CSIT 598 - Machine Learning

Homework 1

Objective

In this homework, we will implement and evaluate several classification algorithms on the MNIST dataset https://yann.lecun.com/exdb/mnist/. The goal is to compare the performance of different machine learning algorithms and analyze their strengths and weaknesses.

Dataset

We will work with the MNIST dataset (handwritten digits), which has been widely used in research for benchmarking. We may use preprocessed versions of the dataset available through libraries such as TensorFlow or scikit-learn for MNIST.

Workflow:

- Load the data
- Normalize the images
- Flatten the images for the classifiers
- Training the model
- Test the model
- Evaluate the model

Tasks

1. Decision Tree Classifier

- Implement a Decision Tree classifier
- Evaluate its performance using the accuracy metric

Solution:

At first, I have trained the decision tree classifier with the training data. Then I tested the model using the test data. The performance of the model is as follows:

Metric	Score
Accuracy	0.88
Precision	0.88
Recall	0.88
F1-score	0.88

The decision tree classifier performed better on MNIST data in all four metrics.

2. Naive Bayes Classifier

- Implement two Naive Bayes classifiers, i.e., Gaussian and Multinomial
- Compare the performance of the two and report your observations

Solution:

Then, I have trained two types of Naïve Bayes classifiers with the training data. After that I tested the models using the test data. The performance of the models is as follows:

Metric	Gaussian NB	Multinomial NB
Accuracy	0.56	0.84
Precision	0.69	0.84
Recall	0.55	0.83
F1 Score	0.51	0.83

Here, we can say that Multinomial Naïve Bayes performed better on the MNIST data.

3. Support Vector Machine (SVM)

- Implement a Support Vector Machine (SVM) classifier
- Experiment with different kernels, i.e., linear and RBF
- Compare the performance of the two and report your observations

Solution:

Then, I have trained two types of SVM classifiers with the training data. Then I tested the models using the test data. The performance of the models is as follows:

Metric	Linear-SVM	RBF-SVM
Accuracy	0.94	0.98
Precision	0.94	0.98
Recall	0.94	0.98
F1 Score	0.94	0.98

Here, we can say that SVM with RBF kernel performed better on the MNIST data.

4. k-Nearest Neighbors (k-NN)

- Implement the k-Nearest Neighbors (k-NN) algorithm.
- Experiment with different values of k and report your optimal k value

Solution:

Then, I have trained the KNN classifier with different values of k with the training data. After that I tested the model using the test data. The performance of the model for different k-value is as follows:

Metric	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9
Accuracy	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97
Precision	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97
Recall	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97
F1 Score	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97

Optimal k Value:

The performance of the k-NN classifier is almost the same across different values of k.

Based on the consistency of results, k=1 could be chosen as the optimal value for simplicity and to avoid overfitting.

Increasing the k value does not significantly affect performance, except for a small dip at k=2. Hence, smaller k values are preferred for this dataset.

5. Ensemble Methods

Implement the following ensemble methods and compare their performances:

- Bagging: Apply Bagging using Decision Trees as base estimators
- Random Forest: Implement a Random Forest classifier
- Boosting: Apply AdaBoost with Decision Trees as base estimators

Solution:

Then, I have trained some ensemble classifiers. Its noted that I applied Decision Trees as base estimators for Bagging and Boosting (AdaBoost). Besides I trained Random Forest with the training data. Then I tested the models using the test data. The performance of the models is as follows:

Parameter for Bagging:

estimator=DecisionTreeClassifier(), n_estimators=100, random_state=42

Parameters for Random Forest:

n_estimators=100, random_state=42

Parameters for Boosting (AdaBoost)

estimator=DecisionTreeClassifier(), n_estimators=100, random_state=42

Metric	Bagging	Random Forest	AdaBoost
Accuracy	0.96	0.97	0.88
Precision	0.96	0.97	0.87
Recall	0.96	0.97	0.87
F1 Score	0.96	0.97	0.87

- Random Forest performs best across all metrics (Accuracy, Precision, Recall, F1 Score) at 0.97, making it the top choice.
- Bagging closely follows with 0.96 across metrics, showing strong but slightly lower performance.
- AdaBoost lags with 0.87 in all metrics, indicating it is less effective on this dataset

6. Performance Comparison and Analysis

- Summarize your findings. Which model performed the best? What are the trade-offs between the different classifiers?

Solution:

Best Performing Model:

Random Forest emerges as the top performer, achieving an accuracy of **0.97** along with corresponding precision, recall, and F1 scores of **0.97**. This model excels in handling high-dimensional data and balances bias and variance effectively.

Other Models:

Bagging follows closely with an accuracy of **0.96**. It is robust against overfitting but slightly less effective than Random Forest due to its lack of randomness in feature selection.

SVM with RBF kernel achieves an accuracy of **0.98**, making it highly effective for complex data like MNIST. Its strong performance in precision, recall, and F1 score indicates it is well-suited for this classification task.

Decision Tree performs reasonably well with an accuracy of **0.88**, but it is prone to overfitting and may not generalize as well as ensemble methods like Random Forest or Bagging.

Naïve Bayes models show varied performance, with **Multinomial NB** achieving an accuracy of **0.84**, while **Gaussian NB** lags behind at **0.56**. Despite being fast and efficient, they struggle with high-dimensional datasets.

Trade-offs:

- **Complexity vs. Performance**: Models like Random Forest and SVM provide high accuracy but require more computational resources and time. In contrast, Naïve Bayes is faster but sacrifices accuracy.
- **Overfitting Risk**: Decision Trees may overfit the training data, whereas ensemble methods like Bagging and Random Forest mitigate this risk through aggregation.
- **Model Interpretability**: Decision Trees are easier to interpret, which can be a significant advantage in applications requiring transparency.

Thank You!