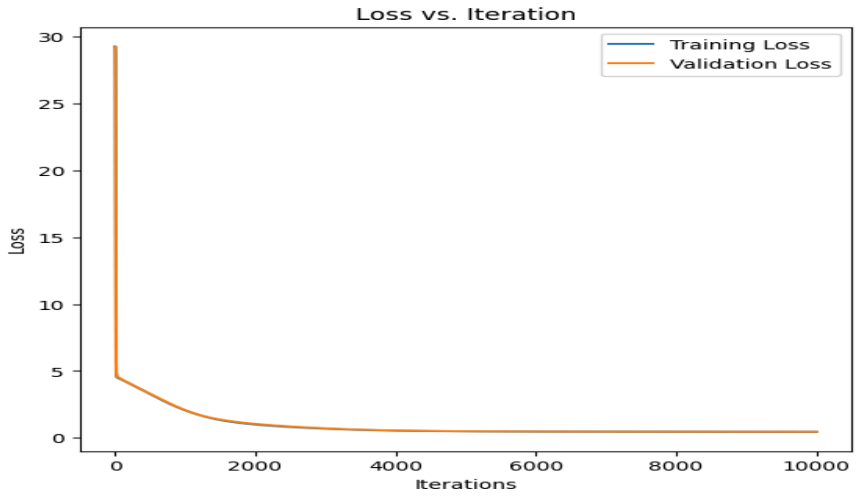
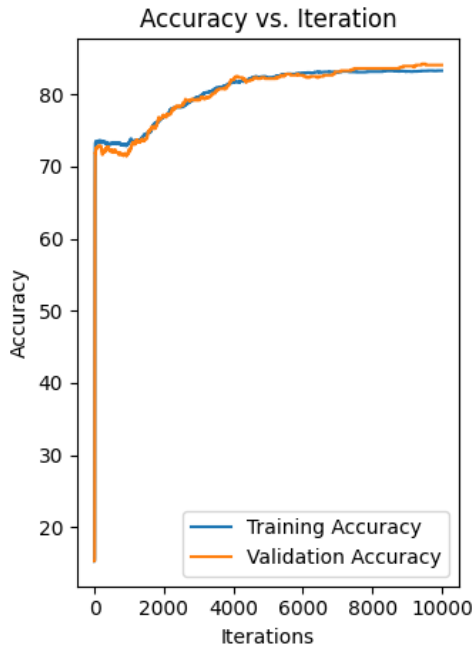
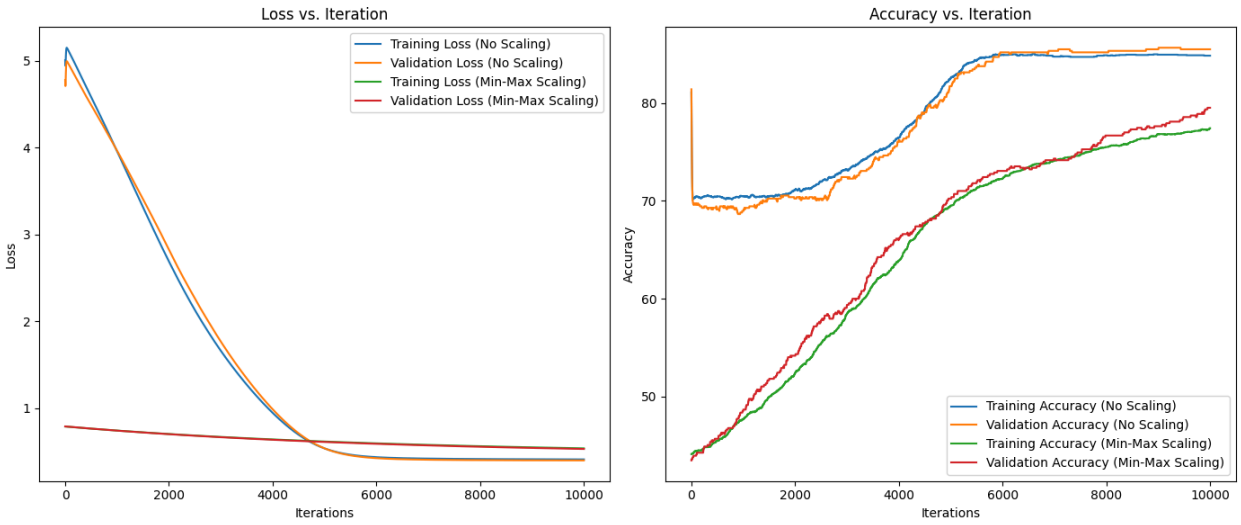


Training a model with a very small learning rate implies that the updates of the model parameters are minimal; hence it may take sufficient time to reduce the loss function. This will surely lead to gradual adjustment, which may take a large number of iterations to converge the model. For instance, in your case, it started off high, and only after roughly 5000 iterations did it start to converge because each update step was quite small. While small learning rates ensure stability, they also require more time for the model to reach an optimal solution. It is easy to tune the learning rate or to use adaptive optimization techniques that allow speeding up of convergence.

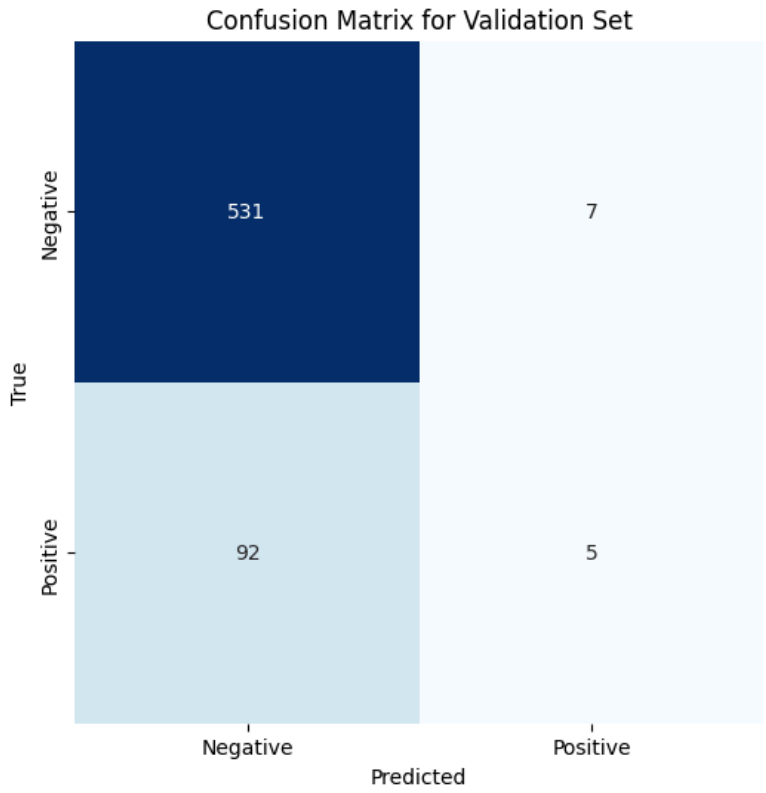


It follows the same trend as the loss function during training: using a small learning rate, the model updates very gradually and therefore improvements in accuracy are very slow. Therefore, convergence of accuracy takes approximately 5000 iterations. At the beginning, the slope of accuracy is very slow since the model slightly updates itself, but gradually it starts leveling off and attains its stability. This is because, while the small learning rate guarantees stability and steady progress, it generally implies a longer training time in order to achieve a desirable level of accuracy.



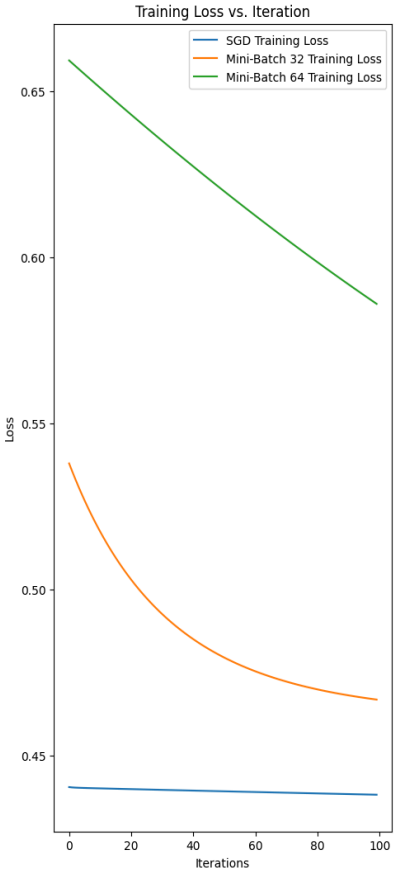
Min-Max Scaling normalizes feature values within a specific range-most often between 0 and 1-and its training process is typically characterized by a smooth loss decrease, while the accuracy goes up. Such normalization enables the model to converge better since all the features will contribute equally in the training process of the model.

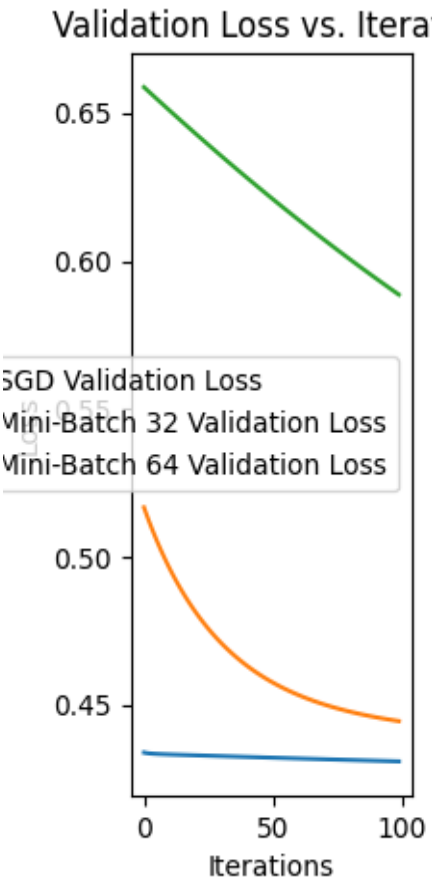
Still, for small learning rates, convergence with Min-Max Scaling remains slow; it takes up to 10,000 steps to achieve stable levels of loss and accuracy. This is because its slow update in the parameters leads to slow progress but often avoids the instability that larger learning rates may introduce. There is a risk, with enough iterations, that the performance could eventually drop below the model's performance curve minimum value, or worse, reach a point of diminishing return, at which it could be overfitting or show very minor returns for the further investment of time in training. In practice, one often monitors performance and likely adjusts the learning rate or performs early stopping to avoid such issues and optimize efficiency in training.

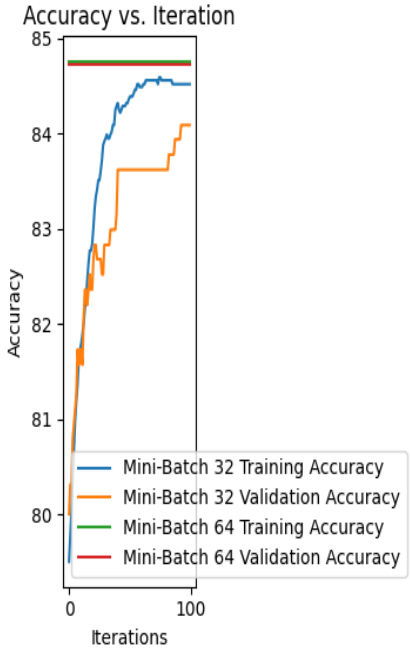


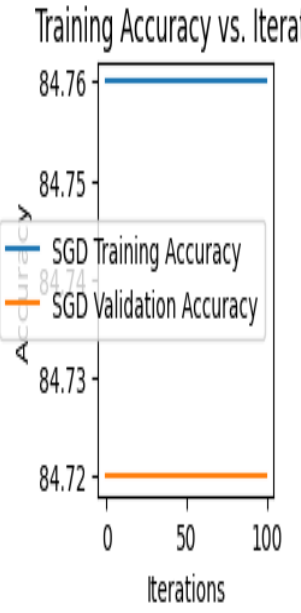
Precision: 0.42
Recall: 0.05
F1 Score: 0.09
ROC-AUC Score: 0.65

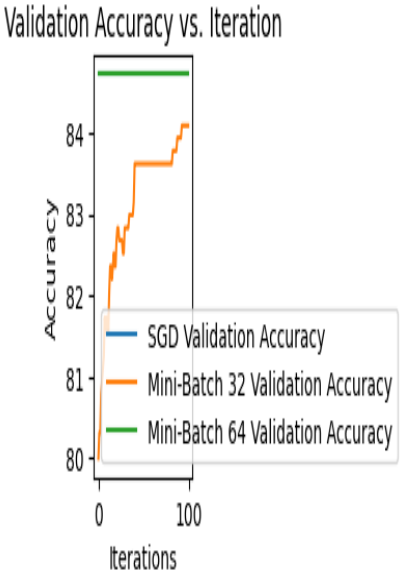
The big losses are reflected in very low precision, recall, and F1 score, each pointing to the model performing really poorly in detecting the positive class. While the ROC-AUC score shows some semblance of differentiating between the classes, on the whole, the general performance is insufficient for practical use, especially where the positive class is very important. This might take a little more feature engineering, algorithm choice, and tuning of hyperparameters to obtain a better balance between the classes, mainly in terms of the recall and precision of the positive class.


Training Loss vs. Iteration
SGD Training Loss
Mini-Batch 32 Training Loss
Mini-Batch 64 Training Loss
Loss
Iterations


Validation Loss vs. Iteration
SGD Validation Loss
Mini-Batch 32 Validation Loss
Mini-Batch 64 Validation Loss
Iterations


Accuracy vs. Iteration
Mini-Batch 32 Training Accuracy
Mini-Batch 32 Validation Accuracy
Mini-Batch 64 Training Accuracy
Mini-Batch 64 Validation Accuracy
Accuracy
Iterations


Training Accuracy vs. Iteration
SGD Training Accuracy
SGD Validation Accuracy
Iterations


Validation Accuracy vs. Iteration
SGD Validation Accuracy
Mini-Batch 32 Validation Accuracy
Mini-Batch 64 Validation Accuracy
Accuracy
Iterations

Whereas in the case of the mini-batch size, a batch of 32 exhibits a logarithmic decline in loss while converging in approximately 300 iterations with smooth, gradual improvements in accuracy. On the contrary, a batch size of 64 decreases linearly and converges in approximately 400 iterations with a steady increase in accuracy. While in SGD, this corresponds to a slow and almost constant decrease in the loss, meaning very slow convergence. It has very small improvements in training and validation accuracy. Mini-batch methods converge more effectively and stably than SGD. Larger batches result in more stable but slower improvement, while small batches often converge faster but may be less stable.

Fold 5 - Accuracy: 0.76

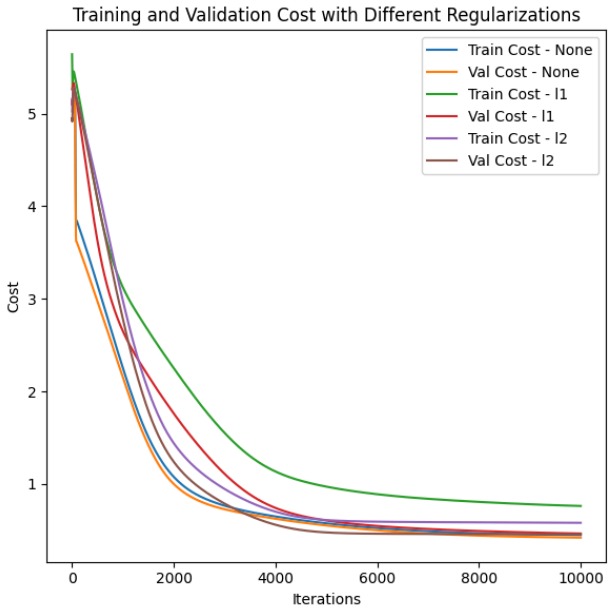
Average Accuracy: 0.76 ± 0.02

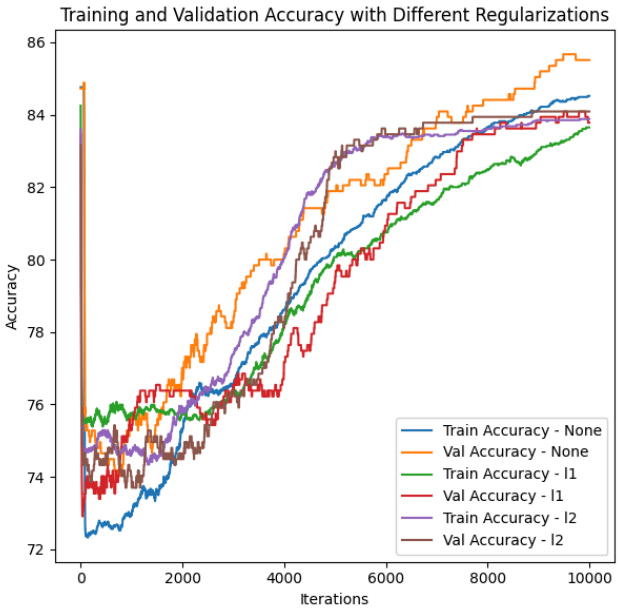
Average Precision: 0.20 ± 0.06

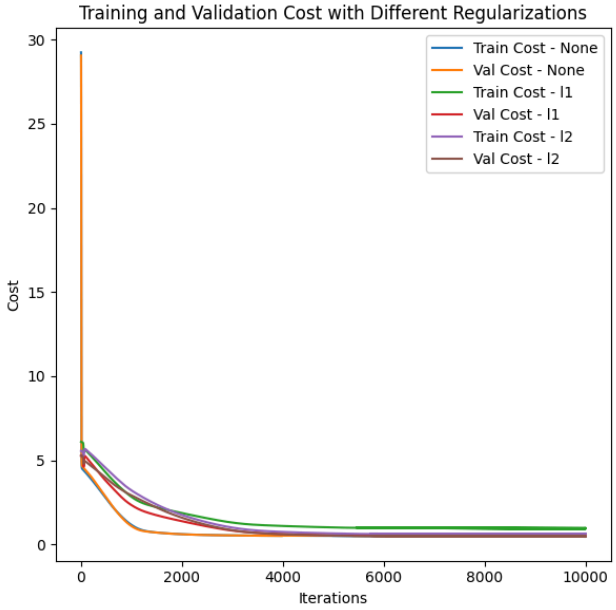
Average Recall: 0.20 ± 0.07

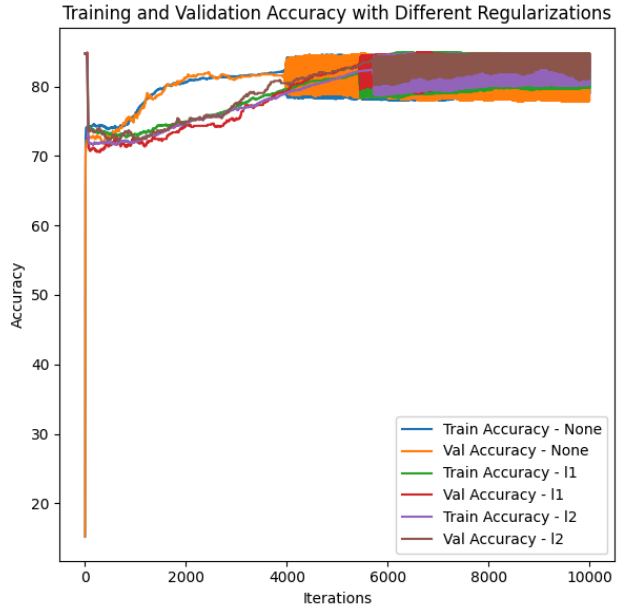
Average F1 Score: 0.20 ± 0.06

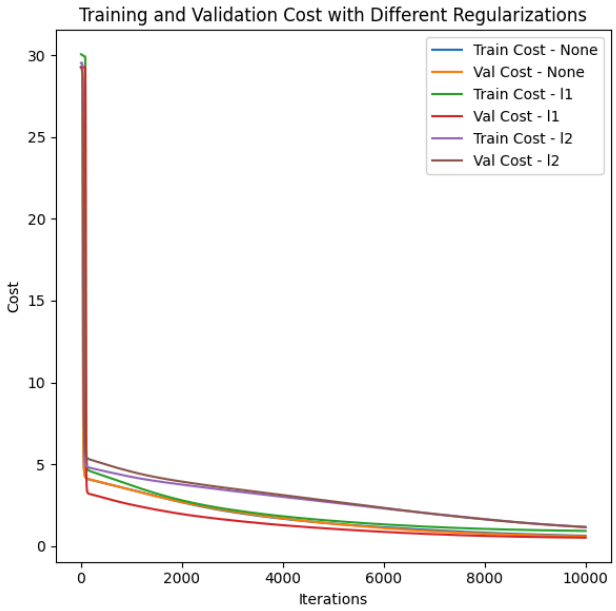
The average accuracy, precision, recall, and F1 score directly reflect the low dispersion between folds in the model's consistency. That implies that with low variance, the performance is dependable and consistent, whereas high variance may suggest the model performs differently each time, probably due to overfitting or sensitiveness to splits in the data. Stable results with low variance means the model generalizes well, and vice versa: with high variance, there is a potential issue that calls for further investigation.

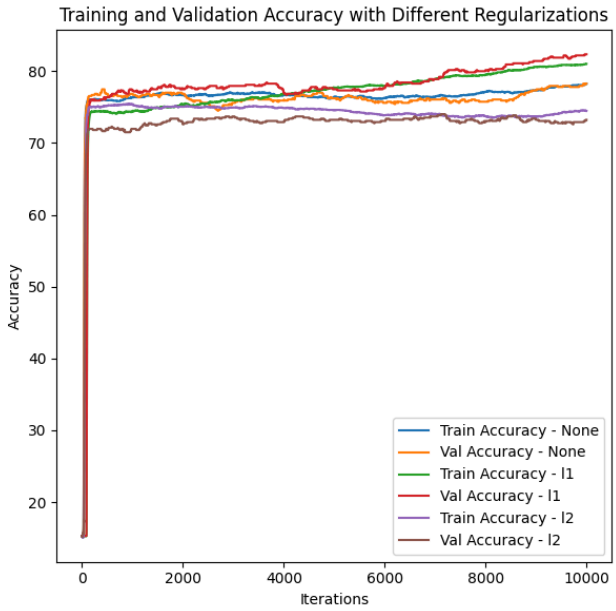

Training and Validation Cost with Different Regularizations
Train Cost - None
Val Cost - None
Train Cost - l1
Val Cost - l1
Train Cost - l2
Val Cost - l2
Cost
Iterations

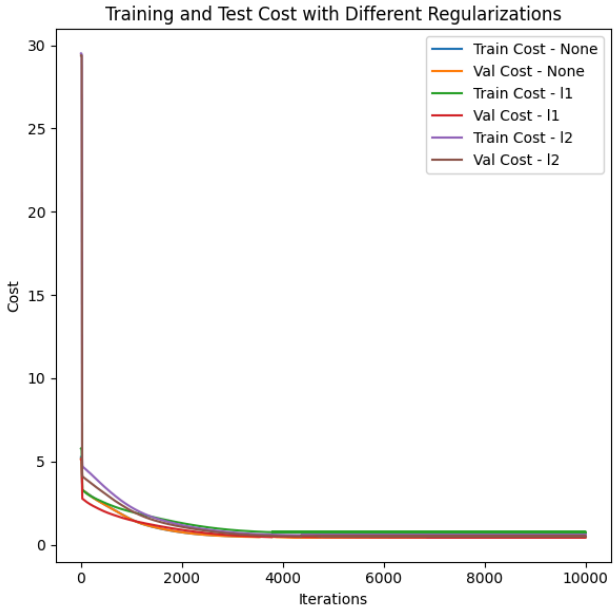

Training and Validation Accuracy with Different Regularizations
Train Accuracy - None
Val Accuracy - None
Train Accuracy - l1
Val Accuracy - l1
Train Accuracy - l2
Val Accuracy - l2
Accuracy
Iterations

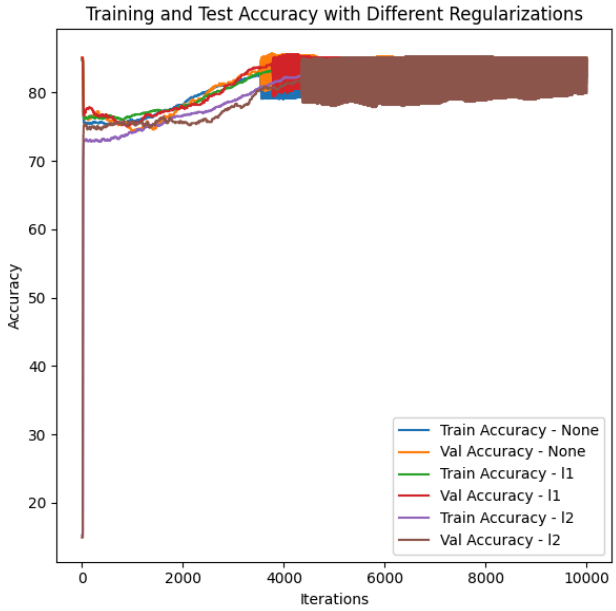

Training and Validation Cost with Different Regularizations
Train Cost - None
Val Cost - None
Train Cost - l1
Val Cost - l1
Train Cost - l2
Val Cost - l2
Cost
Iterations

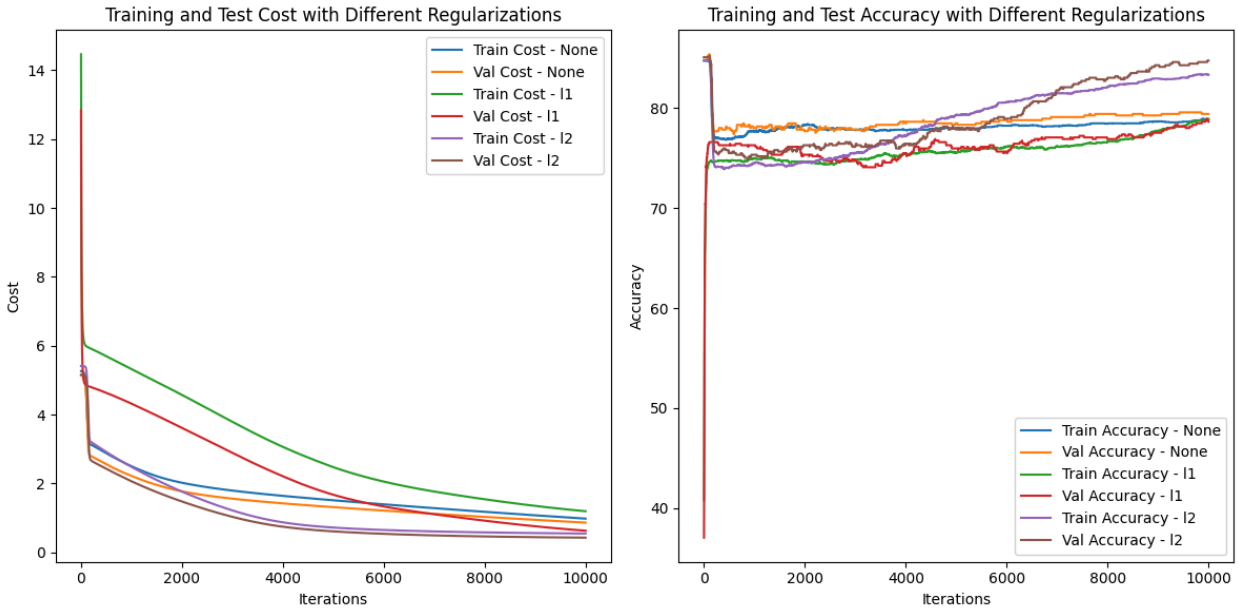

Training and Validation Accuracy with Different Regularizations
Train Accuracy - None
Val Accuracy - None
Train Accuracy - l1
Val Accuracy - l1
Train Accuracy - l2
Val Accuracy - l2
Accuracy
Iterations


Training and Validation Cost with Different Regularizations
Train Cost - None
Val Cost - None
Train Cost - l1
Val Cost - l1
Train Cost - l2
Val Cost - l2
Cost
Iterations


Training and Validation Accuracy with Different Regularizations
Train Accuracy - None
Val Accuracy - None
Train Accuracy - l1
Val Accuracy - l1
Train Accuracy - l2
Val Accuracy - l2
Accuracy
Iterations


Training and Test Cost with Different Regularizations
Train Cost - None
Val Cost - None
Train Cost - l1
Val Cost - l1
Train Cost - l2
Val Cost - l2
Cost
Iterations


Training and Test Accuracy with Different Regularizations
Train Accuracy - None
Val Accuracy - None
Train Accuracy - l1
Val Accuracy - l1
Train Accuracy - l2
Val Accuracy - l2
Accuracy
Iterations



Early stopping stops training if the validation cost doesn't improve for some fixed number of consecutive iterations. This prevents overfitting when the process is stopped even though the model is still improving on the training set but has started generalizing poorly.

Early stopping helps in avoiding overfitting when the model starts to degrade on a validation set even though it's still improving on the training set. This maintains a balance between the cost of training and validation, ensuring better generalization. Without early stopping, models often overfit for more extended periods, especially in situations where learning rates might be larger or regularization is too weak. Early stopping really works when it is combined with regularization methods (L1 or L2), as it now controls overfitting much more and keeps the validation performance stable. In conclusion, early stopping improves model performance by stopping the training at the optimal point for generalization.