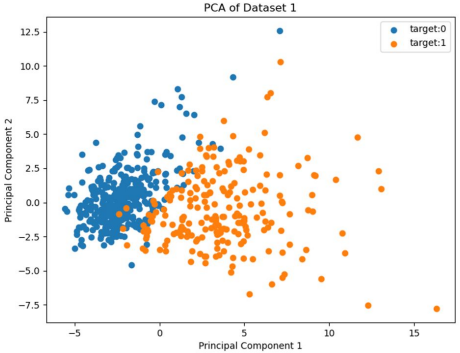
# CS6316 Machine Learning Application Project

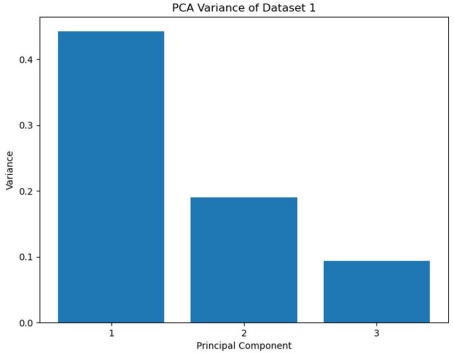
Ye Ma(hjk6th), Tongxuan Tian(nua3jz), and Cheng Yu(rjv8hm)

# Part1 Traditional Classification Algorithms

# 1.1.1 Dataset1 Analysis

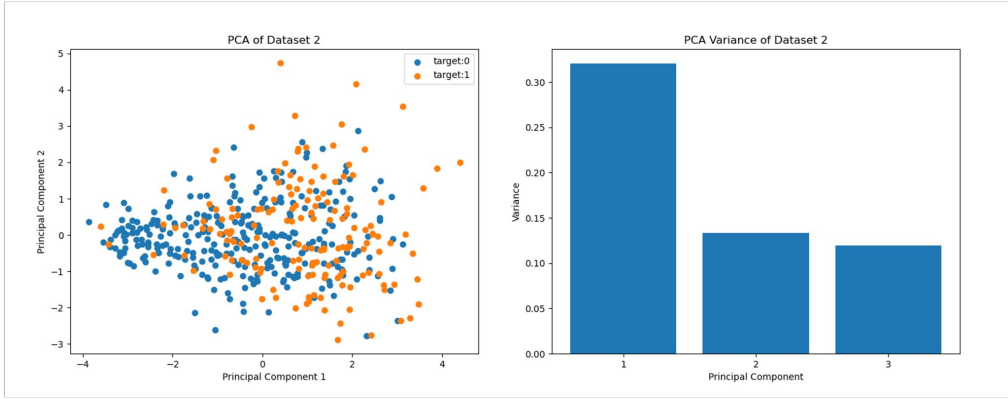
- There's a noticeable but not complete separation between the two classes along the first principal component (horizontal axis). This suggests that the first principal component captures a significant amount of the data's variance related to the class separation.
- The second principal component (vertical 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 Dataset2 Analysis

- 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

- Logistic regression algorithm models the probability of a binary response based on one or more predictor variables.
- 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

- 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.

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:

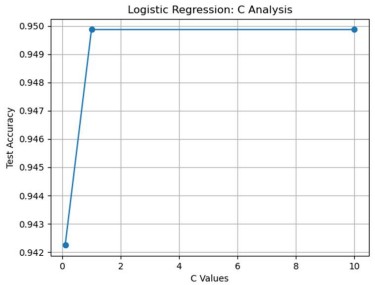
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 95.25 | 95.24 | 91.94 | 93.50 | 94.56 |

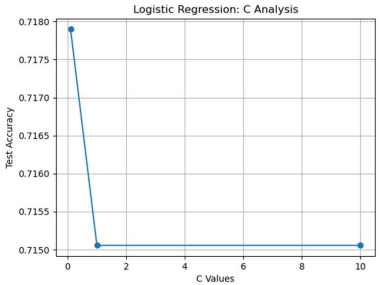
Best Result on Dataset 2:

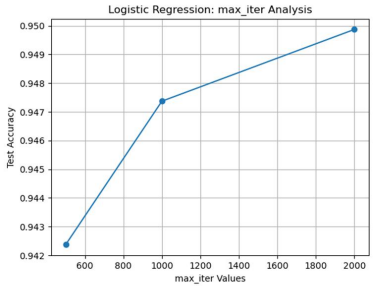
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 71.19 | 61.51 | 48.75 | 53.25 | 65.93 |

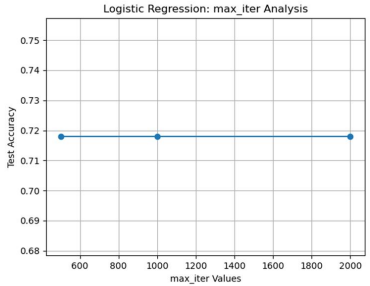
# 1.2.3 Performance Analysis

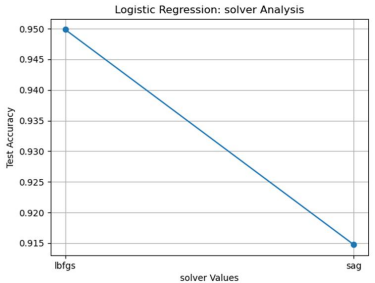
- Dataset 1: The training and validation scores are quite close, with the validation score slightly lower than the training score as the complexity increases (higher C values). This indicates a good bias-variance tradeoff with the potential for overfitting at higher complexities.
- Dataset 2: A larger 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.

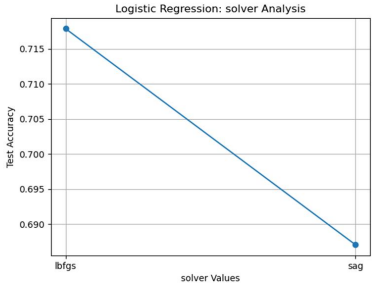


# 1.3 K Nearest Neighbor

## 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

- 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.

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:

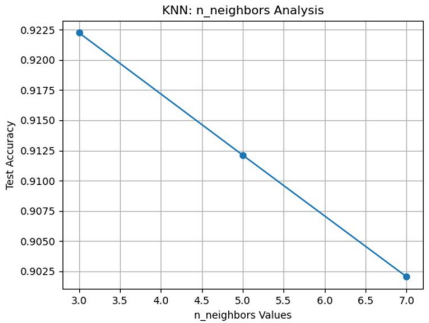
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 92.79 | 92.80 | 87.79 | 89.95 | 91.79 |

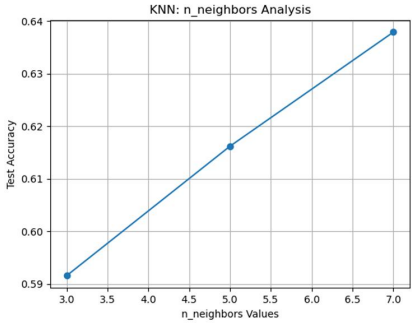
Best Result on Dataset 2:

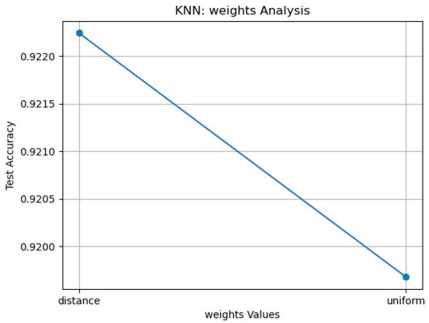
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 65.15 | 48.80 | 30.62 | 37.14 | 57.04 |

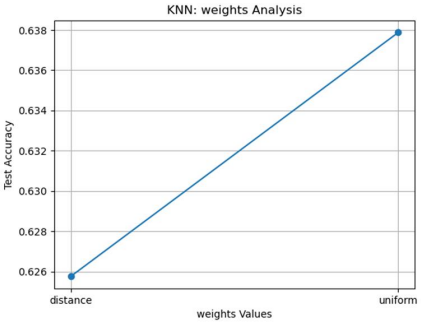
# 1.3.3 Performance Analysis

- 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

| 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:

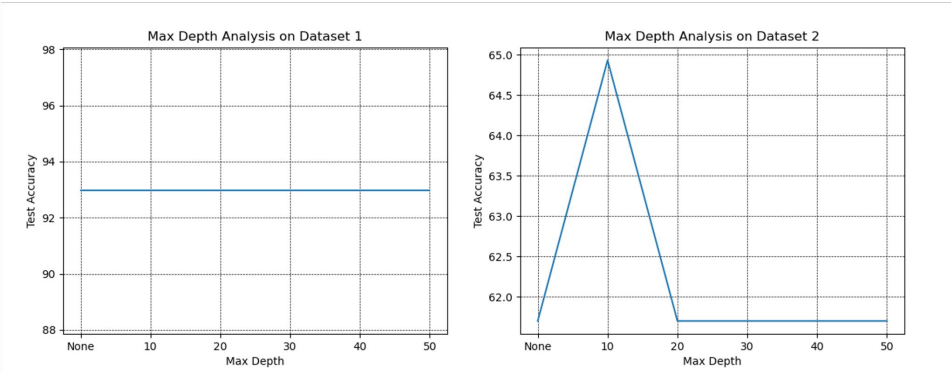
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 70.78 | 60.58 | 43.75 | 50.11 | 64.44 |

# 1.4.3 Performance Analysis

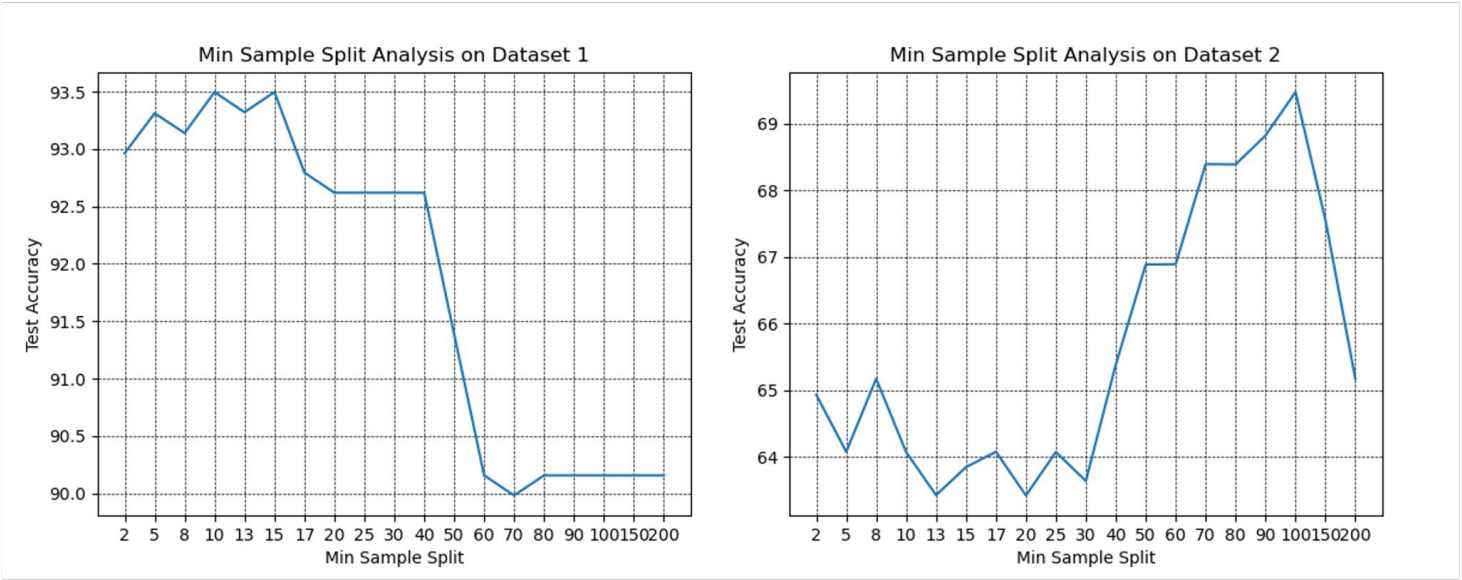
- Max Depth
- Min Samples Split
- Min Sample Leaf

# 1.4.3.1 Max Depth Analysis

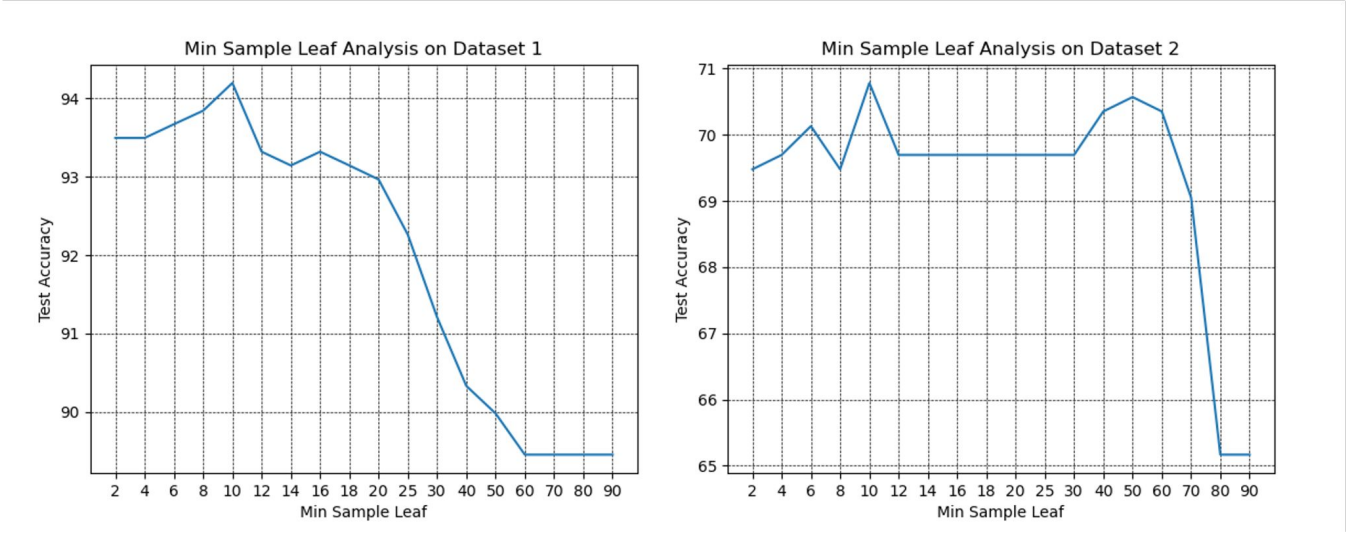
- In both datasets, we can see that max depth of decision tree has little effect on algorithm performance.
- For dataset 1, it is possibly because the dataset is too simple that the decision tree has already learned enough features to make decisions.
- While for dataset 2, it is possibly because dataset 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 Analysis



# 1.4.3.3 Min Sample Leaf Analysis



# 1.5 Support Vector Machine

## 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

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:

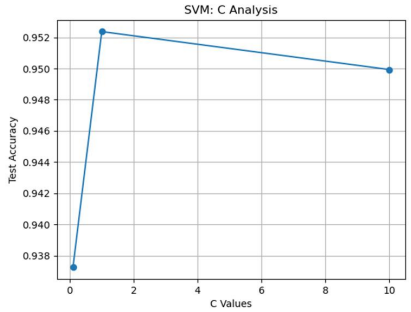
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 95.26 | 95.4 | 91.96 | 93.56 | 94.58 |

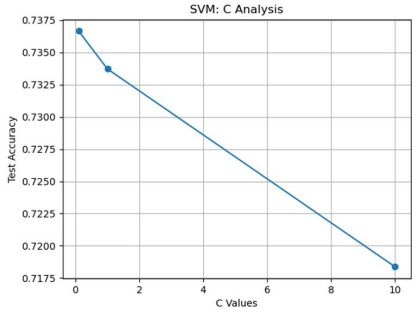
Best Result on Dataset 2:

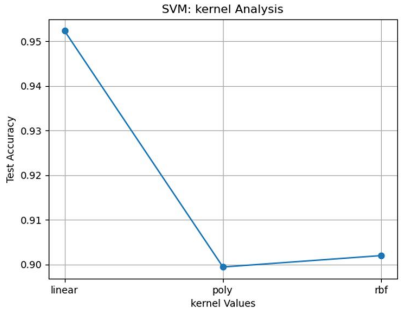
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 71.85 | 63.12 | 50 | 54.42 | 66.73 |

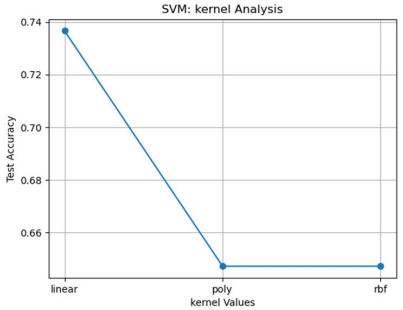
# 1.5.3 Performance Analysis

- 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

- 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.

# 1.6.2 10-fold Cross Validation

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 |

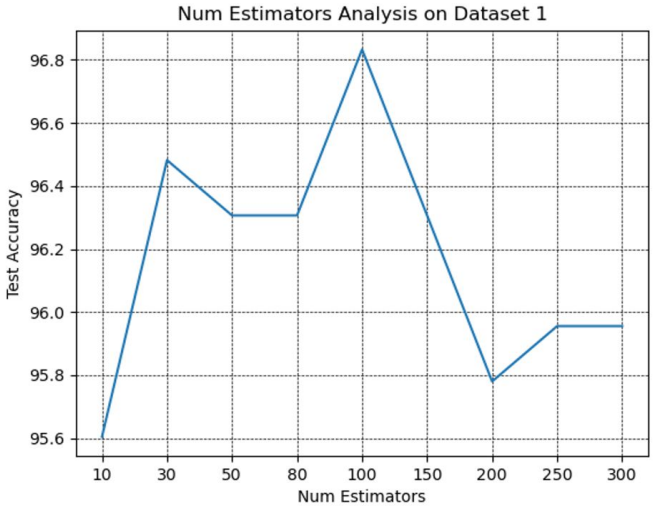
Best Result on Dataset2

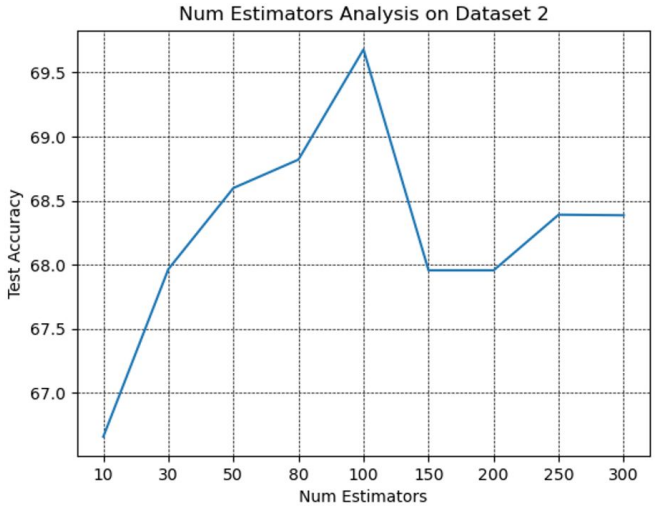
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 71.86 | 63.54 | 46.25 | 52.46 | 65.86 |

# 1.6.3 Performance Analysis

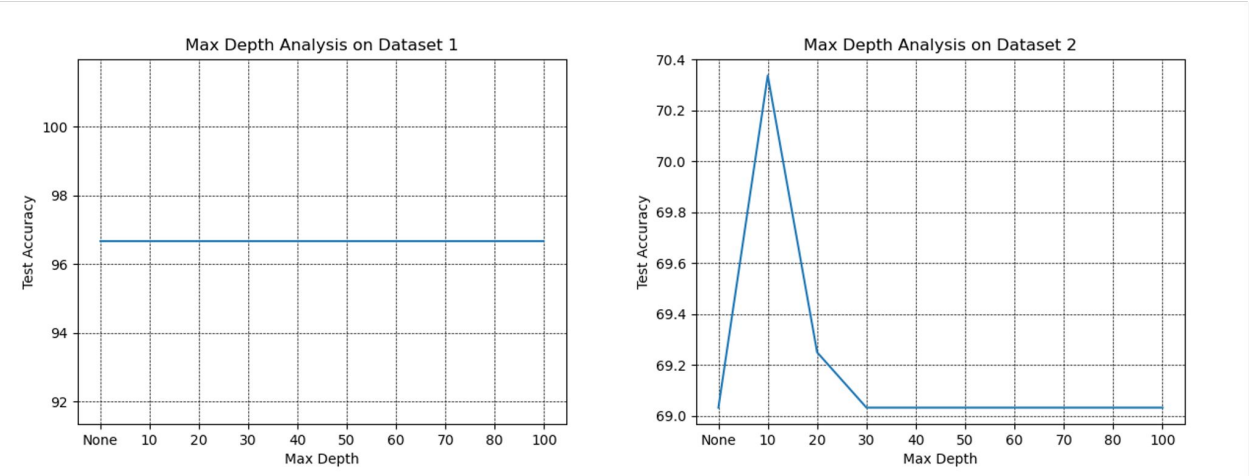
- Numbers of Estimators
- Max Depth
- Min Samples Split

# 1.6.3.1 Numbers of Estimators Analysis

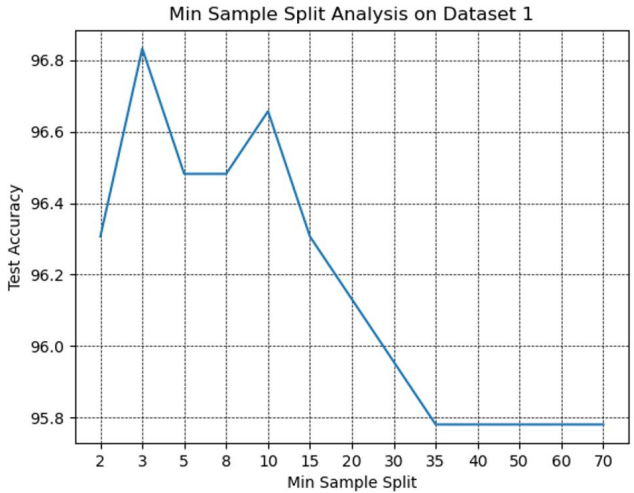


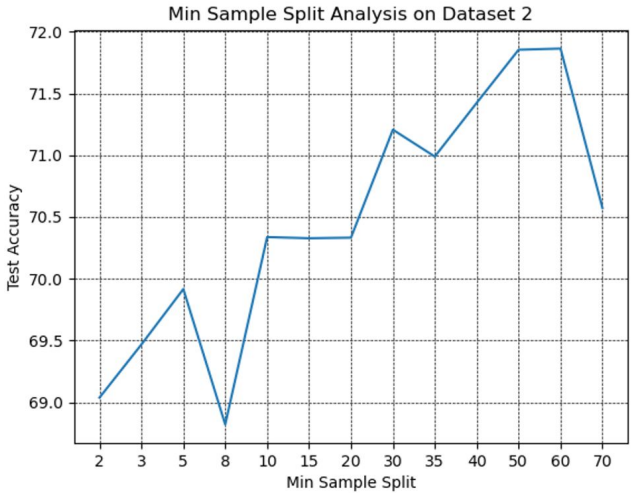



# 1.6.3.2 Max Depth Analysis



# 1.6.3.3 Min Samples Split Analysis



# 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

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

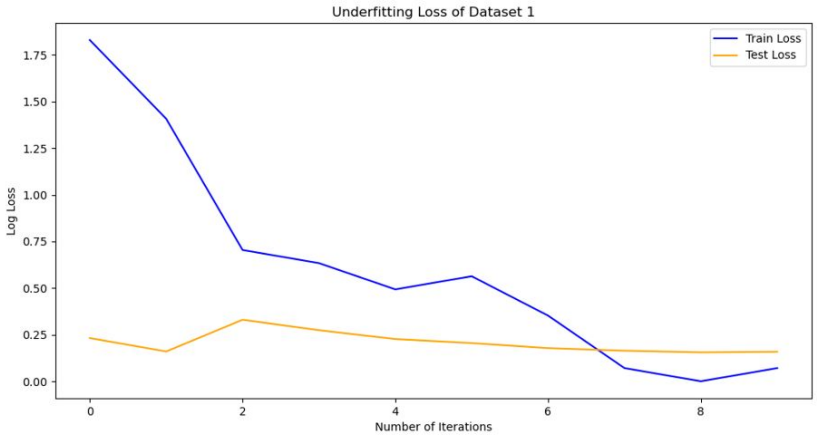
| Metric | Accuracy | Precision | Recall | F1 Score | AUC |
|---|---|---|---|---|---|
| Value(%) | 73.15 | 65.79 | 48.75 | 55.09 | 67.44 |

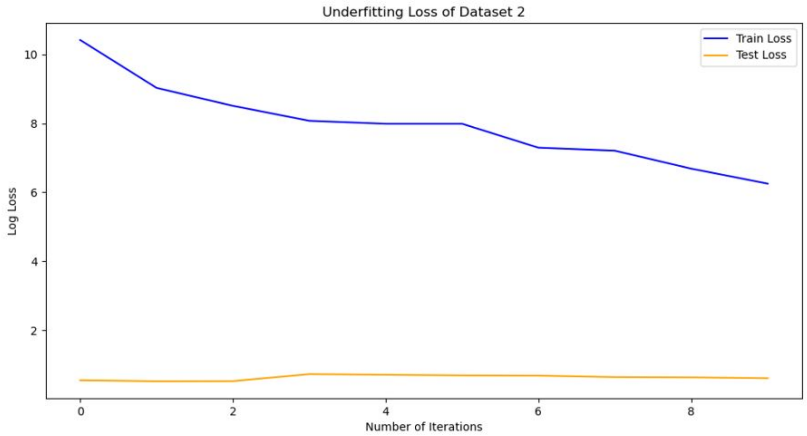
# 1.7.3 Performance Analysis

- Number of estimators
- Base estimators

# 1.7.3.1 Numbers of Estimators Analysis

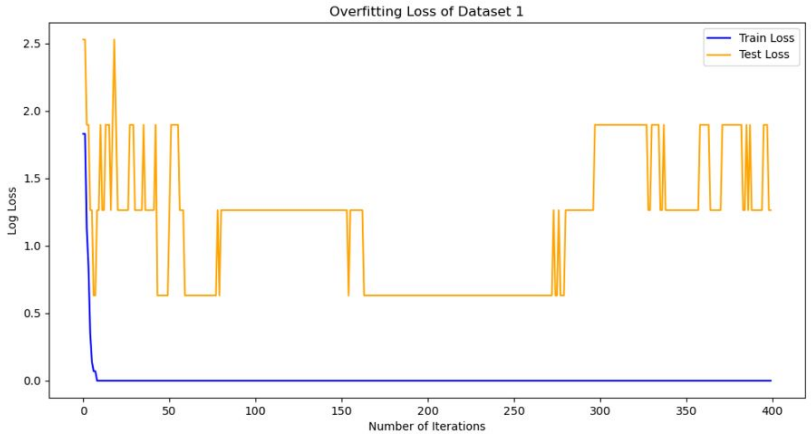
Underfitting

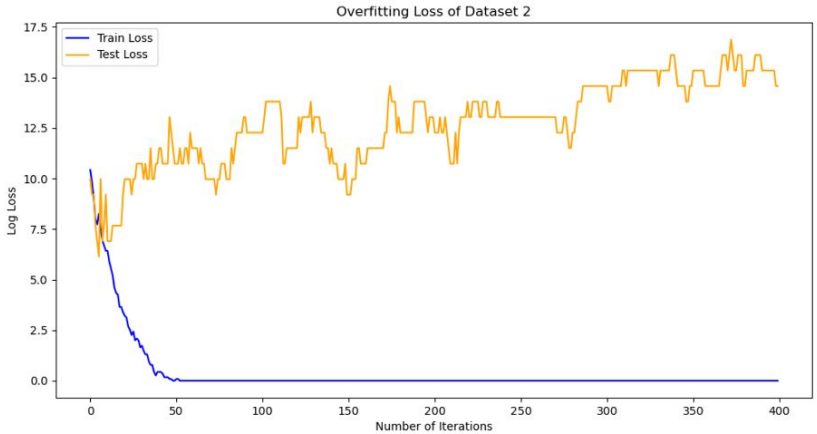


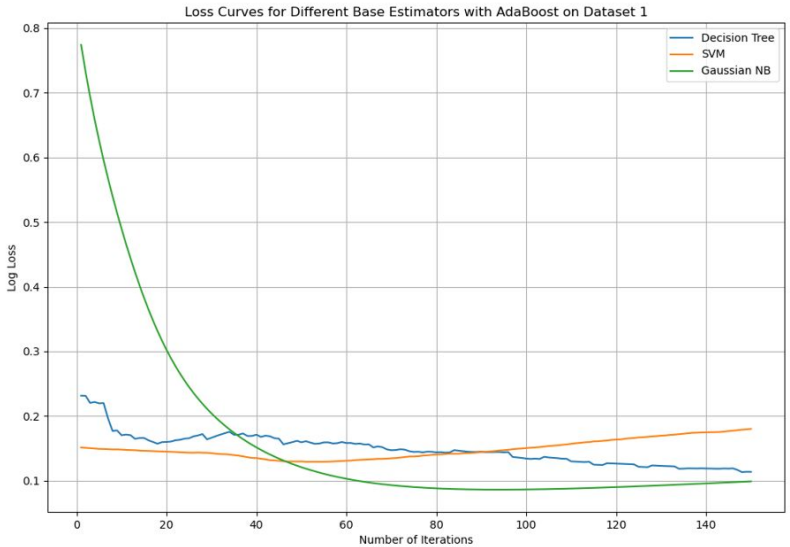
# 1.7.3.1 Numbers of Estimators Analysis

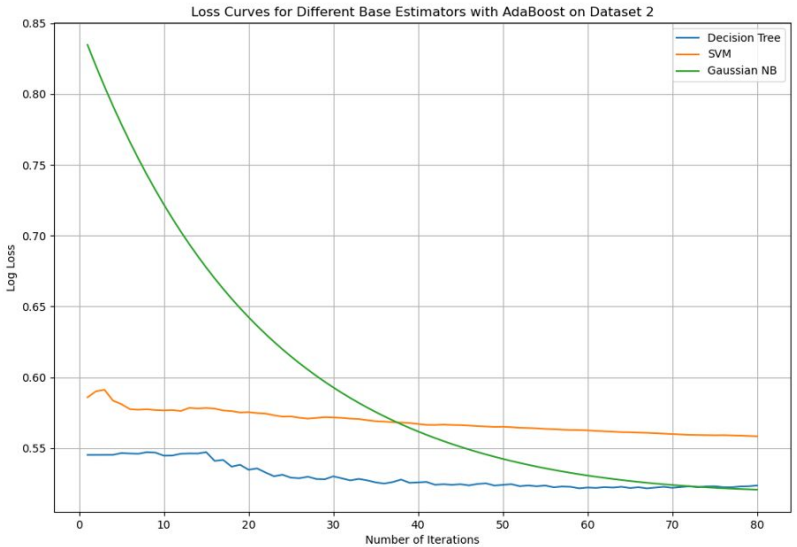
Overfitting



# 1.7.3.2 Base estimators Analysis



# 1.8 Model comparison

- Comparison on Dataset1
- Comparison on Dataset2

# 1.8.1 Comparison on Dataset1

| 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** |

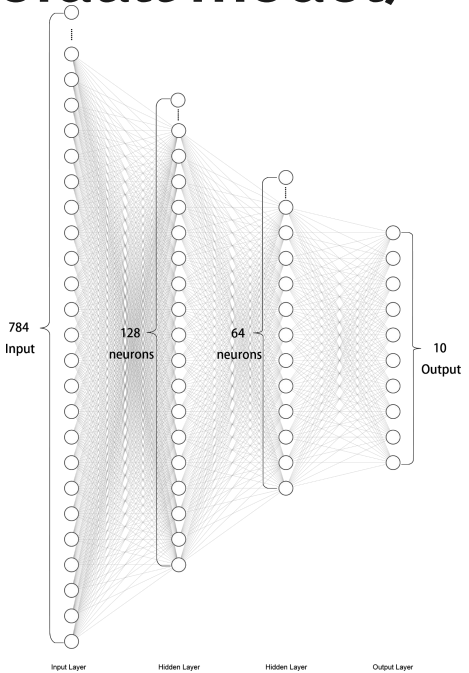
# 1.8.1 Comparison on Dataset2

| 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** |

# Part2 Neural Networks

# 2.1 Model Implementation (default model)

- 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, 192 hidden neurons in total
- A softmax output layer for 10 classes (MNIST labels: 0~9)
- xavier_uniform weight initialization and zeros bias initialization
- Trained for 5, 10, …, 45, 50 epochs (10 different epochs in total)
- Evaluated based on four evaluation metrics: Accuracy, Precision, Recall, and F1 score



# 2.2 Classification Results on MNIST

- The values of four evaluation metrics on the MNIST testing set increase as the number of epochs increases

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

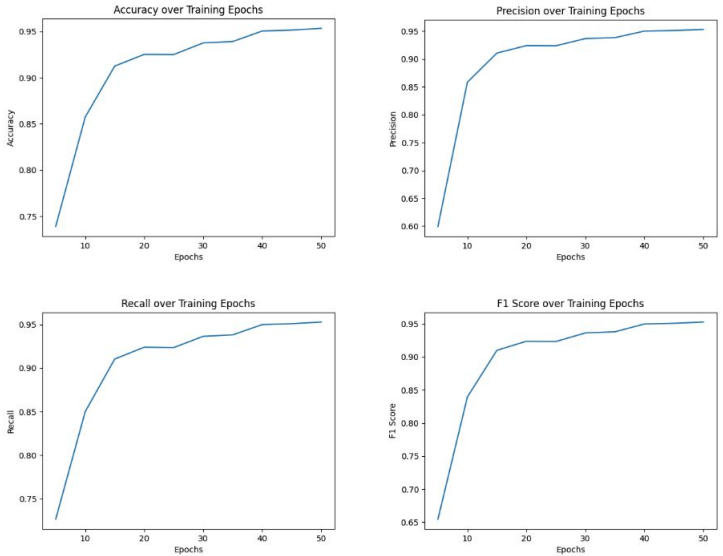


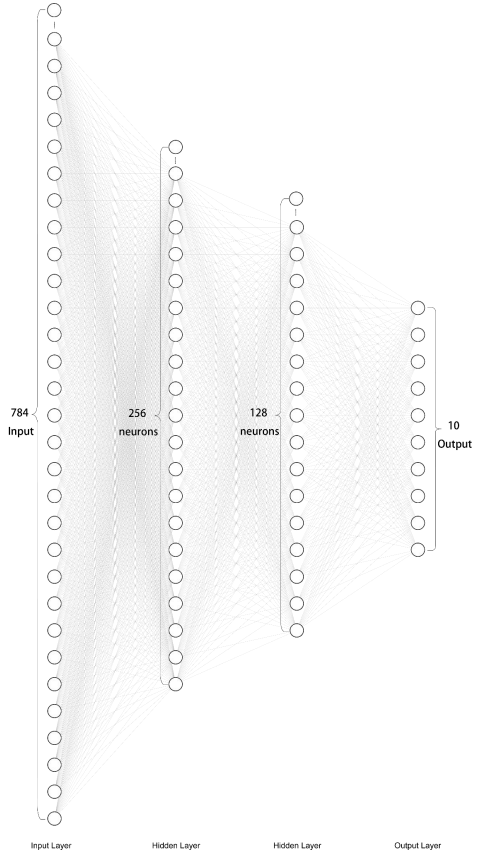

Fig. 10 Classification results over training epochs

# 2.3 Whether Hyperparameters Affect Model Performance

- Number of Hidden Units
- Weight/Bias Initialization
- Learning Rate

# 2.3.1 Number of Hidden Units

- An input layer with 784 (size of each MNIST image: 28x28) neurons
- A sigmoid hidden layer with 256 neurons, a sigmoid hidden layer with 128 neurons, 384 hidden neurons in total
- A softmax output layer for 10 classes (MNIST labels: 0~9)
- xavier_uniform weight initialization and zeros bias initialization
- Trained and tested the new model following the same procedures as section 2.1



# 2.3.1 Classification Results

- 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

| Epochs | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| 5 | 0.8220 | 0.7442 | 0.8115 | 0.7750 |
| ... | ... | ... | ... | ... |
| 50 | 0.9616 | 0.9614 | 0.9612 | 0.9612 |

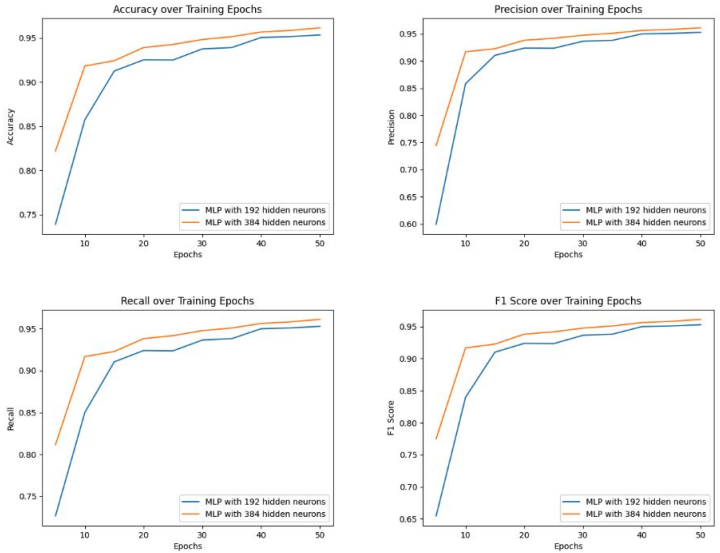



Fig. 12 Classification results over epochs

# 2.3.2 Weight/Bias Initialization

- The default model applies xavier_uniform weight initialization and zeros bias initialization
- Implemented 5 new MLP models that applied different weight/bias initialization
- All 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 |

# 2.3.2 Classification Results

- 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 |

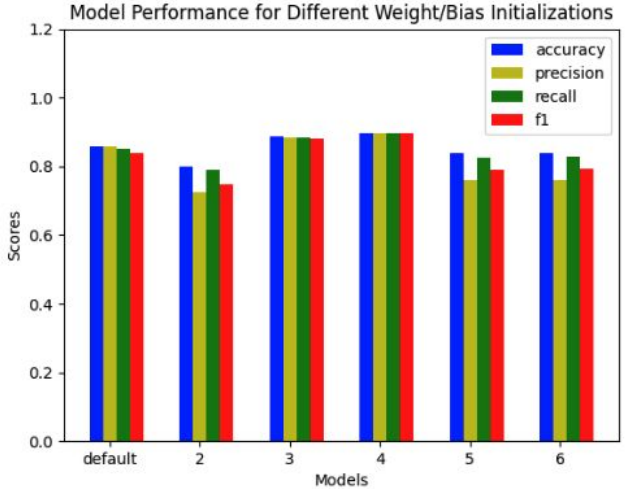



Fig. 13 Performance of different models

# 2.3.3 Learning Rate

- Tested 5 different learning rates: 0.0001, 0.0005, 0.001, 0.005, 0.01
- Used default MLP model
- All models were trained in 10 epochs
- 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

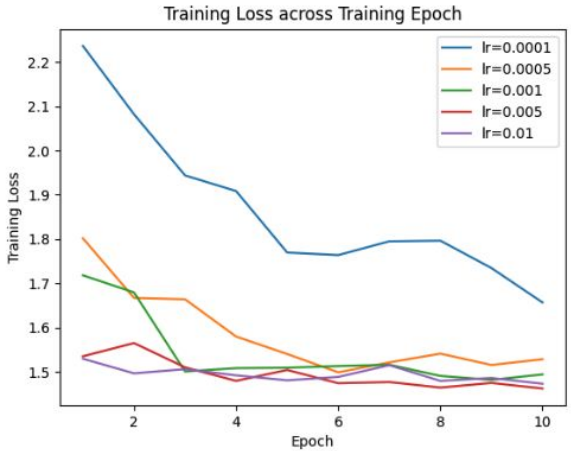



Fig. 14 Training loss across epochs

# 2.3.3 Classification Results

- 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

| 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 |

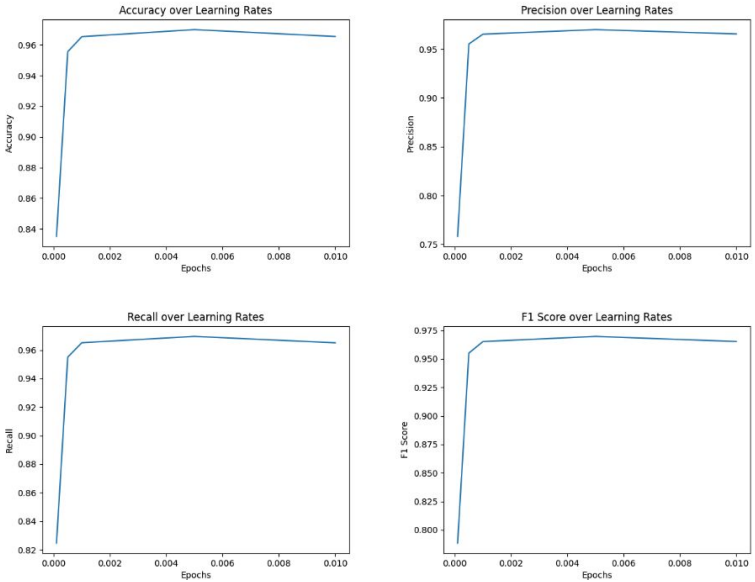



Fig. 15 Classification results over learning rate

# 2.4 Additional Works

- Convolutional Neural Network
- Training Data Augmentation

# 2.4.1 Convolutional Neural Network

- Follows the structure of AlexNet
- Trained and tested following the same procedures as section 2.1

# 2.4.1 Classification Results

- The evaluation metrics values of CNN model in each training epochs are much higher than those of default MLP model

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

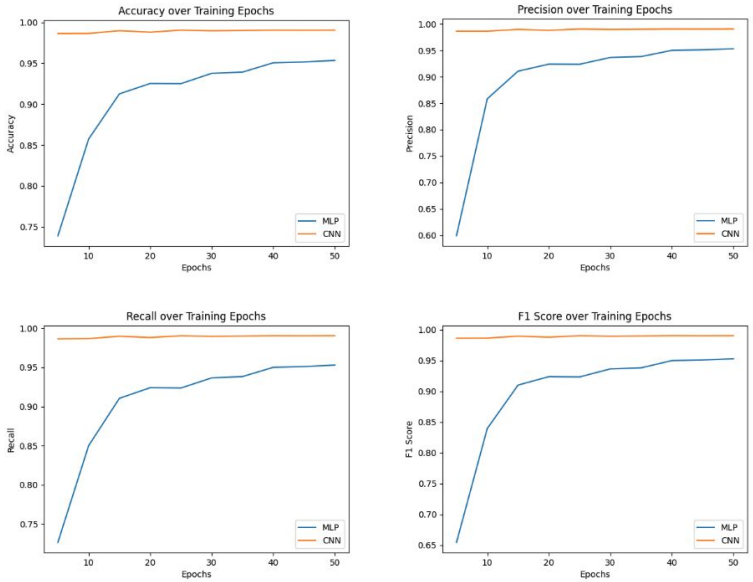
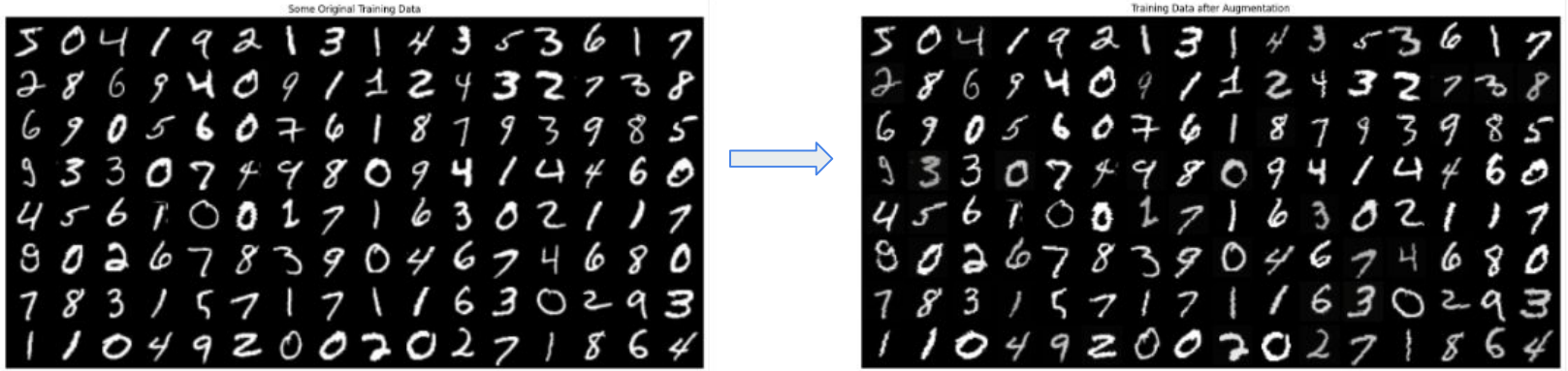



Fig. 16 Classification results over epochs

# 2.4.2 Training Data Augmentation

- Designed an augmented training set by :
  - 1) randomly applying rotation, translation, scale, or shear;
  - 2) modifying image's brightness and contrast;
  - 3) normalizing images
- Trained and tested following the same procedures as section 2.1



# 2.4.2 Classification Results

- Our data augmentation approach cannot improve model's performance
- the performance curves 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 |

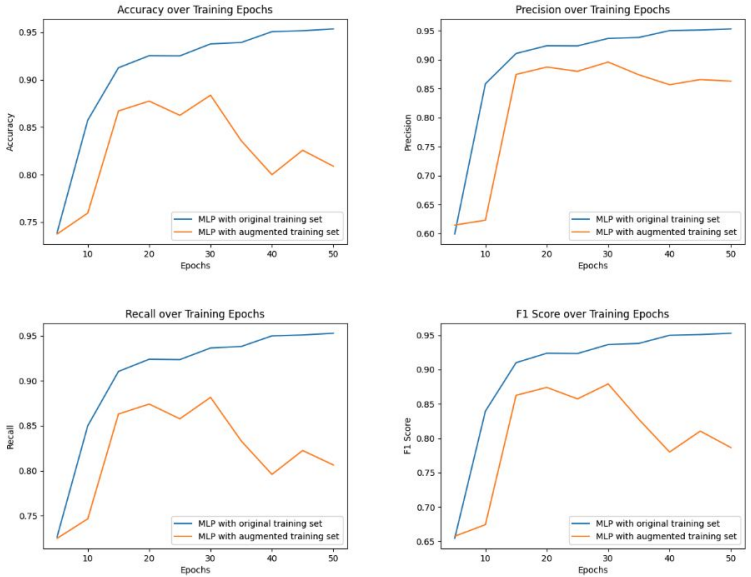



Fig. 19 Classification results over epochs

# Thank You!