Programming Assignment 2 Report

Aditi Jorapur
015617225
aditi.jorapur@sjsu.edu

The Iris Dataset

The Iris Dataset is data containing information about the types of Iris flowers. This dataset contains 150 iris flower samples with each flower being one of the three types: Setosa, Versicolour, and Virginica. The four different features given for each flower are Sepal Length, Sepal Width, Petal Length and Petal Width. The goal of the Iris Dataset using machine learning algorithms is to classify the Iris flowers into one of the three types based on the four different features.

Evaluation Metrics

	Accuracy	F1- Score	ROC AUC
Naive Bayes	0.9467	0.9463	0.9943
Support Vector Machines	0.9533	0.9528	0.9997
Random Forest	0.9533	0.9526	0.9885
XGBoost	0.96	0.9589	0.9831
K-Nearest Neighbors	0.9533	0.9526	0.9974

Preprocessing:
I preprocessed the data using standard scalar to make sure that each feature has a mean of 0 and a standard deviation of 1. This was helpful for the support vector machines and k-nearest neighbors because these models are more on the sensitive side to the scaling of features.

I used five fold cross validation to estimate the accuracy, F1-scores, and ROC AUC of the models on the training and test sets.

Overfitting & Hyperparameter Tuning:
Initially, I computed the accuracy, f1-score, and ROC AUC of each of the five models on both the test and training data. I found overfitting in the Random Forest and XGBoost models.

Random Forest Evaluation Metrics before handling overfitting:

Train	Test
Accuracy: 1.0000	Accuracy: 0.9533

F1 Score: 1.0000	F1 Score: 0.9532
ROC AUC: 1.0000	ROC AUC: 0.9934

I noticed the model was overfitting because the Accuracy, F1 Score, and ROC AUC for the train data was 1.0 which was very high but also these values compared to the test evaluation metrics had a big difference. To reduce the overfitting, I tuned the hyperparameters of the Random Forest model. When tuning the hyperparameters I:

- Set the number of trees to 10
- Set the max depth of trees to 5
- Set the minimum number of samples to split to 5
- Set the minimum number of samples to be a leaf to 5
- Set the max features to be considered to ‘sqrt’ (square root of the total number of features will be considered)
- Bootstrap set to true (samples of the data are used with replacement)

The hyperparameters that I tuned helped reduce overfitting for the Random Forest model.

XGBoost Evaluation Metrics before handling overfitting:

Train	Test
Accuracy: 1.0000	Accuracy: 0.9400
F1 Score: 1.0000	F1 Score: 0.9397
ROC AUC: 1.0000	ROC AUC: 0.9758

I noticed the model was overfitting because the Accuracy, F1 Score, and ROC AUC for the train data was 1.0 which was very high but also these values compared to the test evaluation metrics had a big difference. To reduce the overfitting, I tuned the hyperparameters of the XGBoost model. When tuning the hyperparameters I:

- Set the number of rounds to 5
- Set the maximum depth to 3
- Set the learning rate to 0.1 (shrinks each trees contribution)
- L1 & L2 regularization both at 0.1

The hyperparameters that I tuned helped reduce overfitting for the XGBoost model.

Observations

All five of the models have high accuracies indicating good performance. I noticed a variation among the accuracies and other evaluation metrics throughout the models, however, they are overall very similar. The two models with the highest ROC AUC scores were XGBoost and Support Vector Machines which may indicate that the classes had better discrimination. After performing actions to attempt to reduce overfitting, there was less overfitting observed throughout the models. XGBoost has the highest accuracy out of all the models with 0.96. Support Vector Machines and Random Forest are close behind with 0.9533. XGBoost also has the highest F1-score with 0.9599. This means that there was a good balance abstained with recall and precision. The Support Vector Machine model was able to have the highest ROC AUC score at 0.9997 which means it was able to split the classes in a very good way.