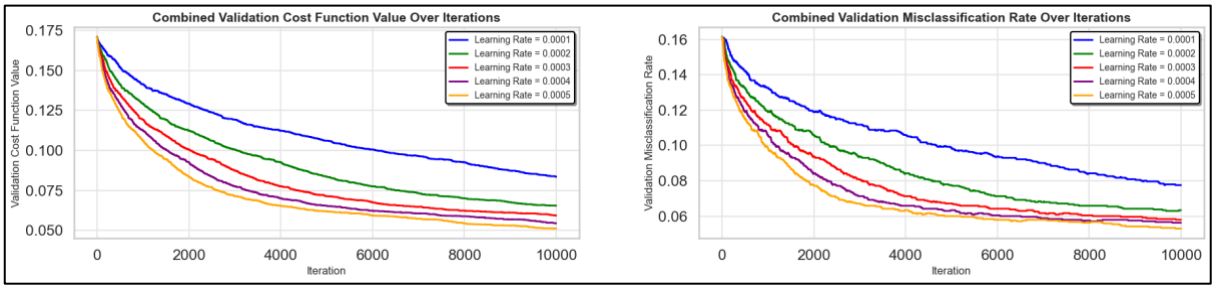







Scaled Validation Cost Function Value Over Iterations
Learning Rate = 0.0004
Learning Rate = 0.0004
Scaled Validation Cost Function Value
Iteration
Scaled Validation Misclassification Rate Over Iterations
Learning Rate = 0.0004
Learning Rate = 0.0004
Scaled Validation Misclassification Rate
Iteration


Scaled Validation Cost Function Value Over Iterations
Learning Rate = 0.0005
Learning Rate = 0.0005
Scaled Validation Cost Function Value
Iteration
Scaled Validation Misclassification Rate Over Iterations
Learning Rate = 0.0005
Learning Rate = 0.0005
Scaled Validation Misclassification Rate
Iteration

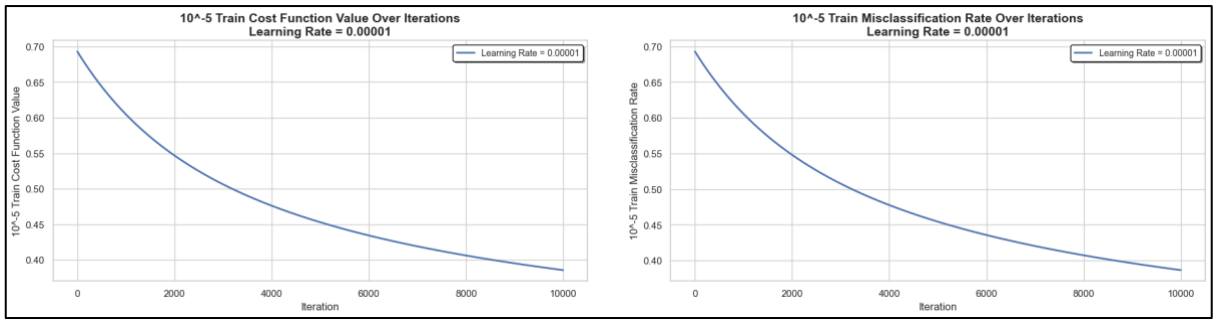

Combined Validation Cost Function Value Over Iterations
Learning Rate = 0.0001
Learning Rate = 0.0002
Learning Rate = 0.0003
Learning Rate = 0.0004
Learning Rate = 0.0005
Validation Cost Function Value
Iteration
Combined Validation Misclassification Rate Over Iterations
Learning Rate = 0.0001
Learning Rate = 0.0002
Learning Rate = 0.0003
Learning Rate = 0.0004
Learning Rate = 0.0005
Validation Misclassification Rate
Iteration

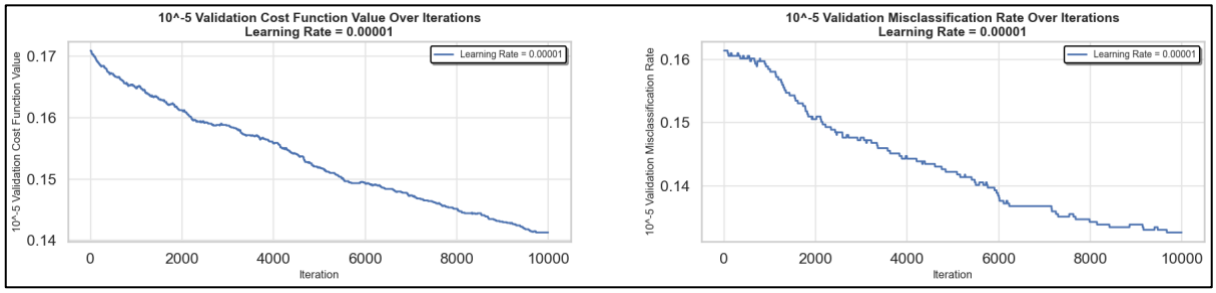
D5:

Plotting:

- In the left plot, show the values of the cost function (y-axis) for each iteration (x-axis) for the training and validation sets. That is, your plot should contain two results.

- In the right plot, show the fraction of misclassifications (y-axis) for each iteration (x-axis) using the logistic regression model prediction for a threshold of 0.5. Similarly, you should provide two results (one for the training and one for the validation set).





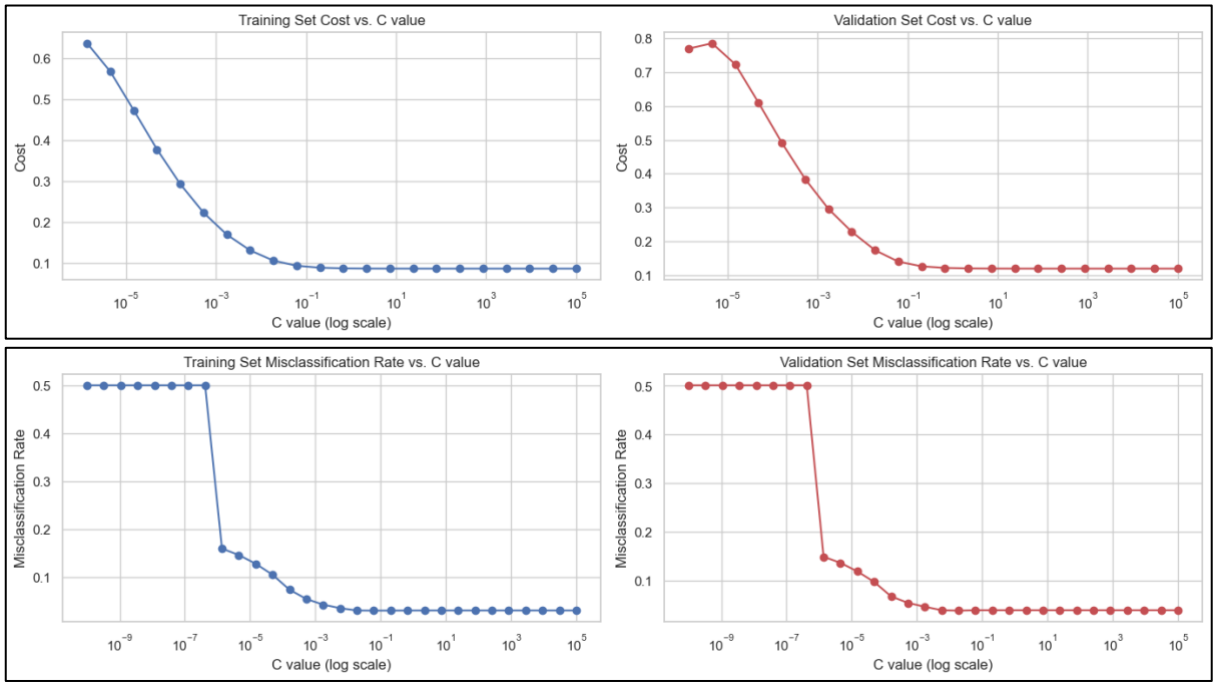
D6:

Given these results, the current learning rate is a safe choice, particularly if the goal is to prevent overfitting and ensure a stable convergence. If faster training is desired and there is room to monitor and control for potential overfitting, a slight increase in the learning rate could be tested. It's also important to note that, as the misclassification rate is levelling off in the validation set, further training beyond 10,000 iterations may not lead to substantial improvements and could risk overfitting. Regularization techniques, early stopping, or a learning rate schedule could be employed to handle this if further iterations are considered.

D7:

Plotting:

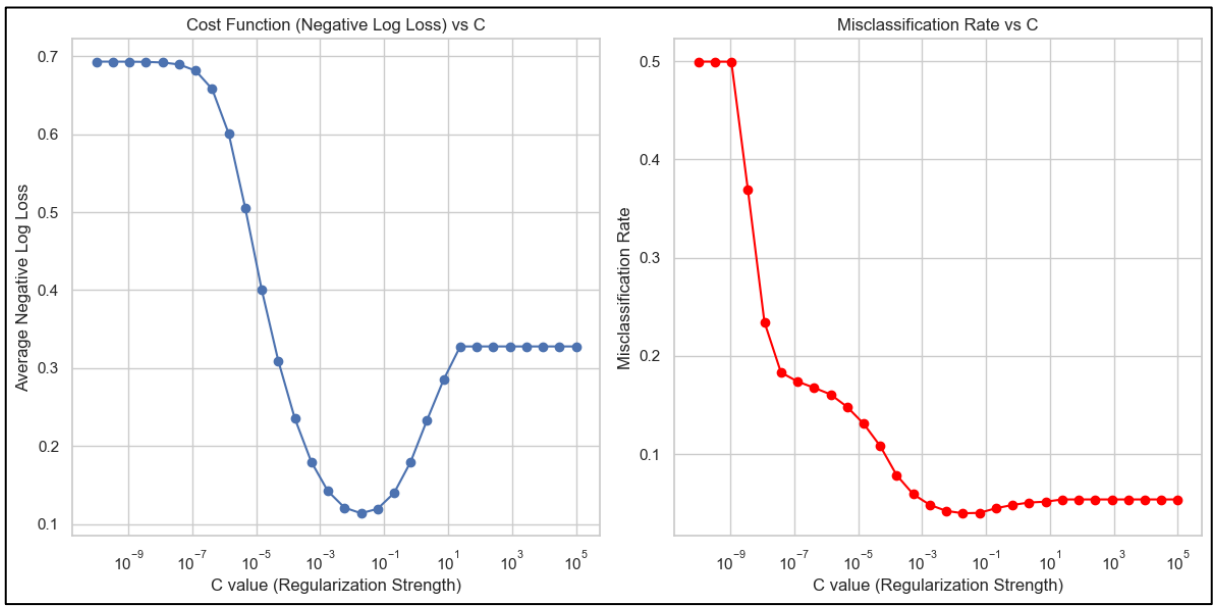
- In the left plot, show the values of the cost function (y-axis) for each C value (x-axis) for the training and validation sets. That is, your plot should contain two results.
- In the right plot, show the fraction of misclassifications (y-axis) for each C value (x-axis) using the logistic regression model prediction for a threshold of 0.5. Similarly, you should provide two results (one for the training and one for the validation set).



D8:

Plotting:

- In the left plot, show the values of the cost function (y-axis) for each C value (x-axis). Note that you will have ten values for each C value (10 folds). Therefore, you should report the average value.
- In the right plot, show the fraction of misclassifications (y-axis) for each C value (x-axis) using the logistic regression model prediction for a threshold of 0.5.



D9:

I would choose C as 10^-2. The misclassification rate becomes relatively flat after 10^-2, indicating that the model has reached a level of complexity that is sufficient for the given problem, and further reducing regularization does not lead to better performance. The U-shaped curve in the third plot suggests that 10^-2 is in the sweet spot where the model is complex enough to learn the underlying patterns in the data but not so complex that it starts to overfit. Hence 10^-2 will be the hyperparameter value that I will choose.

D10:

D10 Results:

D10 Results				
Optimal C	Train Loss	Validation Loss	Train Misclassification Fraction	Validation Misclassification Fraction
52.983169	0.108462	0.136323	0.034179	0.041084

D11:

Interpreting the results from the above table:

- The train loss is 0.1084620.108462, which is relatively low, indicating that the model fits the training data well.
- The validation loss is slightly higher at 0.1363230.136323, which is expected as models generally perform a bit worse on unseen data. However, the difference is not substantial, which suggests good generalization.
- The train misclassification fraction is 0.0341790.034179, which is quite low and indicates high accuracy on the training data.
- The validation misclassification fraction is 0.04010840.0401084, only slightly higher than the training misclassification rate, which suggests that the model has generalized well and is not overfitting.

The actual optimal C value from the results is much higher than what was estimated from the plots. In the plots, the curve began to plateau around C=10^-2, but the value from the results suggests that a much higher C, implying less regularization, is optimal according to the cross-validation procedure used.

This would suggest that the model requires less regularization than initially thought, perhaps due to the inherent complexity of the dataset or its size. These results are valuable as they provide a concrete C value to use for this model, along with performance metrics that suggest good generalization from training to unseen data.

D12:

The Precision-Recall curve before selecting the best alpha shows very high precision for the majority of recall values but then drops sharply as recall approaches 1. This suggests that the model is very conservative, only making positive predictions when very certain, which leads to high precision but not full recall.

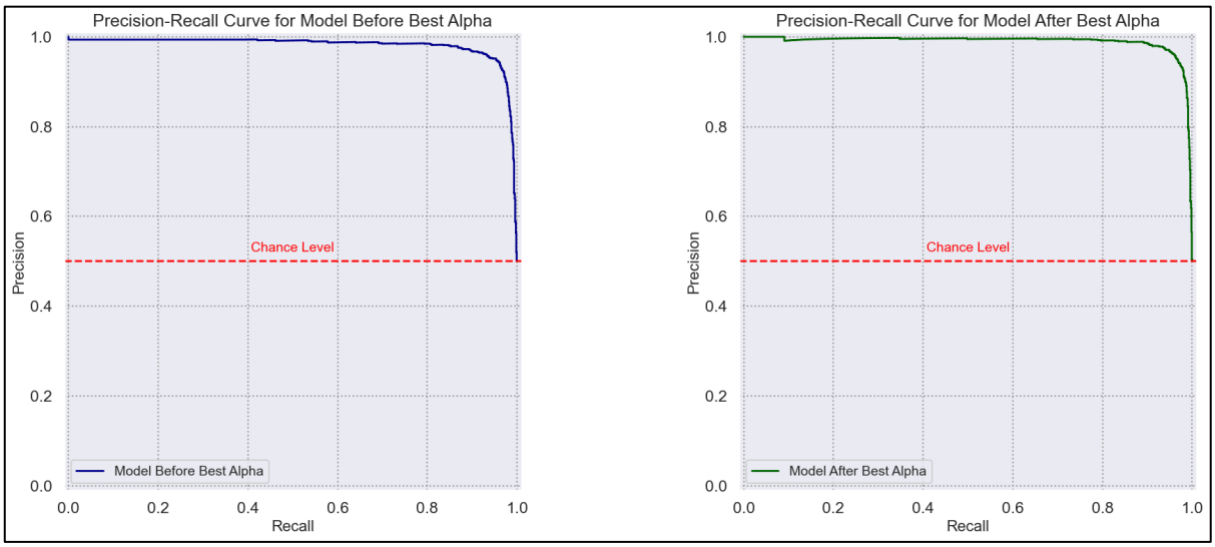
After selecting the best alpha, the Precision-Recall curve hugs the top-right corner, showing both high precision and high recall across the entire range of threshold values. This indicates a highly effective model that can correctly identify a large fraction of positive instances without a high number of false positives.

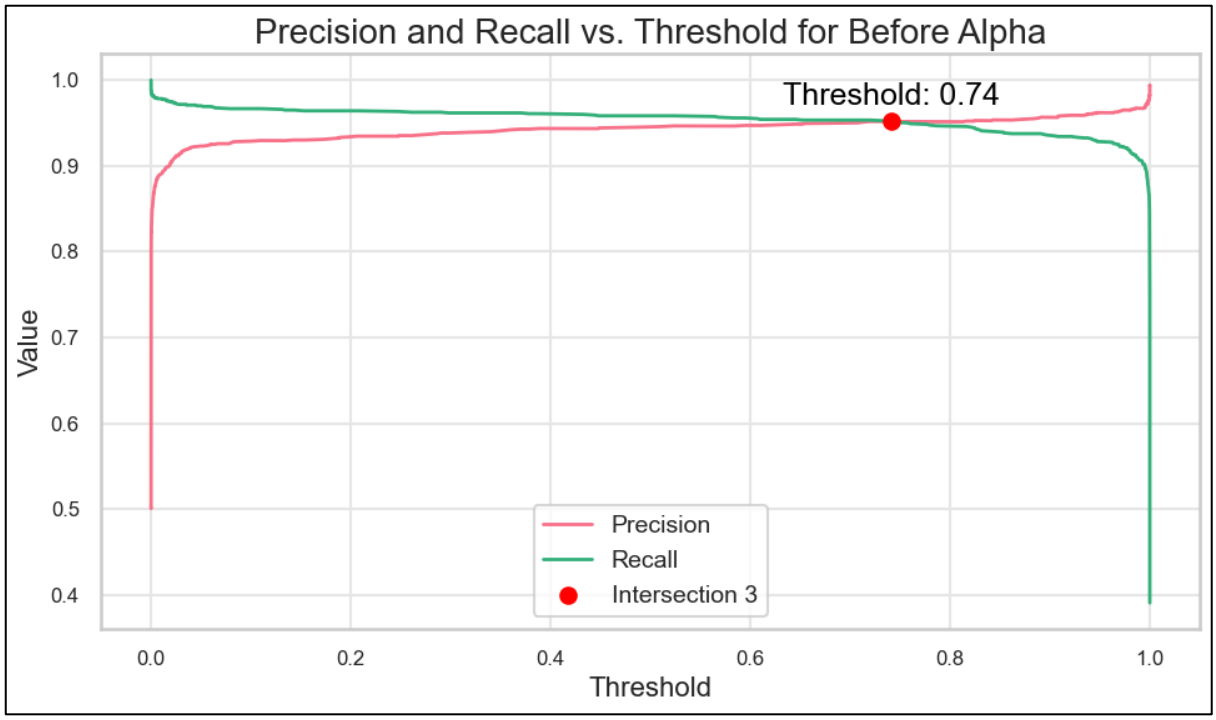
A well-performing model would maintain high precision as recall increases. The performance measure indicates that after tuning the model with the best alpha, the model's performance has improved, being able to maintain high precision over a larger range of recall values compared to before.

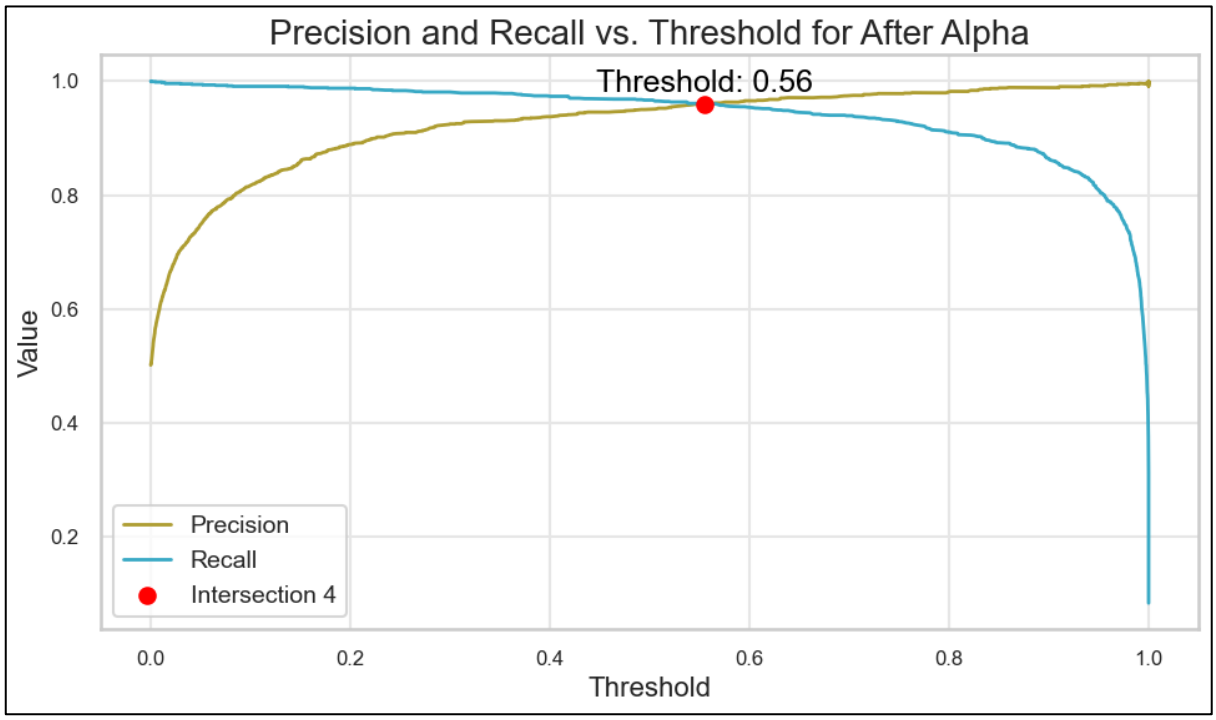
The choice of a threshold is a trade-off depending on the cost of false positives vs. false negatives. If the cost of a false negative is high (meaning it's critical to catch all true positives), you would favour a threshold that gives you a higher recall. Conversely, if the cost of a false positive is higher (you need to be sure about your positive predictions), you'd choose a threshold that gives you higher precision.

after applying the best alpha, the model's performance on the validation set is excellent, showing a strong ability to distinguish between classes while maintaining high precision and recall. The threshold chosen will depend on the specific requirements of the validation task and the relative costs of misclassification types. The PR curve after tuning indicates that you have a range of thresholds to choose from that maintain high precision, allowing for flexibility based on specific needs.

Below are the plots to relate to the above explanation.




Precision and Recall vs. Threshold for Before Alpha
Threshold: 0.74
Value
Threshold
Precision
Recall
Intersection 3
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0


Precision and Recall vs. Threshold for After Alpha
Threshold: 0.56
Value
Threshold
Precision
Recall
Intersection 4
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0

D13:

Here are the results for D13:

D13 Results	
Best Threshold:	0.72
Best F1 Score:	0.95
New Misclassification Fraction:	0.0480
Improvement in Misclassification Fraction:	0.0135

The best threshold of 0.72 suggests that this is the point where the balance between precision and recall is optimized according to the F1 score, which is the harmonic mean of precision and recall. This threshold is neither too strict nor too lax, striking a balance that maximizes the classifier's performance on the validation set. An F1 score of 0.95 is exceptionally high, indicating that the model has a very good balance of precision and recall. In practical terms, the model is able to correctly identify true positives while keeping the number of false positives and false negatives low. The misclassification fraction of 0.0480 indicates that 4.8% of the validation set examples were misclassified under the best threshold. This is the error rate of the model on the validation set. The improvement in misclassification fraction of 0.0135 shows a significant reduction in the rate of misclassified examples when compared to the previous threshold, which would be the one used by default (often 0.5 for binary classifiers).

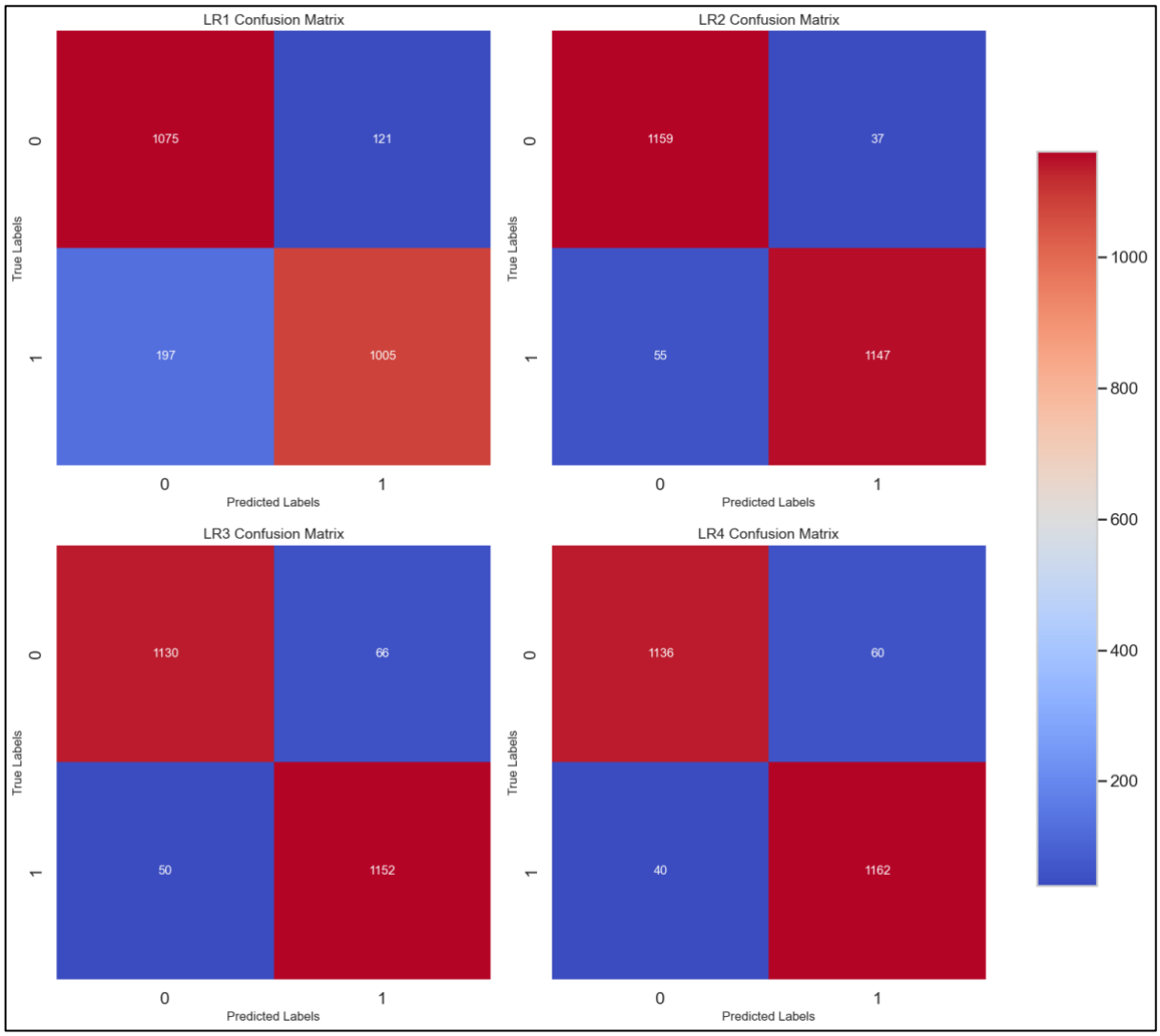
The improvement in misclassification fraction demonstrates that the fine-tuning process indeed found a better operational point than the default, likely leading to fewer errors in the model's predictions on the validation set.

D14:

Here are the results for D14:

Model Performances			
Model:	Precision:	Recall:	False Positive Rate:
LR1	0.89	0.84	0.10
LR2	0.97	0.95	0.03
LR3	0.95	0.96	0.06
LR4	0.95	0.97	0.05

Confusion Matrix for all models:



D15:

Here are the comparison for D15:

Performance Metrics:

- LR1: Shows good precision and recall, but it has the highest false positive rate among the models. This might indicate it's more prone to classifying negative instances as positive.
- LR2: Has excellent precision but a slightly lower recall than LR1. It significantly reduces the false positive rate compared to LR1, which suggests it's better at identifying true negatives.
- LR3: Offers a balance between precision and recall, both very high, with a low false positive rate. It seems to provide the best balance among all models.
- LR4: Similar to LR3, it has high precision and the highest recall, but its false positive rate, while low, is slightly higher than LR3.

Confusion Matrices:

- LR1: Shows a higher number of false positives and false negatives compared to the other models.
- LR2: Decreases false positives at the expense of some true positives (lower recall).
- LR3: Has fewer false positives and false negatives than LR1 and LR2, indicating a better balance.
- LR4: Similar to LR3, with even fewer false negatives but slightly more false positives.

Generalization Capacity:

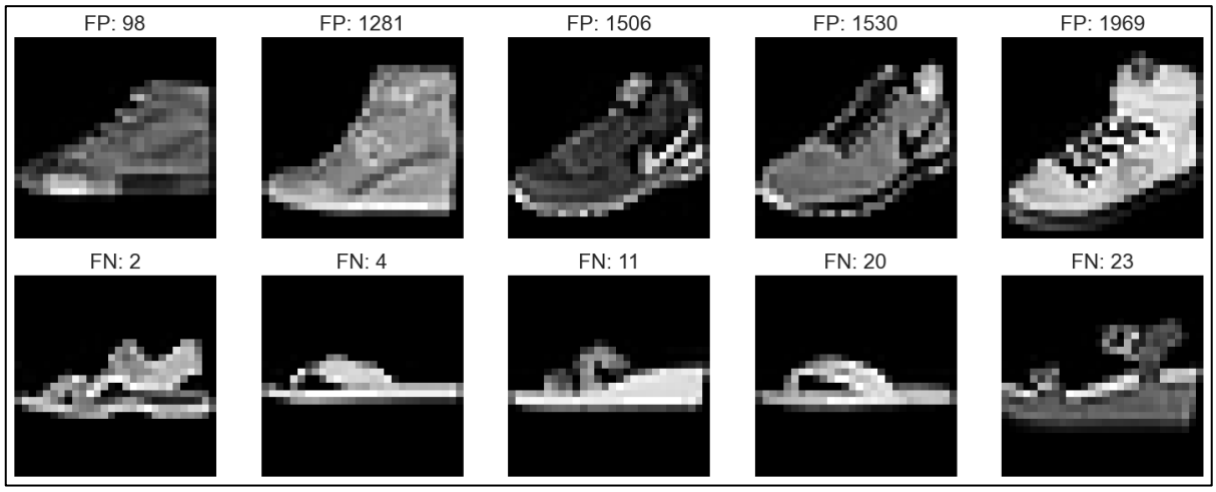
- LR1: Its higher false positive rate may indicate it's not generalizing as well as the others, potentially overfitting to the positive class.
- LR2: The reduction in false positives indicates it has a better generalization capacity than LR1. However, its lower recall suggests it might be underfitting slightly or being too conservative in predicting the positive class.
- LR3 and LR4: Both demonstrate excellent generalization capacity with high precision and recall and low false positive rates. They are likely the best at generalizing to unseen data, with LR4 being slightly better at identifying positive cases and LR3 at identifying negative cases.

Given the performance metrics and confusion matrices, LR3 and LR4 appear to have the strongest generalization capacity. They manage to maintain high precision and recall with a very low rate of false positives. In real-world scenarios, the choice between these models

would likely depend on the particular cost associated with false positives versus false negatives. If predicting true positives (recall) is more critical, LR4 would be preferred. If avoiding false positives is more important, LR3 might be a better choice.

D16:

showing five images that are false positives on the test set and five images that are false negatives



D17:

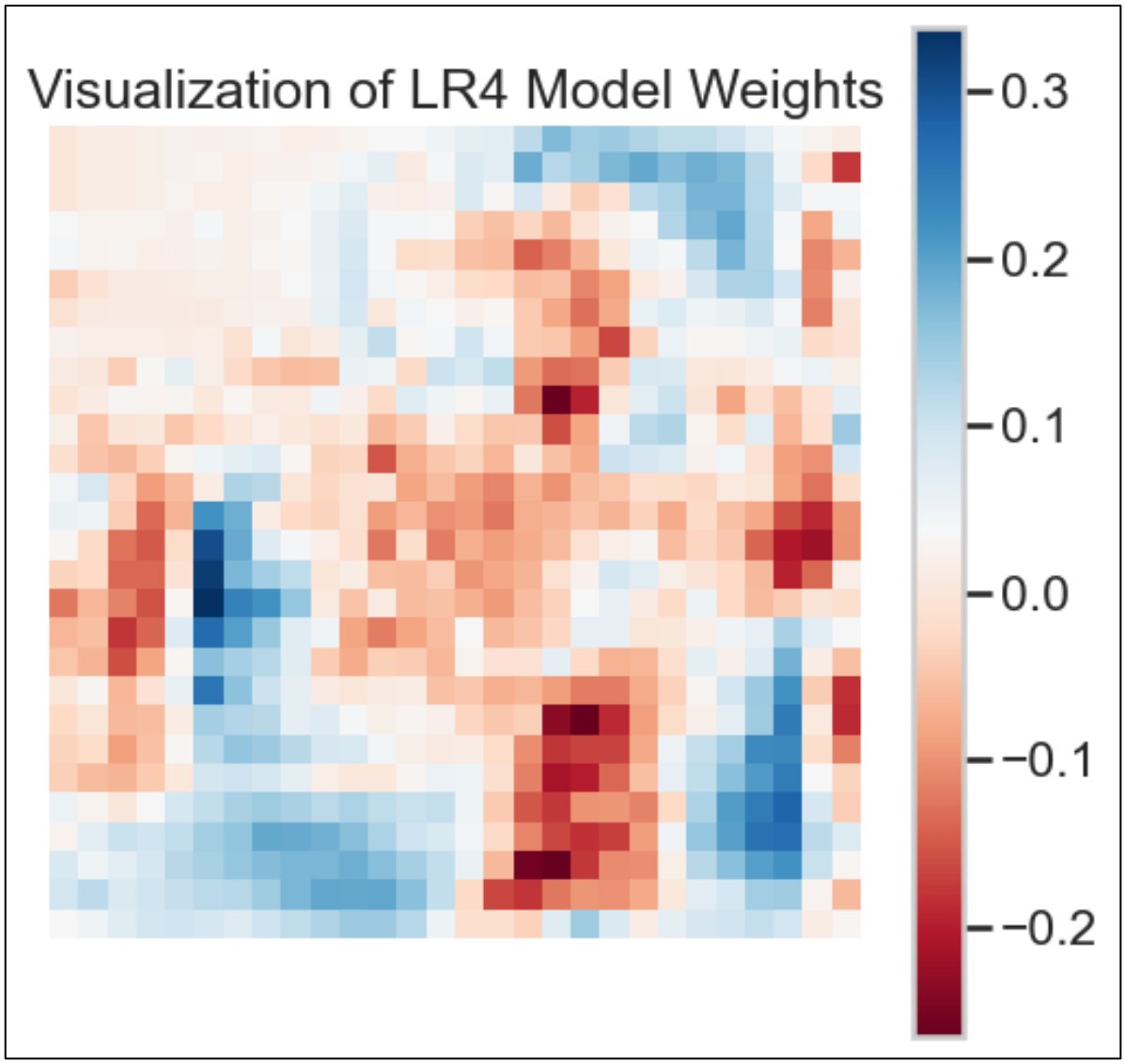
The model seems to struggle with shoes that have complex patterns or high contrast, possibly misinterpreting these features as indicative of the positive class and thus leading to false positives.

For false negatives, the issue appears to be the opposite: the model might be associating the lack of complexity or distinct features with the negative class, causing it to miss when such shoes actually belong to the positive class.

The mistakes could be due to overemphasis on certain features that are not as strongly indicative of class membership as the model assumes. This might be a result of the training data not having enough variety in the examples for the model to learn the true variability within each class.

D18:

This Heatmap shows the weights of matrix to associate the weights with the images.



D19:

Areas with Strong Positive Weights (Red): These are likely areas where the presence of certain pixel intensities is strongly associated with the positive class. If LR4 was distinguishing between sandals and sneakers, these red areas might correspond to the distinctive parts of sneakers, such as thicker soles, more coverage on the top of the foot, or specific patterns that are common in sneaker designs.

Areas with Strong Negative Weights (Blue): These areas are likely where the absence of certain pixel intensities or the presence of opposite pixel intensities is associated with the negative class. In the context of distinguishing between sandals and sneakers, these blue areas might correspond to features characteristic of sandals, such as open areas (straps instead of a closed shoe), thinner soles, or the shape of the footbed.

Areas with Neutral Weights (White or Light Colours): These regions seem to have little to no impact on the model's predictions. They are probably parts of the image that do not vary significantly between sandals and sneakers or do not provide useful information for distinguishing between the two.

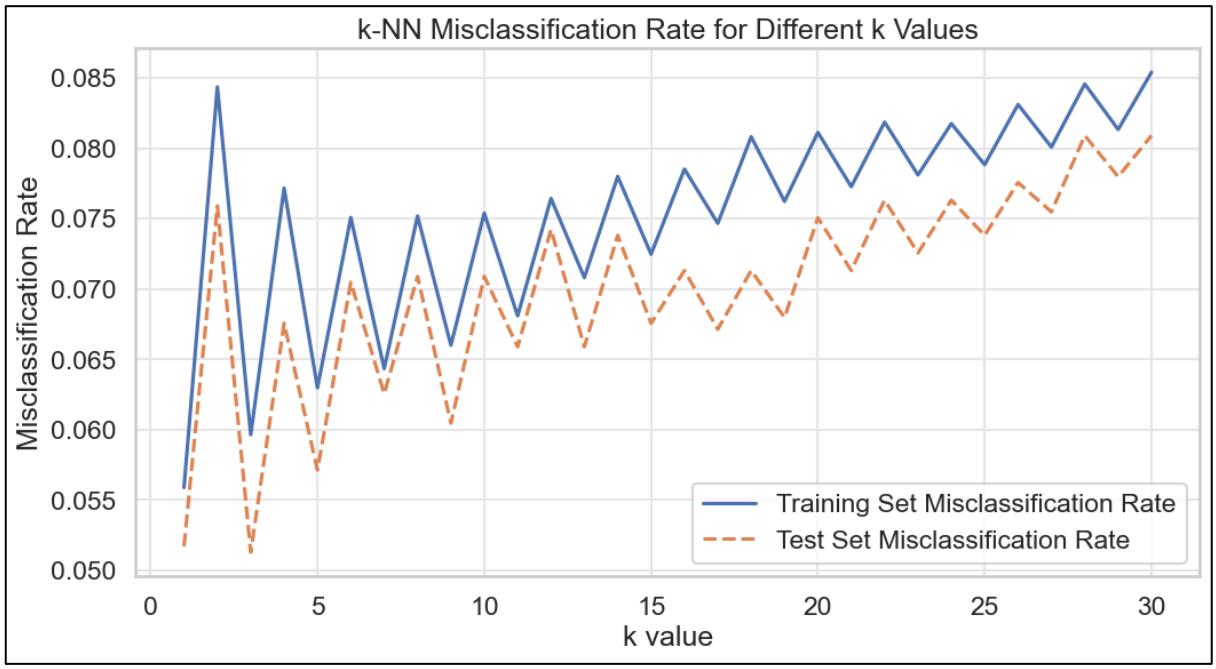
The model likely learned to focus on certain patterns and shapes that are uniquely different between sandals and sneakers. For instance, if the positive class is sneakers, the red areas may represent the typical high-coverage top of a sneaker or the distinctive sole pattern. If the positive class is sandals, the red areas might correspond to straps or open spaces.

Conversely, the blue areas suggest where the model expects to see less intensity for the positive class, which could be regions typically occupied by parts of sandals if the positive class is sneakers, and vice versa.

The magnitude of the weights signifies the importance of that feature in making a classification decision. Larger absolute values (either positive or negative) indicate features that are highly influential in the model's predictions.

D20:

plot showing the fraction of misclassifications (y-axis) for each k value (x- axis) for the training and validation set



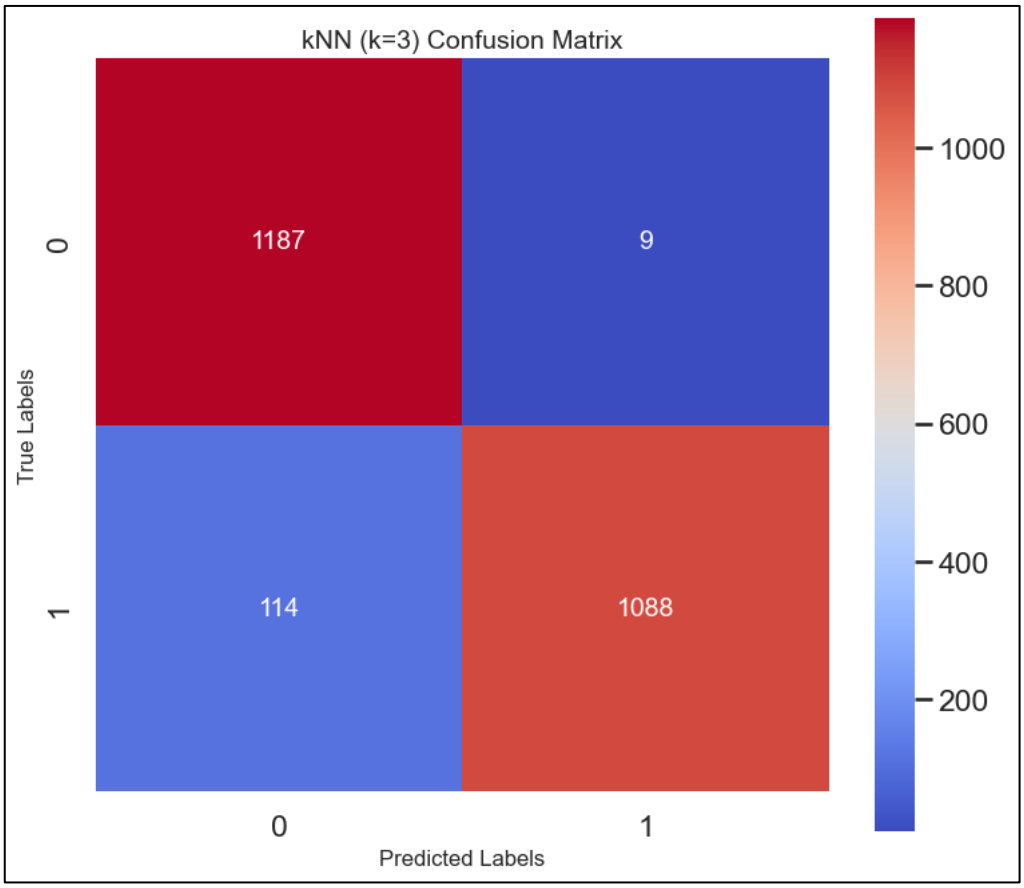
D21:

- Training Set Misclassification Rate: Generally, as k increases, the training misclassification rate increases. This is expected because a larger k smooths the decision boundary by considering more neighbors, which reduces the model's sensitivity to the noise in the training data, and thus the model becomes less tailored to the training set.
- Test Set Misclassification Rate: The test misclassification rate shows a degree of volatility but generally trends upward as k increases. Initially, increasing k may help the model generalize better by reducing overfitting, which can lower the test misclassification rate. However, beyond a certain point, a larger k may oversimplify the model, causing it to miss important nuances and leading to an increase in the test misclassification rate.

The choice of k should be based on the value that minimizes the test set misclassification rate because the main goal is to improve the model's performance on unseen data. In this graph, the test misclassification rate seems to be lowest at around k=6. Around this value, the test set performance is improved without compromising the model's ability to generalize, as indicated by the relatively low misclassification rate. Choosing k=6 strikes a balance between reducing noise and avoiding the creation of a decision boundary that's too simplistic.

D22:

Confusion matrix of KNN:



Performance Metrics of KNN:

KNN Performance Metrics		
Precision:	Recall:	False Positive Rate:
0.992	0.905	0.008

D23:

- Precision: The k-NN model has the highest precision of all models, indicating that it has the highest probability of its positive predictions being correct. However, LR2 and LR4 also show high precision, with LR2 being particularly close to k-NN.
- Recall: The k-NN model has a lower recall compared to LR3 and LR4, which means it has a higher rate of false negatives. However, it has a higher recall than LR1.
- False Positive Rate: The k-NN model has a significantly lower FPR than all logistic regression models, which means it is least likely to falsely label negative instances as positive.

The k-NN model has a superior balance of precision and a very low FPR, but its recall is slightly lower than the best logistic regression models. This suggests that while the k-NN model is excellent at confirming positive instances (high precision and low FPR), it is slightly less capable of identifying all positive instances (lower recall). Logistic regression models LR3 and LR4 demonstrate very good generalization with high recall rates but slightly higher FPR than k-NN. This indicates they may be better at capturing most of the positive instances but at the cost of misclassifying some negatives. LR1 and LR2, while having decent metrics, do not perform as well as LR3, LR4, or the k-NN model in terms of the balance between precision, recall, and FPR.

D24:

Here are some ways that we can improve the modelling of this assignment.

Data Quality and Pre-processing:

- Increase Dataset Size: If feasible, collect more data to provide the model with more examples to learn from.
- Feature Scaling: Apply normalization or standardization to ensure all features contribute equally to the model's predictions. Model Selection and Hyperparameter Tuning:
- Model Selection: Evaluate different models to find the best performer for your specific dataset, considering both simple models (to avoid overfitting) and more complex models (to capture more complex patterns).
- Hyperparameter Tuning: Use methods like grid search, random search, or Bayesian optimization to find the optimal hyperparameters for your model.
- Ensemble Methods: Combine multiple models to improve predictions. Techniques like bagging, boosting, and stacking can lead to better performance.

Evaluation and Validation:

- Cross-Validation: Use k-fold cross-validation to assess the model's performance more reliably and to ensure it generalizes well.
- Performance Metrics: Consider the problem's context to choose the right performance metrics, such as F1-score for imbalanced classes, ROC-AUC for binary classification, etc.

Experimentation and Iteration:

- Error Analysis: Review the model's mistakes to understand where it fails and why. This can inform further improvements in data preprocessing or feature engineering.

- Continuous Iteration: Modeling is an iterative process. Use insights from each round of modeling to refine your approach continuously.
- Experiment Tracking: Keep track of experiments with tools like MLflow or Tensor Board to systematically evaluate changes over time.

Advanced Techniques:

- Deep Learning: For certain types of data, such as images, deep learning models can capture complex patterns that traditional models cannot.
- Transfer Learning: Leverage pre-trained models on similar tasks to improve performance, especially when the dataset is not very large.
- Data Augmentation: In image, generate new training samples through various transformations to increase the robustness of the model.