CS6316 Machine Learning Application Project

Ye Ma(hjk6th), Tongxuan Tian(nua3jz), and Cheng Yu(rjv8hm)

Part (1) Traditional classification algorithms

1.1 Dataset Analysis

1.1.1 Dataset 1

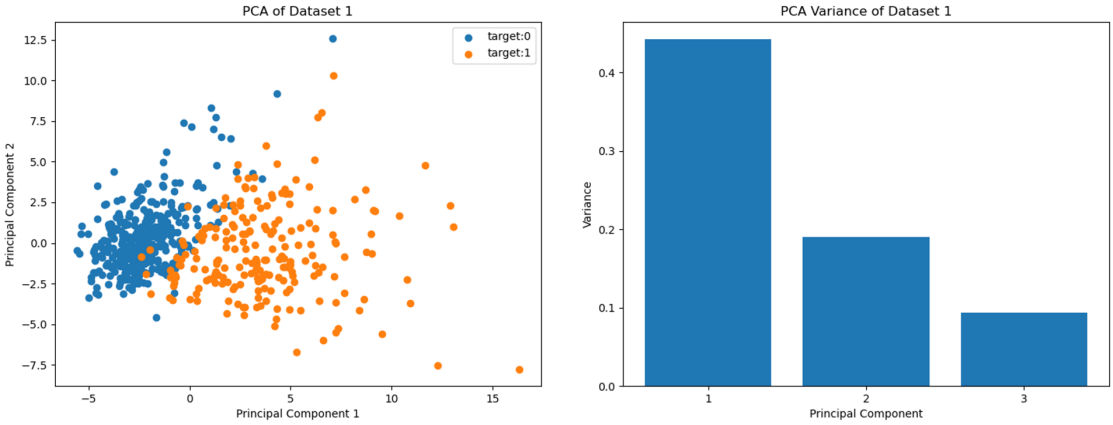


Fig. 1 Dataset 1 PCA

- There's a noticeable but not complete separation between the two classes along the first principal component (horizontal axis). This suggests that the first principal component captures a significant amount of the data's variance related to the class separation.
- The second principal component (vertical axis) also contributes to the separation, but to a lesser extent as indicated by the spread of the points.
- Some overlap between the classes is evident, suggesting that while PCA has revealed some structure in the data, the classes are not entirely separable in the reduced two-dimensional space.
- Overall, the visualization of pca results shows that dataset 1 has a clear pattern which should be easy to be captured by basic classification algorithms.

1.1.2 Dataset 2

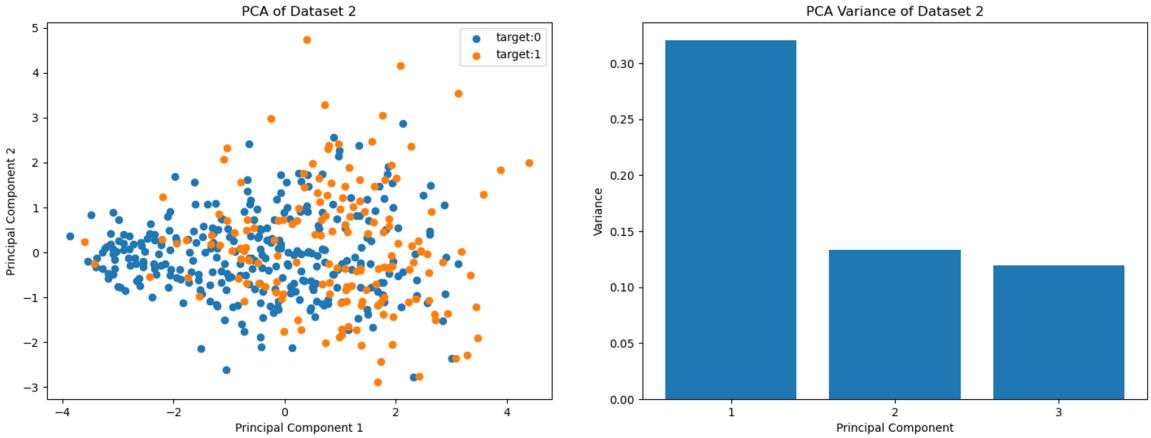


Fig. 2 Dataset 2 PCA

- Unlike the PCA of Dataset 1, the classes in Dataset 2 appear more intermixed, indicating that these two principal components do not distinctly separate the classes in the reduced-dimensional space.
- The relatively dense center where both classes are present suggests that the dataset may have a more complex structure that is not easily captured by linear dimensionality reduction.

1.2 Logistic Regression

1.2.1 Overview

Logistic regression algorithm models the probability of a binary response based on one or more predictor variables. In 10-fold cross validation, we will try different combinations of "*C*" which controls the strength of regularization applied to the model, "*max iter*" and "*solver*" to find the best parameter for each dataset. In performance analysis, we will also focus on how these 3 parameters will affect model performance.

1.2.2 10-fold Cross Validation Result

Parameter Search Space:

Parameter	Values	Best Value for Dataset 1	Best Value for Dataset 2
C	[0.1, 1, 10]	1	0.1
Max iter	[500, 1000, 2000]	2000	500
Solver	['lbfgs', 'sag']	'lbfgs'	'lbfgs'

Best Result on Dataset 1:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	95.25	95.24	91.94	93.50	94.56

Best Result on Dataset 2:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	71.19	61.51	48.75	53.25	65.93

- LR on Dataset 1 with high scores across all metrics on the training (CV score) and test sets, indicating a good fit. There's little difference between the CV and test scores, suggesting that the model generalizes well and is neither overfitting nor underfitting.
- The drop of Dataset 2 compared to Dataset 1 indicates that the model might be underfitting on Dataset 2, further supported by the lower test metrics. The dataset could be more challenging to model.

1.2.3 Performance Analysis

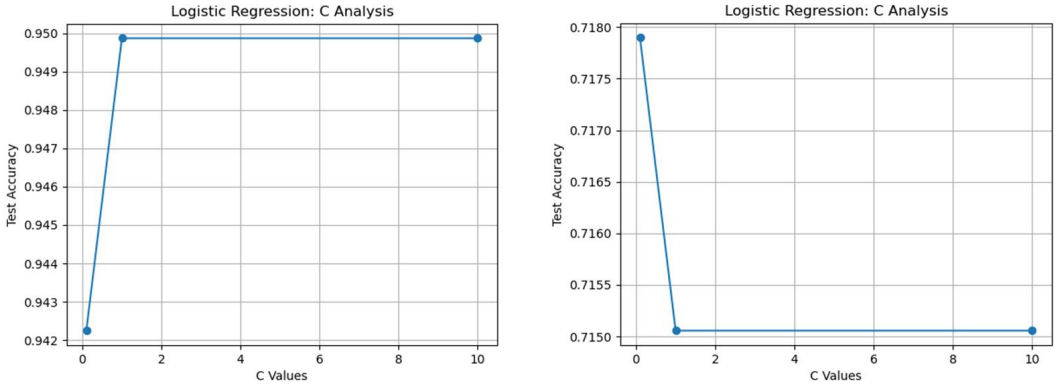


Fig. 3 C of LR varies, max_iter and solver on best param: left column- dataset1, right column- dataset2

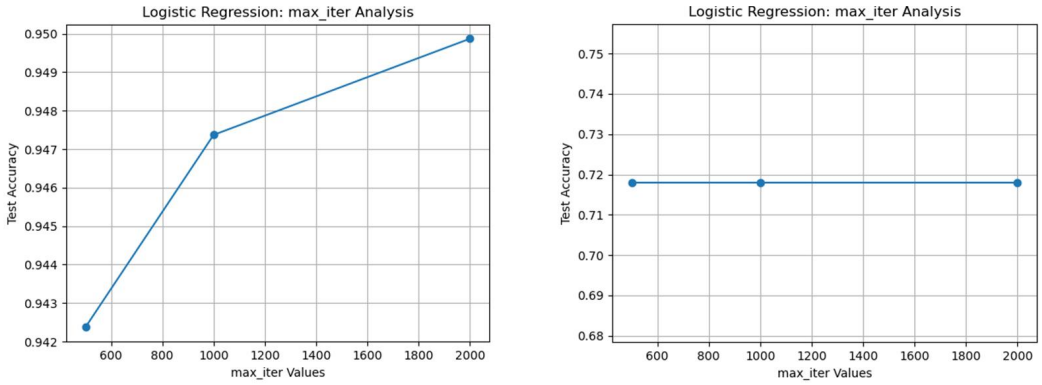


Fig. 4 max_iter of LR varies, C and solver on best param: left column- dataset1, right column- dataset2

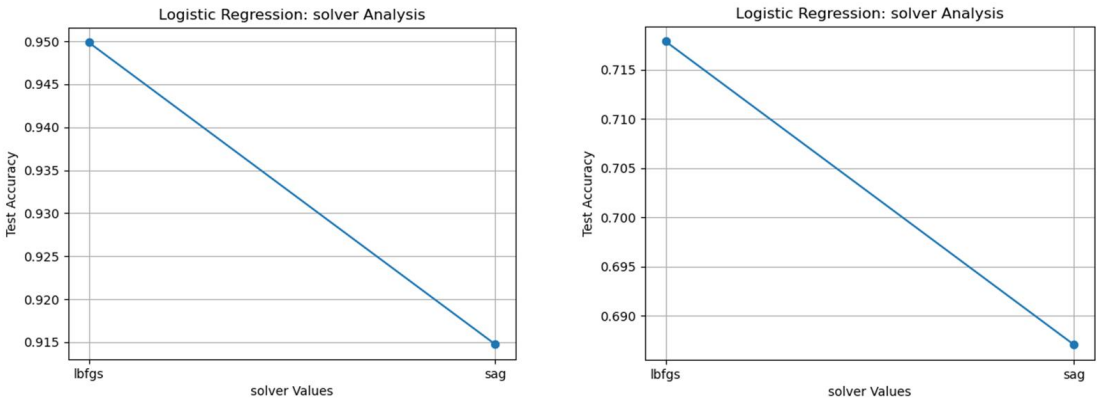


Fig. 5 solver of LR varies, C and max_iter on best param: left column- dataset1, right column- dataset2

LR:

- Dataset 1: The training and validation scores are quite close, with the validation score slightly lower than the training score as the complexity increases (higher C values). This indicates a good bias-variance tradeoff with the potential for overfitting at higher complexities.
- Dataset 2: A larger gap between the training and validation scores suggests the model may be overfitting. The best CV parameters have a lower C value, indicating that regularization helps reduce overfitting.

1.3 K Nearest Neighbor

1.3.1 Overview

The K Nearest Neighbor(KNN) algorithm operates by identifying the 'k' nearest data points to a given input and makes predictions based on these neighbors. In classification, KNN assigns the class most common among the k neighbors. In 10-fold cross validation, we will try different combinations of "*n_neighbor*" and "*weight*" to find the best parameter for each dataset. In performance analysis, we will also focus on how these 2 parameters will affect model performance.

1.3.2 10-fold Cross Validation Result

Parameter Search Space:

Parameter	Values	Best Value for Dataset 1	Best Value for Dataset 2
num of neighbors	[3, 5, 7]	3	7
weights	['uniform', 'distance']	'distance'	'uniform'

Best Result on Dataset 1:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	92.79	92.80	87.79	89.95	91.79

Best Result on Dataset 2:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	65.15	48.80	30.62	37.14	57.04

- KNN performs well on Dataset 1, with close training and testing scores, which suggests good generalization without significant overfitting or underfitting.
- The performance on Dataset 2 is much poorer, which could indicate underfitting, where the model is too simple to capture the data distribution.

1.3.3 Performance Analysis

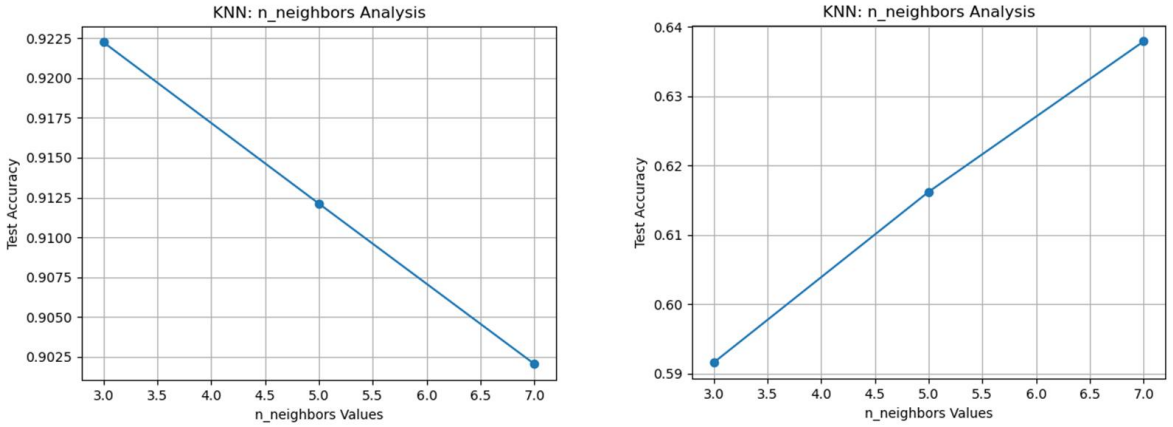


Fig. 6 n_neighbors of KNN varies, weights on best param: left column- dataset1, right column- dataset2

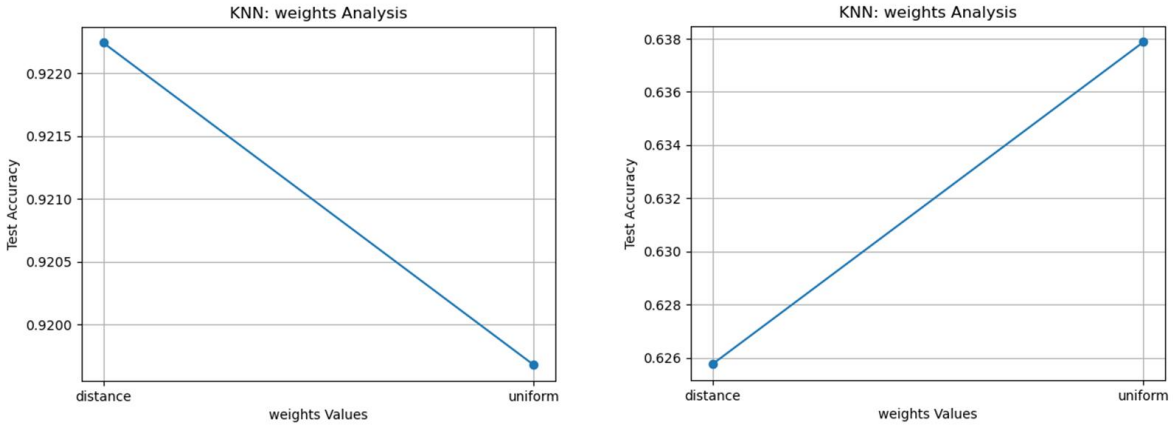


Fig. 7 weights of KNN varies, n_neighbors on best param: left column- dataset1, right column- dataset2

KNN:

- Dataset 1: The plot shows a high variance pattern, where the training score is almost perfect, but the validation score is significantly lower. The KNN model overfits the training data, especially with 'distance' weights, which give perfect training scores.
- Dataset 2: The KNN model still exhibits overfitting, but the variance is less extreme than in Dataset 1.

1.4 Decision Tree

1.4.1 Overview

Decision tree algorithm involves splitting data into branches at each node, based on feature values, leading to final decision leaves. The process relies on criteria like Gini impurity or information gain for splitting. In 10-fold cross validation, we will try different combinations of "*max depth*", "*min samples leaf*", "*min samples leaf*" and "*criterion*" to find the best parameter for each dataset. In performance analysis, we will also focus on how the first 3 parameters will affect model performance.

1.4.2 10-fold Cross Validation Result

Parameter Search Space:

Parameter (seed=123)	Values	Best Value for Dataset 1	Best Value for Dataset 2
Max Depth	[None, 10, 20, 30, 40, 50]	10	10
Min Samples Split	[2, 5, 10]	10	100
Min Samples Leaf	[1, 2, 4]	4	10
Criterion	['gini', 'entropy']	entropy	entropy

Best Result on Dataset 1:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	93.50	93.15	89.61	91.20	92.69

Best Result on Dataset 2:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	70.78	60.58	43.75	50.11	64.44

1.4.3 Performance Analysis

1.4.3.1 Max Depth

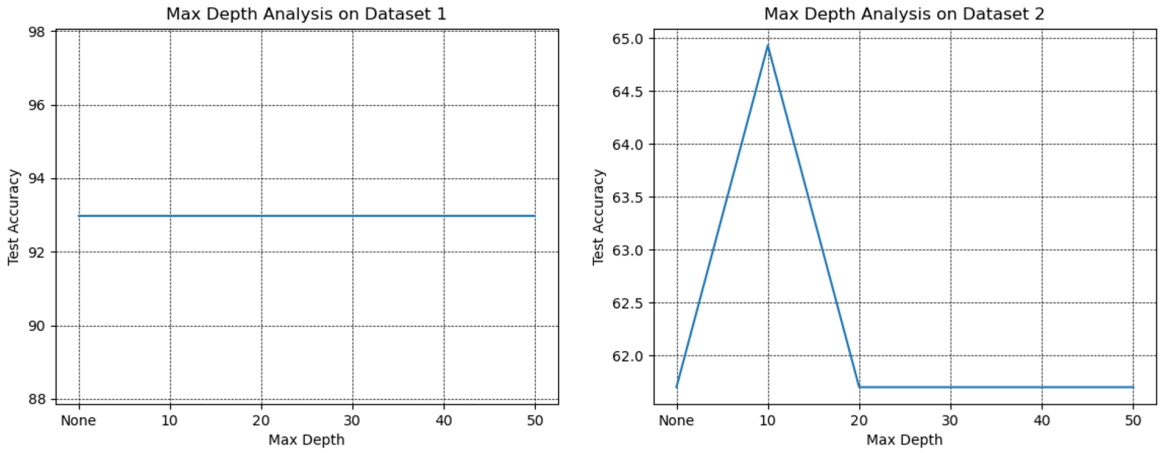


Fig. 8 Test accuracy under different *max depth*

In both datasets, we can see that max depth of decision tree has little effect on algorithm performance. For dataset 1, it is possibly because the dataset is too simple that the decision tree has already learned enough features to make decisions. While for dataset 2, it is possibly because dataset 2 is too complex like we visualized in section 1.1.1, and the feature for training is not strong enough for deeper trees to make correct predictions. And noise in dataset 2 is also a possible reason.

1.4.3.2 Min Samples Split

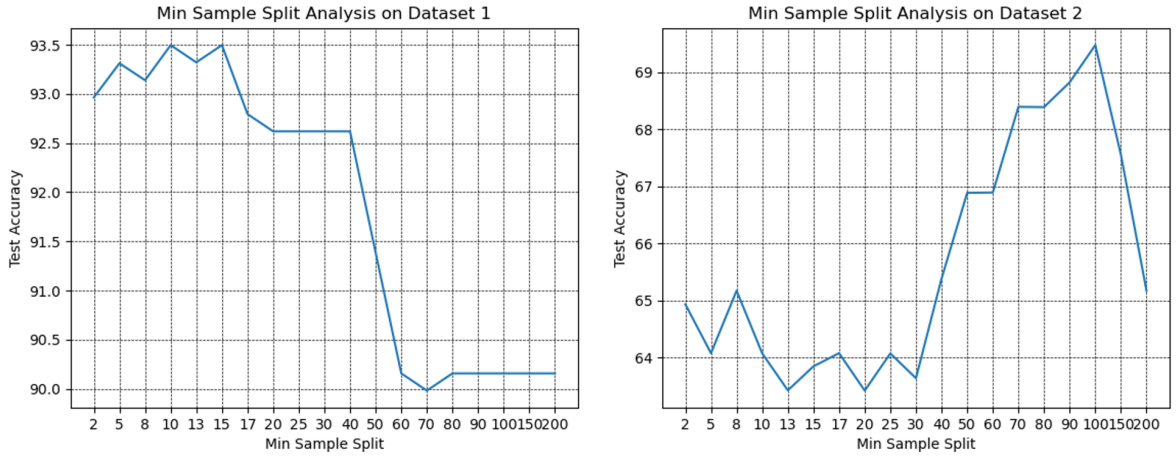


Fig. 9 The influence of *min sample split* parameter on decision tree

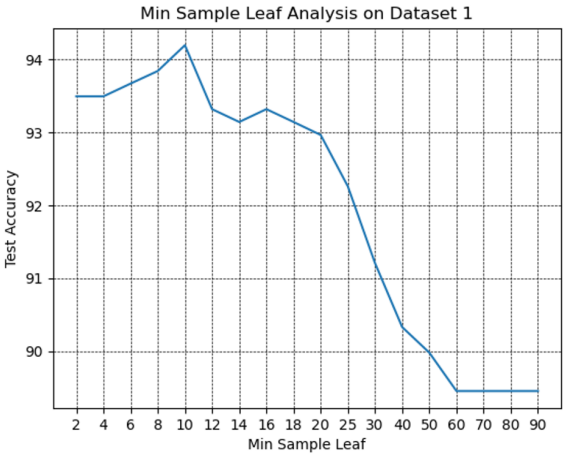
Dataset 1 Analysis:

- Initially, the model's accuracy is relatively high with a low *min_samples_split*, indicating that the model is capturing the data's patterns well. As *min_samples_split* increases, we observe a significant drop in accuracy, which then stabilizes. This sharp decrease could indicate that the model is overfitting the data; it's not capturing enough information due to the imposed restrictions on node splitting.
- A lower *min_samples_split* corresponds to a more complex model with potentially lower bias, as the decision tree can make more splits and thus fit the training data more closely.
- However, as *min_samples_split* increases, the model becomes less complex, leading to higher bias. The stability in test accuracy for higher values of *min_samples_split* suggests that variance is not a major issue for this dataset, but rather that the model is not capturing enough of the data's variance.

Dataset 2 Analysis:

- There is more fluctuation in test accuracy across different *min_samples_split* values, with a peak around a *min_samples_split* of 100. This peak suggests an optimal point where the model is neither too simple nor too complex for the data at hand. Beyond this point, test accuracy decreases dramatically, indicating that the model may be overfitting the data.
- The fluctuations in test accuracy for lower values of *min_samples_split* suggest a high variance in the model's predictions. At these values, the model may be fitting to noise rather than the underlying pattern, indicative of overfitting.
- At the optimal *min_samples_split* value (around 100), the model likely strikes the right balance between bias and variance, fitting the data well without capturing noise.
- When *min_samples_split* is too high, the variance decreases since the model is less sensitive to fluctuations in the training data. However, the bias increases because the model is too general, leading to overfitting。

1.4.3.3 Min Sample Leaf



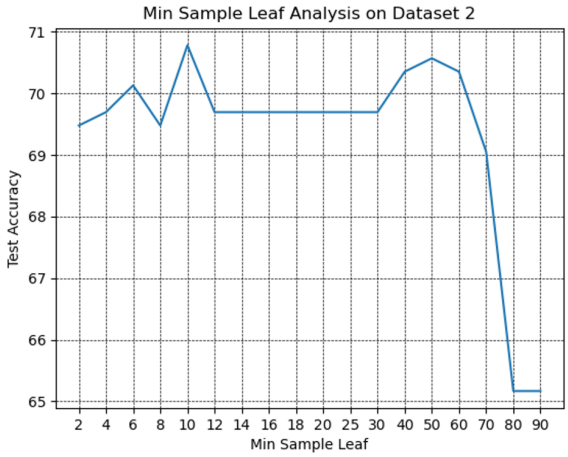


Fig. 10 Test accuracy under different *min sample leaf*

Dataset 1:

- There's a peak in accuracy at a certain point, followed by a decline as *min_samples_leaf* increases, suggesting that the tree might be too constrained and therefore underfitting the data at higher values of *min_samples_leaf*. Underfitting occurs because the decision tree is unable to capture the nuances of the data when each leaf is required to contain a large number of samples.
- At lower *min_samples_leaf* values, the decision tree has enough flexibility to potentially overfit the training data, but the high test accuracy indicates that this is not the case, and the model is likely well-calibrated.
- As *min_samples_leaf* increases, the bias in the model also increases, leading to underfitting. This is because the model becomes too generalized and is not capturing the detailed patterns in the data, leading to a simpler model with worse test performance.

Dataset 2:

- The sharp decrease after the peak indicates underfitting, where the decision tree is too simple to capture the complexity of the data, similar to Dataset 1.
- The initial fluctuation in test accuracy at low *min_samples_leaf* values suggests a higher variance, where the model might be sensitive to the specific noise in the training data, but the peak indicates the model is still performing well.
- As *min_samples_leaf* increases and accuracy drops, the model's variance is likely decreasing, but the bias is increasing because the model becomes too generalized, hence underfitting.

1.5 Support Vector Machine(SVM)

1.5.1 Overview

SVM works by finding the hyperplane that best separates different classes in the feature space. In classification, SVM looks for the hyperplane with the largest margin, meaning it tries to maximize the distance between the data points of different classes. For non-linearly separable data, SVM uses kernel tricks to transform the data into a higher dimension where a hyperplane can be used for separation. In 10-fold cross validation, we will try different combinations of "*C*" which controls the strength of regularization applied to models, and "*kernels*" which means the kernel we use, to find the best parameter for each dataset. In performance analysis, we will also focus on how the first 2 parameters will affect model performance.

1.5.2 10-fold Cross Validation Result

Parameter Search Space:

Parameter	Values	Best Value for Dataset 1	Best Value for Dataset 2
C	[0.1, 1, 10]	1	0.1
Kernels	['linear', 'rbf', 'poly']	'linear'	'linear'

Best Result on Dataset 1:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	95.26	95.4	91.96	93.56	94.58

Best Result on Dataset 2:

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	71.85	63.12	50	54.42	66.73

- SVM does better on Dataset 1 with similar training and testing scores, indicating a model that generalizes well.
- The performance is weaker on Dataset 2, which might indicate the dataset is inherently harder to model, leading to underfitting.

1.5.3 Performance Analysis

1.5.3.1 Regularization Analysis

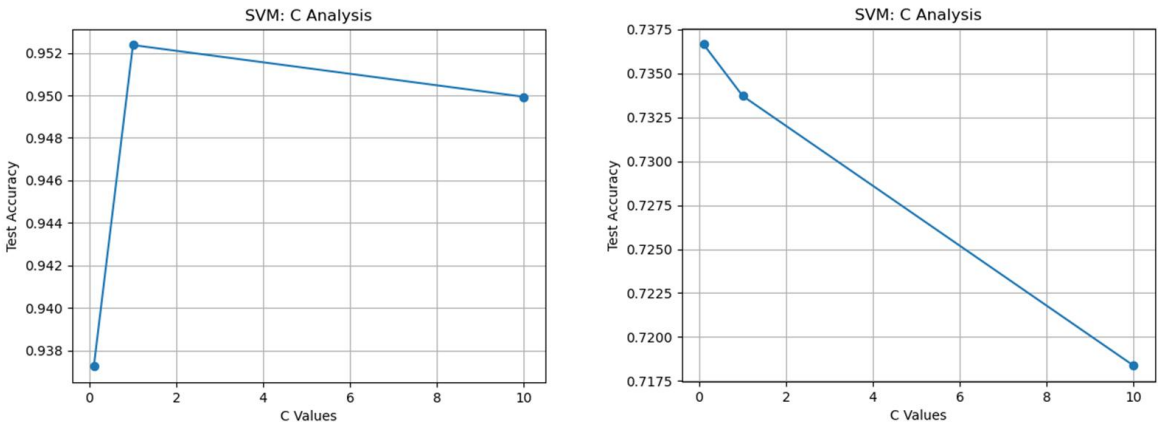


Fig. 11 C of SVM varies, kernels on best param: left column- dataset1, right column- dataset2

A high value of C means that the SVM algorithm will choose a smaller margin if that allows it to classify all training examples correctly. In other words, a high C puts emphasis on minimizing

the training error, which can lead to low bias but high variance, making the model prone to overfitting.

A low value of C allows for a larger margin and therefore more misclassifications on the training data. This implies that the model is aiming for a simpler decision boundary and is more tolerant of errors on the training set. A low C typically means higher bias but lower variance, potentially underfitting the model if C is too low.

1.5.3.2 Kernel Analysis

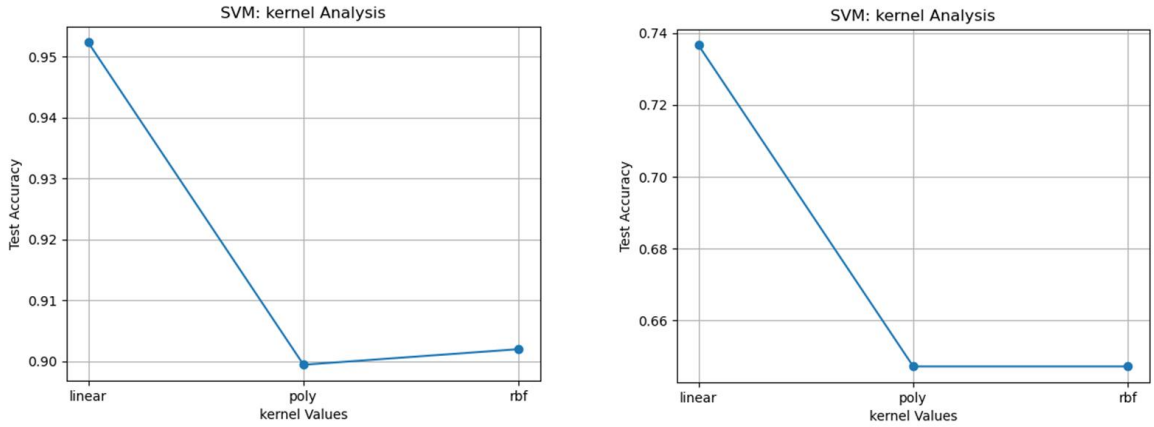


Fig. 12 kernels of SVM varies, C on best param: left column- dataset1, right column- dataset2

SVM:

- Dataset 1: The SVM shows a good balance with high training and validation scores on Dataset 1. However, the larger gaps between training and validation scores indicate signs of overfitting with the 'rbf' kernel and higher C values.
- Dataset 2: Similar to LR and KNN, the SVM shows a higher variance on Dataset 2, with a wider gap between training and validation scores, especially at higher complexities.

1.6 Random Forest

1.6.1 Overview

Random forest is an ensemble machine learning algorithm that combines multiple decision trees to improve prediction accuracy and stability. In 10-fold cross validation, we will try different combinations of "*number of estimators*", "*max depth*", "*min samples split*" and "*max feature option*" to find the best parameter for each dataset. In performance analysis, we will also focus on how the first 3 parameters will affect model performance.

1.6.2 10-fold Cross Validation Result

Parameter Search Space

Parameter (seed=123)	Values	Best Value for Dataset 1	Best Value for Dataset 2
Num of Estimators	[10, 30, 50, 80, 100]	100	80
Max Features	['sqrt', 'log2', None]	None	sqrt
Max Depth	[None, 10, 20, 30]	30	30
Min Samples Split	[2, 3, 5, 8, 10]	3	70

Best Result on Dataset1

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	96.83	97.16	94.32	95.69	96.32

Best Result on Dataset2

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	71.86	63.54	46.25	52.46	65.86

1.6.3 Performance Analysis

1.6.3.1 Num of Estimators

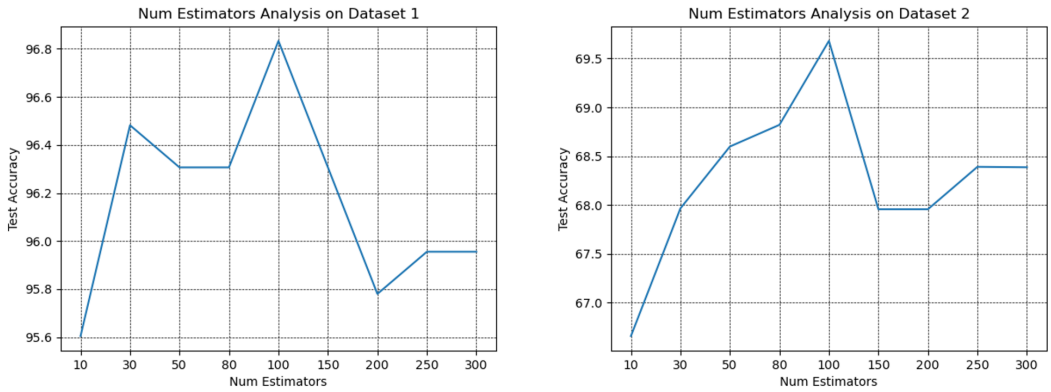


Fig. 13 Test accuracy under different *num estimators*

Dataset 1 Analysis:

- The test accuracy fluctuates with the increase in the number of estimators, showing a significant variance in performance.
- A peak in accuracy is visible around 100 estimators, suggesting that this might be an optimal range for the number of trees, balancing bias and variance effectively.

- The drop in accuracy beyond 100 estimators, followed by a rise and then another drop, indicates that increasing the number of trees beyond a certain point introduces more variance without necessarily reducing bias.
- High accuracy scores near 96% suggest that bias is generally low across the board for this dataset.

Dataset 2 Analysis:

- The overall test accuracy is much lower, which may indicate a higher bias in the model for Dataset 2. This could be due to the dataset being more complex or the Random Forest model not being well-tuned for the patterns in the data.
- There is an increase in test accuracy as the number of estimators grows up to around 150, after which the accuracy notably decreases. This suggests that up to 150 estimators, the model may be reducing bias.
- The sharp drop in accuracy after 150 estimators could be due to an increase in variance, where the additional trees are fitting to noise rather than the underlying data pattern.

1.6.3.2 Max Depth

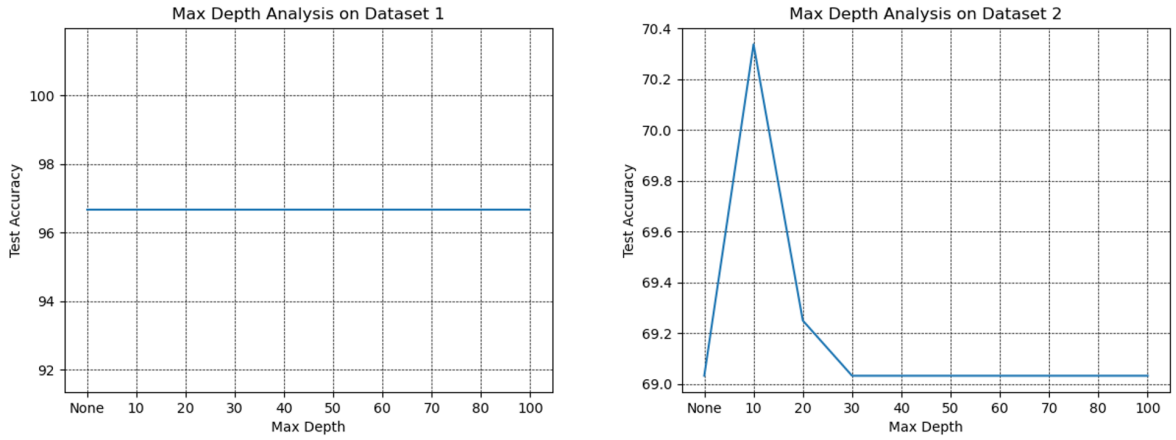


Fig. 14 Test accuracy under different *max depth*

Dataset 1 Analysis:

- The test accuracy is relatively constant regardless of the max_depth value, including when there is no maximum depth set (indicated as *None*). This could imply that the dataset is either not very complex or the Random Forest model has sufficient predictive power to handle the data complexity even with shallower trees.
- The lack of a significant decrease in accuracy at higher depths suggests that overfitting is not a substantial issue for this dataset with respect to the model complexity controlled by *max_depth*.

- The stability of accuracy across different depths indicates a low variance in model predictions with changing *max_depth*.

Dataset 2 Analysis:

- There is a sharp peak in accuracy when *max_depth* is set around 10, followed by a rapid decrease in performance as the depth increases.
- The peak at a lower *max_depth* suggests that a more shallow tree is better able to generalize unseen data for Dataset 2. A shallow tree captures enough of the data structure to make accurate predictions without capturing noise, which would be the case with deeper trees.
- The decrease in accuracy beyond a *max_depth* of 10 indicates that the model begins to overfit. As trees are allowed to grow deeper, they likely start to fit idiosyncrasies in the training data that do not generalize well to the test data, leading to higher variance.
- The fact that the accuracy drops below the level observed with *max_depth=None* after the peak suggests that too much complexity in the model is detrimental for Dataset 2, reinforcing the presence of overfitting with increased depth.

1.6.3.3 Min Samples Split

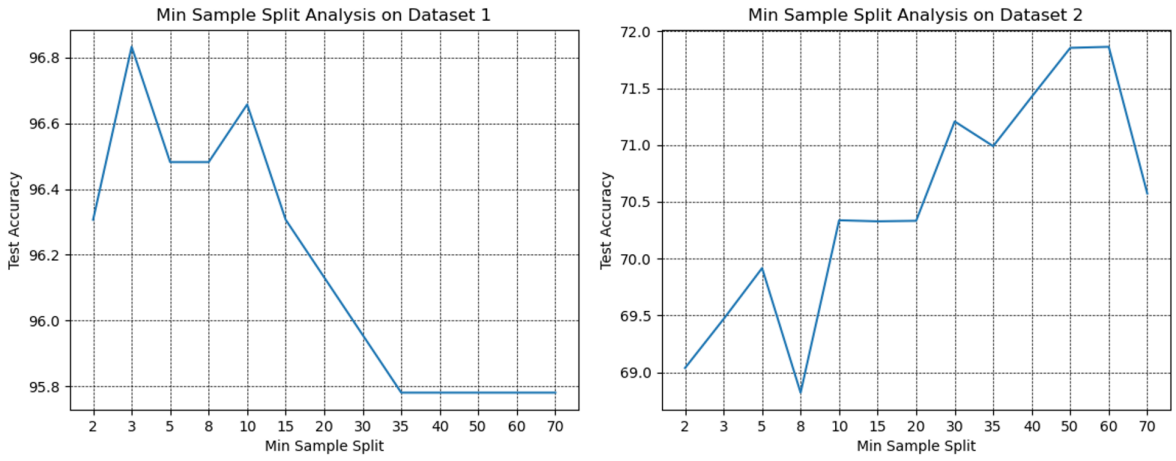


Fig. 15 Test accuracy under different *min sample split*

Dataset 1:

- The test accuracy remains relatively high and stable as *min_samples_split* increases from 2 to around 10, and then there's a noticeable downward trend as *min_samples_split* continues to increase.
- The stable high accuracy at lower *min_samples_split* values suggests that the model is not overfitting significantly, as it maintains good generalization to the test data.

- The decline in accuracy with larger *min_samples_split* values indicates underfitting, where the model becomes too conservative, leading to a loss of predictive power as it cannot capture the underlying structure of the data adequately.
- Initially, the bias is likely low as the model can split on a small number of samples, allowing it to capture more complex patterns. However, as *min_samples_split* increases, the model's ability to capture this complexity is reduced, leading to increased bias.
- The variance is likely not significantly impacted at lower *min_samples_split* values given the stability of the accuracy. However, as *min_samples_split* increases, the reduction in model complexity can lead to decreased variance but at the cost of increased bias.

Dataset 2:

- Test accuracy peaks sharply at a *min_samples_split* of around 50 and then decreases, suggesting that there is a specific range where the model best generalizes to the test data.
- The sharp increase to the peak might suggest that at lower *min_samples_split* values, the model could be slightly overfitting—capturing noise rather than underlying patterns. However, given the magnitude of the peak, this might also reflect a dataset-specific optimal point for this parameter.
- The drop in accuracy as *min_samples_split* increases beyond this peak suggests that the model starts to underfit, becoming too generalized and unable to capture necessary details for accurate predictions.
- The variance appears to be high at lower *min_samples_split* values, as suggested by the substantial changes in test accuracy. The peak indicates a balance between bias and variance.
- As *min_samples_split* increases beyond the optimal range, the model's bias increases, leading to a loss of accuracy due to underfitting, as the model cannot make as many splits and thus captures less detail

1.7 Boosting

1.7.1 Overview

We will choose adaboost here in our report. AdaBoost combines multiple weak learners, typically simple decision trees, to create a strong classifier. The algorithm iteratively trains weak learners, adjusting the weights of incorrectly classified instances so that subsequent learners focus more on difficult cases. In each round, a weak learner is added to the ensemble and weighted based on its accuracy. The final model is a weighted sum of these weak learners. In 10-fold cross validation, we will try different combinations of "*number of estimators*", "*learning rate*" and "*base estimator*" to find the best parameter for each dataset. In performance analysis, we will analyze the bias-variance trade-off in terms of the log loss of adaboost in each round by controlling the "*number of estimators*" and "base estimator" parameters.

1.7.2 10-fold Cross Validation Result

Parameter Search Space

Parameter (seed=123)	Values	Best Value for Dataset 1	Best Value for Dataset 2
Num of Estimators	[50, 100, 150, 200]	150	80
Learning Rate	[0.01, 0.1, 1]	1	0.01
Base Estimator	[Decision Tree, SVM, GuassianNB]	Decision Tree	Decision Tree

Best Result on Dataset1

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	97.01	98.14	93.85	95.88	96.36

Best Result on Dataset2

Metric	Accuracy	Precision	Recall	F1 Score	AUC
Value(%)	73.15	65.79	48.75	55.09	67.44

1.7.3 Performance Analysis

1.7.3.1 Number of estimators

Since adaboost will train a new weak learner in each iteration, we can analyze the overfitting and underfitting of adaboost by controlling the "n_estimators" parameter, which represents the number of weak learners(base estimator).

Overfitting Analysis

We set *n_estimators=400* and observed overfitting. The overfitting graph indicates that the training loss decreases sharply and remains close to zero after a small number of iterations, which suggests that the model has fit the training data very well. However, the test loss shows high variability and, overall, it remains significantly above the training loss throughout the training process. This behavior is characteristic of overfitting, where the model learns the training data too well, including the noise, making it less generalizable to unseen data. And from the perspective of bias-variance trade-off, we have the conclusions below:

- Low Bias: The training loss approaches zero quickly, suggesting that the model's predictions are very close to the true outcomes in the training data.
- High Variance: The test loss demonstrates considerable fluctuations and does not converge to a low value, implying that the model's performance is highly sensitive to the particular data it was trained on.

In essence, the model has a low bias but high variance, fitting the training data too well to the point of capturing noise, which detrimentally affects its performance on the test data.

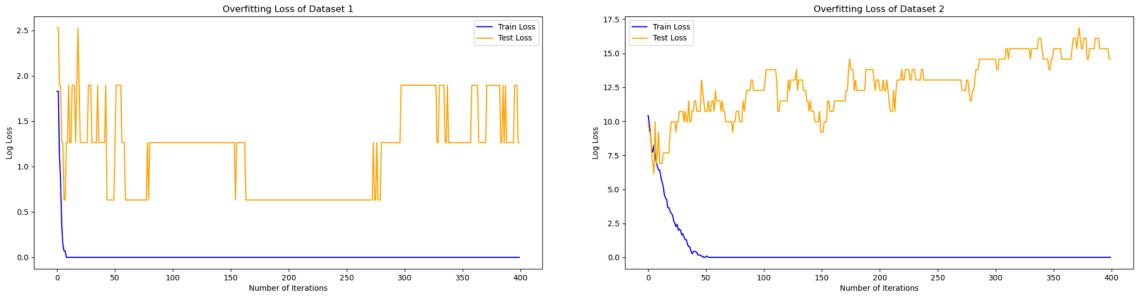


Fig. 16 Train and test loss at each iteration

Underfitting Analysis

We set *n_estimators=10* and observed underfitting. In the underfitting graph, both training and test losses start high. The training loss decreases and somewhat stabilizes at a value that is still significantly higher than what is observed in the overfitting scenario. The test loss follows a similar downward trend but does not converge closely with the training loss, suggesting the model has not learned the underlying patterns in the data sufficiently. And from the perspective of bias-variance trade-off, we have the conclusion below:

- High Bias: Both training and test losses are relatively high, indicating that the model's predictions are consistently far from the true outcomes, even in the training data.
- Low Variance: The test loss, while still higher than the training loss, does not show significant fluctuations, suggesting that the model's predictions are not sensitive to the training data it was trained on.

In this scenario, the model has high bias and low variance, meaning it is not complex enough to capture the underlying patterns and structures in the data.

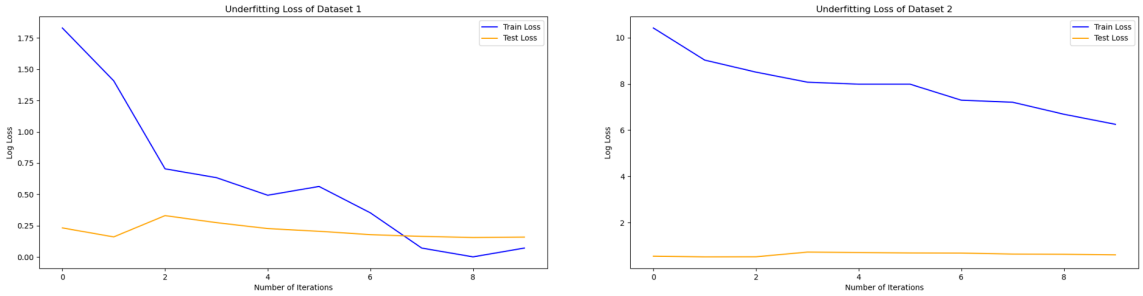


Fig. 17 Train and test loss at each iteration

1.7.3.2 Base Estimator Analysis

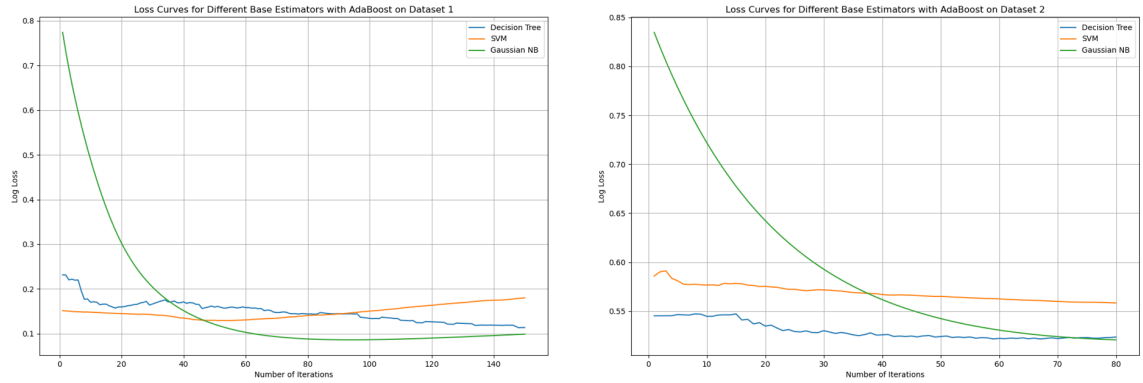


Fig. 18 Train and test loss at each iteration of different *base estimator*

Dataset 1 Analysis:

- Decision Tree: The rapid decline and stabilization of the loss curve suggest a low bias as the model appears to fit the training data well. However, without seeing corresponding variance data, it is challenging to determine if the model suffers from high variance. If test data loss were similarly low, it would indicate a well-balanced model.
- SVM: The SVM's gradual loss decline and higher stabilization point indicate a slightly higher bias compared to the Decision Tree, possibly due to an inability to capture all the nuances in the training data. A gentle slope of the loss curve can also mean that the variance is not excessive, assuming the test loss follows a similar pattern.
- Gaussian NB: Gaussian NB's curve starts with a lower loss, decreasing rapidly to a minimal value, which implies low bias. The quick stabilization also suggests low variance, assuming the model's performance on unseen data is consistent with these results.

Dataset 2 Analysis

- Decision Tree: Similar behavior to Dataset 1 with a quick loss stabilization indicates a low bias. The model is consistent across different datasets, which may suggest a good balance between bias and variance.
- SVM: The higher and more gradual loss reduction could imply a higher bias, and without overfitting indications, it may also suggest moderate variance. However, the model may not be as well-adjusted to the dataset characteristics as the Decision Tree or Gaussian NB.
- Gaussian NB: The consistent performance with a low loss value across datasets suggests that Gaussian NB maintains a low bias and variance, indicating an excellent balance in the bias-variance trade-off.

1.8 Model Comparison

1.8.1 Comparison on Dataset 1

Algorithm	Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	AUC(%)
Logistic Regression	95.25	95.24	91.94	93.50	94.56
KNN	92.79	92.80	87.79	89.95	91.79
Decision Tree	93.50	93.15	89.61	91.20	92.69
SVM	95.26	95.4	91.96	93.56	94.58
Random Forest	96.83	97.16	94.32	95.69	96.32
AdaBoost	97.01	98.14	93.85	95.88	96.36

On dataset 1, since dataset 1 is a rather simple dataset as we visualized in section 1.1.1, all the algorithms achieve good test accuracy. And adaboost is the algorithm that has the best performance with 97.01% accuracy on dataset 1.

1.8.2 Comparison on Dataset 2

Algorithm	Accuracy(%)	Precision(%)	Recall(%)	F1 Score(%)	AUC(%)
Logistic Regression	71.19	61.51	48.75	53.25	65.93
KNN	65.15	48.80	30.62	37.14	57.04
Decision Tree	70.78	60.58	43.75	50.11	64.44

SVM	71.85	63.12	50	54.42	66.73
Random Forest	71.86	63.54	46.25	52.46	65.86
AdaBoost	73.15	65.79	48.75	55.09	67.44

On dataset 2, due to the complexity and possible noise of dataset 2, all the algorithms don't have very high performance. Similarly, adaboost still ranks top-1 among all the algorithms, which is possibly because adaboost is an ensemble technique that combines multiple weak classifiers to create a strong classifier. Hence, it has better flexibility and generalization ability than other algorithms, which leads to its better performance.

Part (2) Neural Network

2.1 Implementation

2.1.1 Model Structure

As Fig. 19 shows, our implemented multilayer perceptron (MLP) model has an input layer with 784 (size of each MNIST image: 28x28) neurons, a sigmoid hidden layer with 128 neurons, a sigmoid hidden layer with 64 neurons, and a softmax output layer for 10 classes (MNIST labels: 0~9).

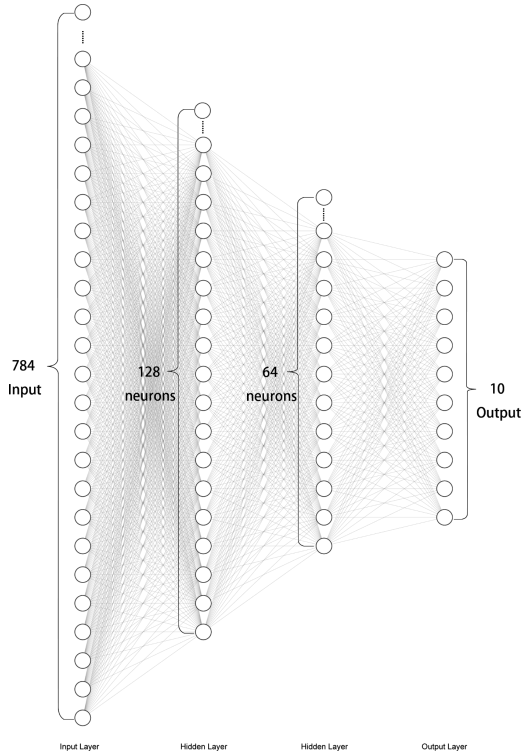


Fig. 19 Model Structure

2.2 Classification Results on MNIST

We trained the MLP model with MNIST training set for 5, 10, …, 45, 50 epochs (10 different epochs in total), and then tested the model's performance based on four evaluation metrics: Accuracy, Precision, Recall, and F1 score. As shown in Fig. 20, the values of four evaluation metrics on the MNIST testing set increase as the number of epochs increases (the 4 metrics of 5 epochs are 0.7389, 0.5990, 0.7265, and 0.6545 respectively while the 4 metrics of 50 epochs reach 0.9535, 0.9530, 0.9529 and 0.9529).

Epochs	**Accuracy**	**Precision**	**Recall**	**F1**
5	0.7389	0.5990	0.7265	0.6545
…	…	…	…	…
50	0.9535	0.9530	0.9529	0.9529

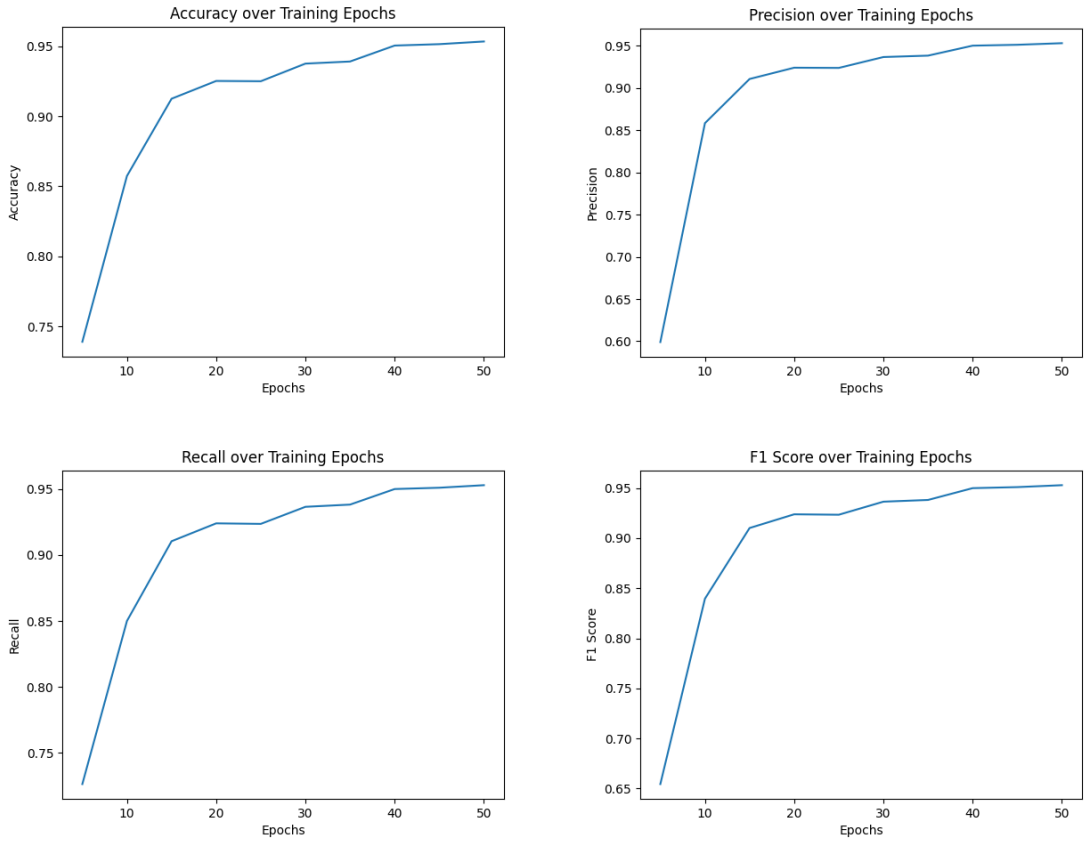


Fig. 20 Classification results over training epochs

2.3 Whether Hyperparameters Affect Model Performance

2.3.1 Number of Hidden Units

Our original MLP model has 192(128+64) hidden neurons in total, here we implemented a new MLP model which has 384(256+128) hidden neurons as Fig. 21 shows. We trained and tested the new model following the same procedures as section 2.2.

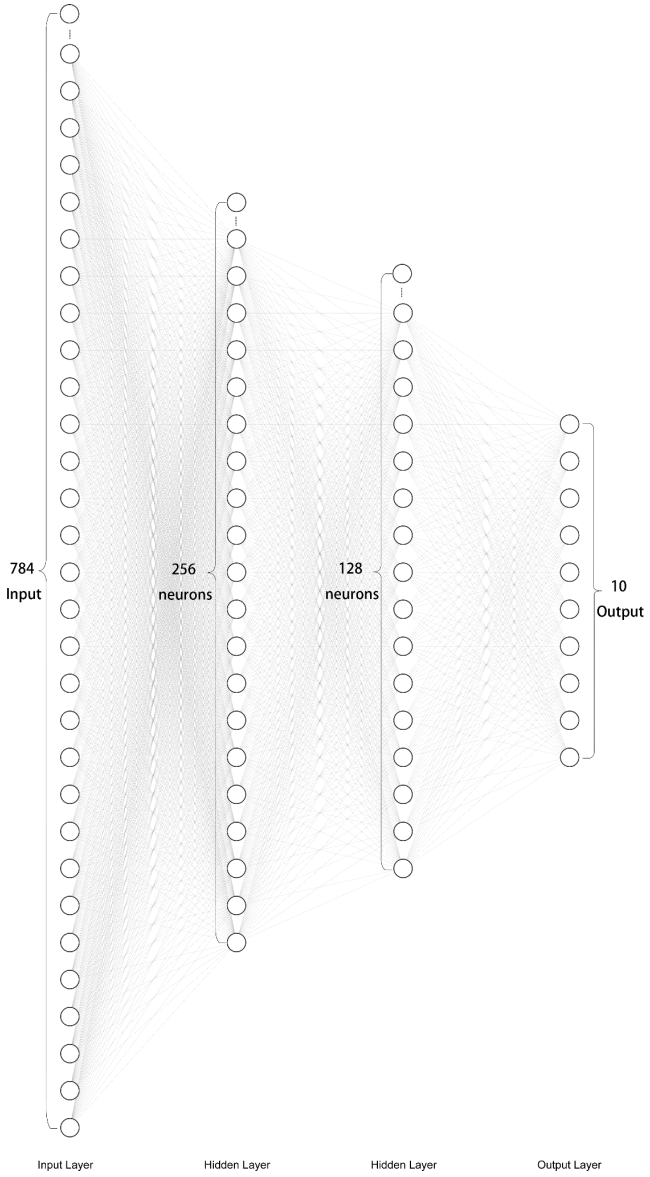


Fig. 21 Model Structure

As shown in Fig. 22, the values of four evaluation metrics of the model with 384 neurons are higher than those of the model with 192 neurons in all different epochs. For example, the 4 metrics of the new model trained in 5 epochs are 0.8220, 0.7442, 0.8115, and 0.7750 respectively

while those of the original model trained in 5 epochs are only 0.7389, 0.5990, 0.7265, and 0.6545 respectively.

Epochs	Accuracy	Precision	Recall	F1
5	0.8220	0.7442	0.8115	0.7750
…	…	…	…	…
50	0.9616	0.9614	0.9612	0.9612

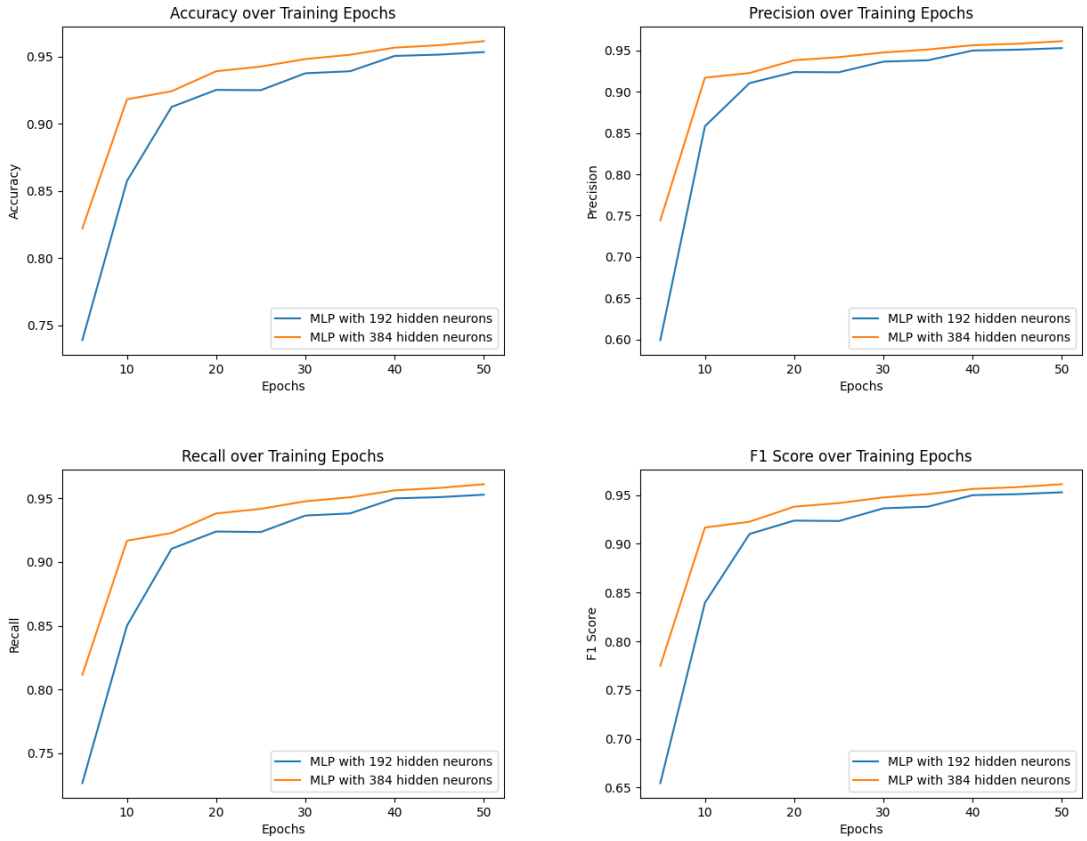


Fig. 22 Classification results over epochs

2.3.2 Weight/Bias Initialization

Our default model applies xavier_uniform weight initialization and zeros bias initialization. For this experiment, we implemented 5 new MLP models that applied different weight/bias initialization as the following table shows, all these new models were trained in 10 epochs.

Model	Weight Initialization	Bias Initialization
default	xavier_uniform	zeros

2	xavier_normal	zeros
3	kaiming_uniform	zeros
4	kaiming_normal	zeros
5	xavier_uniform	uniform
6	xavier_uniform	normal

As Fig. 23 shows, model 4 has the highest values of the 4 evaluation metrics among all models, which applies kaiming_normal weight initialization and zeros bias initialization.

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**
default	0.8574	0.8583	0.8501	0.8396
2	0.7984	0.7245	0.7894	0.7478
3	0.8871	0.8848	0.8831	0.8813
4	0.8980	0.8958	0.8958	0.8952
5	0.8372	0.7606	0.8268	0.7903
6	0.8389	0.7603	0.8287	0.7913

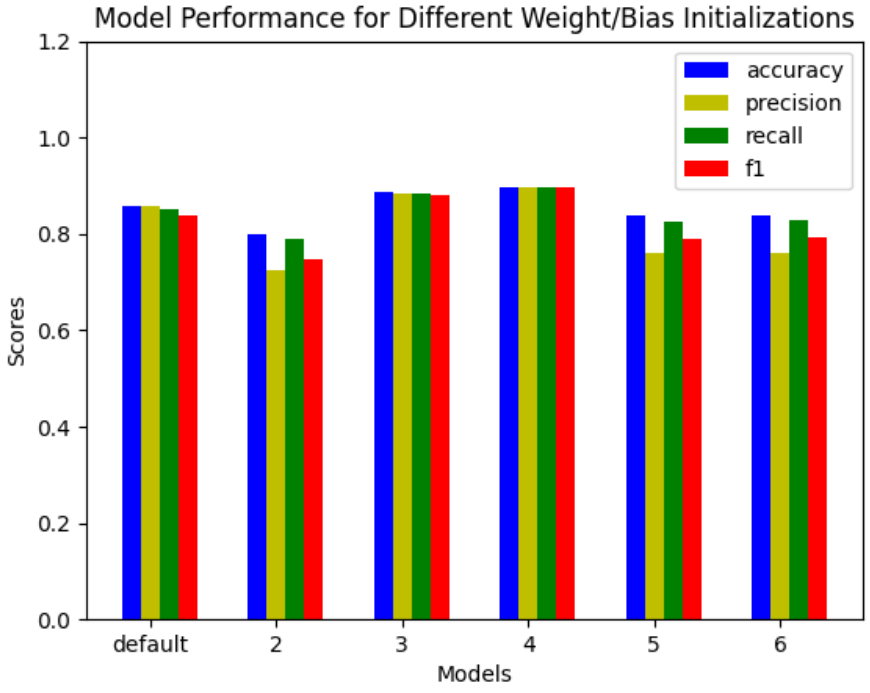


Fig. 23 Performance of different models

2.3.3 Learning Rate

For this experiment, we tested 5 different learning rates: 0.0001, 0.0005, 0.001, 0.005, 0.01, and the model was also trained in 10 epochs. Fig. 24 shows the training loss across epochs for different learning rates, we can see that the training loss curve of learning rate 0.0001 has the largest amplitude, and the training loss curve of learning rate 0.01 has the smallest amplitude.

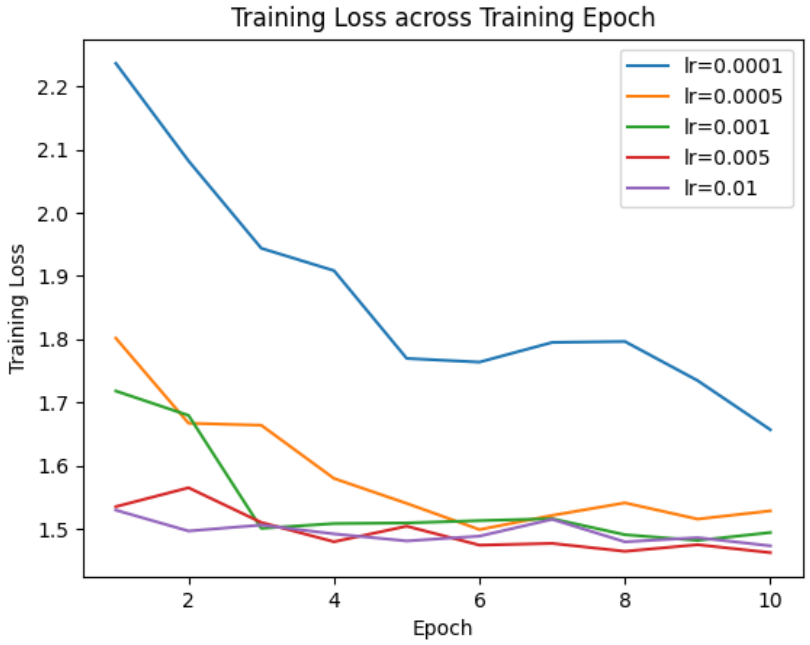


Fig. 24 Training loss across epochs

As shown in Fig. 25, the values of 4 evaluation metrics of MLP model surge from 0.8351, 0.7582, 0.8246, and 0.7884 to 0.9556, 0.9553, 0.9550, and 0.9551 when learning rate increases from 0.0001 to 0.0005, and the values of metrics between 0.0005 and 0.01 are similar (all metrics values reach their maximum value when learning rate is 0.005).

Learning Rate	**Accuracy**	**Precision**	**Recall**	**F1**
0.0001	0.8351	0.7582	0.8246	0.7884
0.0005	0.9556	0.9553	0.9550	0.9551
0.001	0.9654	0.9654	0.9651	0.9652
0.005	0.9700	0.9701	0.9695	0.9697
0.01	0.9655	0.9657	0.9651	0.9652

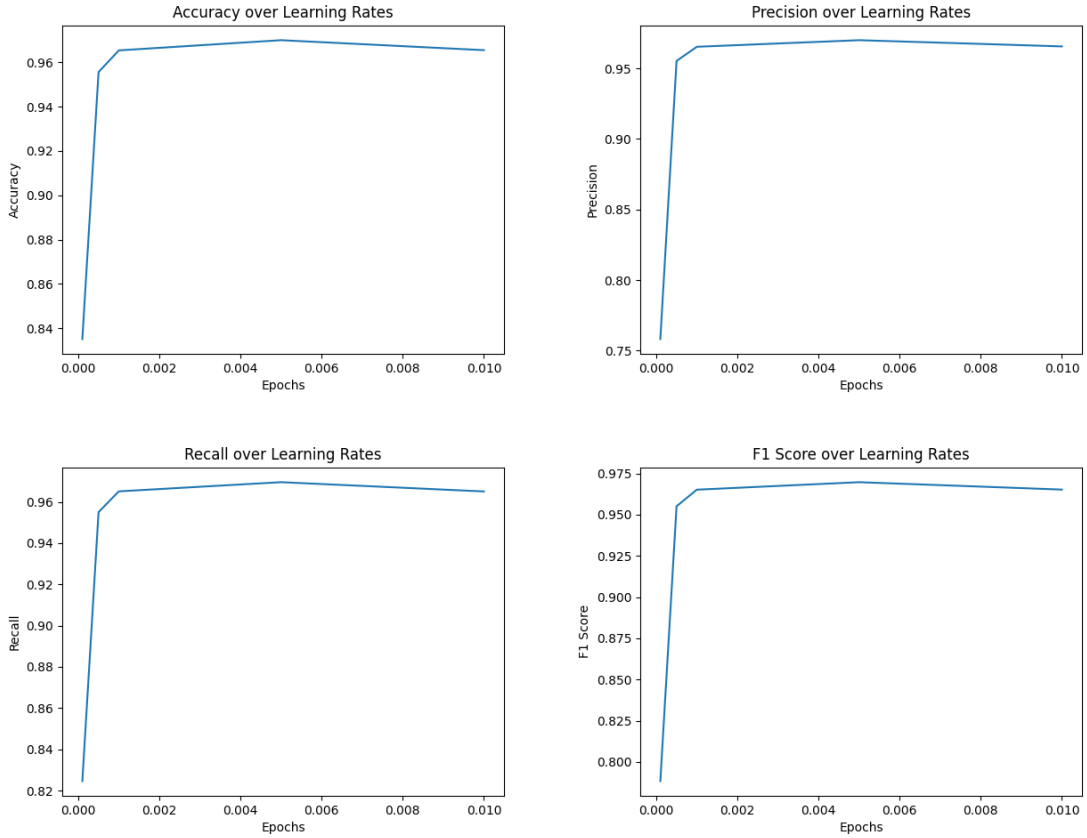


Fig. 25 Classification results over learning rate

2.4 Additional Works

2.4.1 Convolutional Neural Network

To compare the performance of different types of models, we also implemented a convolutional neural network that follows the structure of AlexNet. We can intuitively observe from Fig. 26 that evaluation metrics values of CNN model in each training epochs are much higher than those of default MLP model, all metrics values of CNN in 5 epochs reach around 0.99 while those of the default MLP trained in 5 epochs are only 0.7389, 0.5990, 0.7265, and 0.6545 respectively.

Epochs	**Accuracy**	**Precision**	**Recall**	**F1**
5	0.9864	0.9863	0.9863	0.9863
…	…	…	…	…
50	0.9917	0.9917	0.9916	0.9916

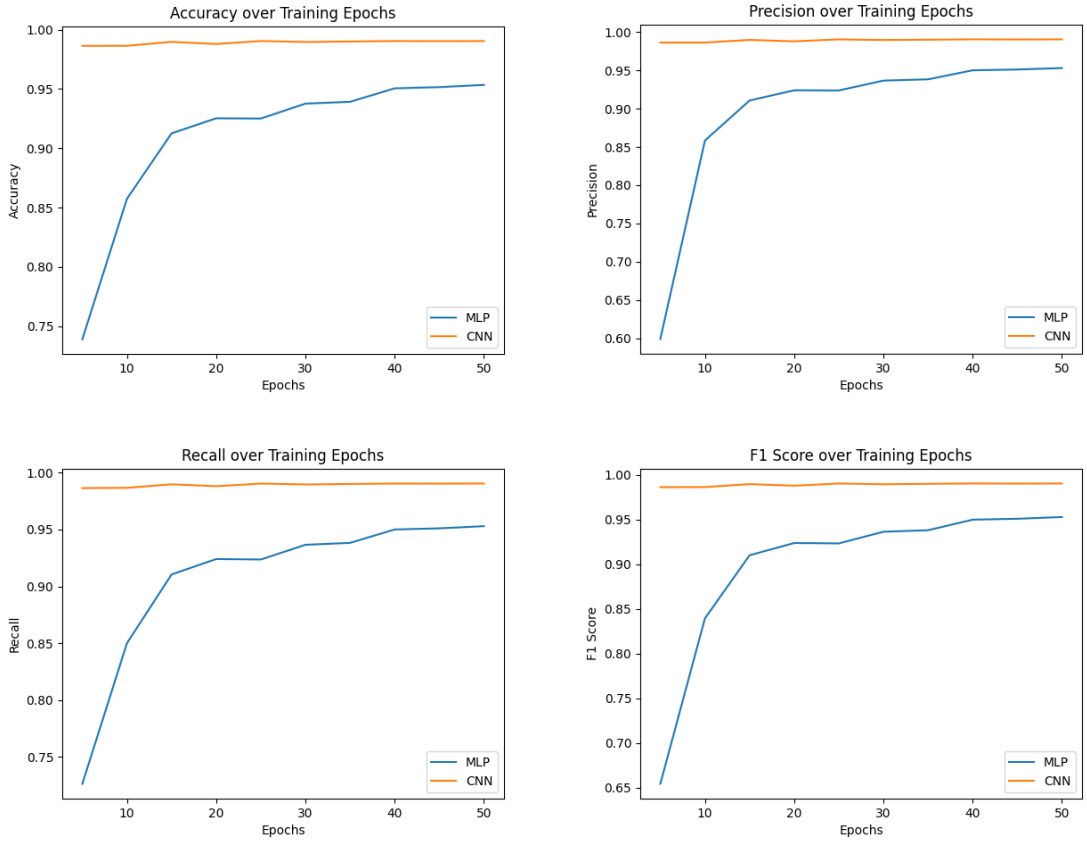


Fig. 26 Classification results over epochs

2.4.2 Training Data Augmentation

Fig. 27 shows some original training images, we designed an augmented training set as shown in Fig. 28 by: 1) randomly applying rotation, translation, scale, or shear; 2) modifying image's brightness and contrast; 3) normalizing images. Then we trained the default MLP model with the augmented training set following the same procedures as section 2.2.

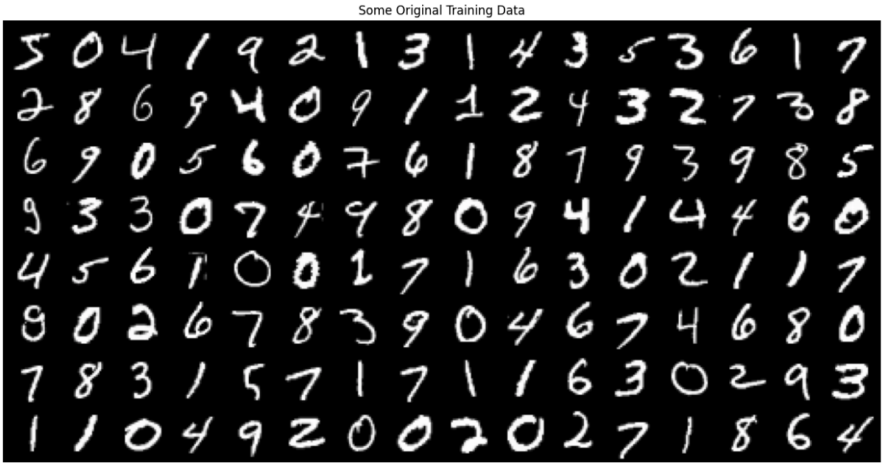


Fig. 27 Some original images

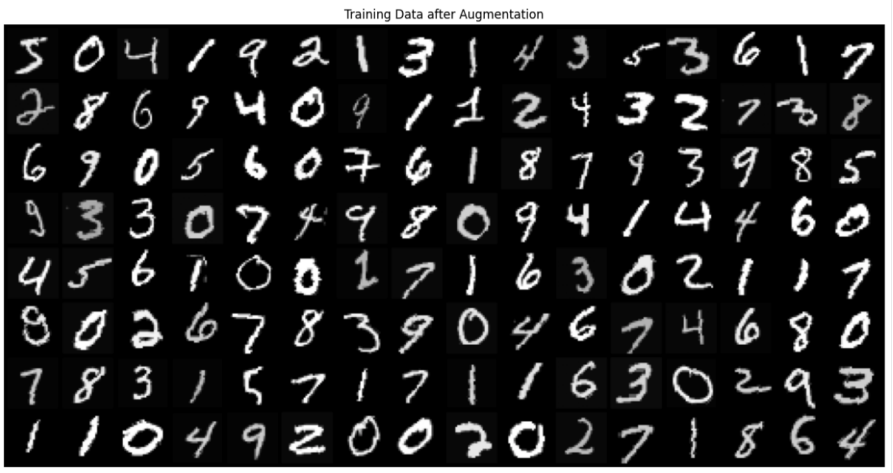


Fig. 28 Same images after augmentation

However, as shown in Fig. 29, it seems that the augmented training set cannot improve the default MLP model's performance as the values of the 4 metrics of MLP trained with augmented training set is lower than those of MLP trained with original training set in every epochs. For example, the 4 metrics of the default model trained in 50 epochs are 0.9535, 0.9530, 0.9529, and 0.9529 while those of the model trained with the augmented training set decrease to 0.8089, 0.8629, 0.8064, and 0.7865. Another interesting thing is that our data augmentation seems to have a negative impact on MLP model's performance, because the performance curves in Fig. 29 even begin decreasing with the increasing of epochs.

Epochs	**Accuracy**	**Precision**	**Recall**	**F1**
5	0.7375	0.6145	0.7248	0.6576
…	…	…	…	…
50	0.8089	0.8629	0.8064	0.7865

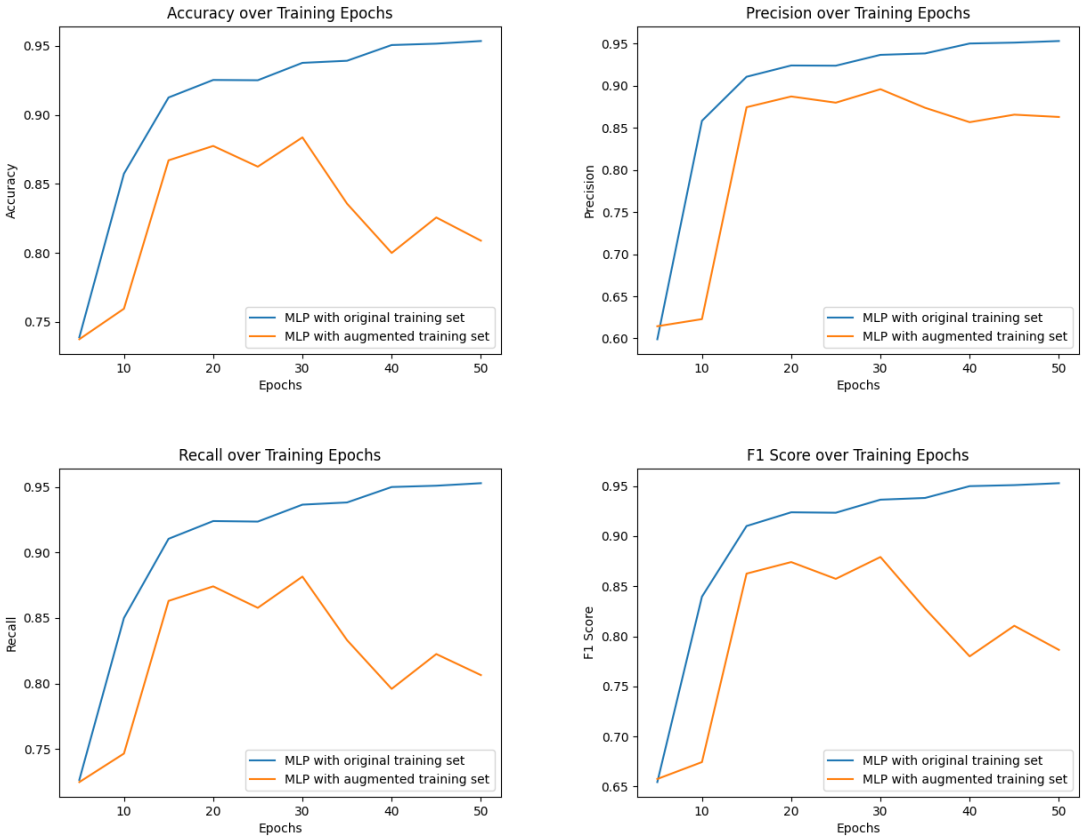


Fig. 29 Classification results over epochs