# Soumya Mukherjee | CH24M571 | Assignment 4

# Lab Report Question 1

**1.0 Preface**

The question 1 of the assignment asks us to do the following on the synthetic data set :

1. Develop, optimize, and evaluate classification models using Random forest, Ada boost, and KNN classifiers. It contains 5000 samples and 50 features, along with a target column that indicates the class labels 0, 1 and 2.

**2.0 Methodology :**

The dataset was loaded from the assignment4.csv file. It contains 5000 samples and 50 features, along with a target column that indicates the class labels 0, 1 and 2. The data was pre-processed , models were built and the scores were compared .

**2.0.1 Data Pre-processing :**

The dataset was checked for null and NaN values . Aggregation of null values showed no null values . Data was scaled using StandardScalar . In the second question , we have been asked to do feature – scaling using SelectFromModel . The purpose of this is to reduce the dimensionality and improve the model performance.

Data was split in 80:20 as asked in the question

**2.0.2 Model Implementation**

The models trained on were Random Forest , AdaBoost and KNN .

For training the following params\_grids were used

Random Forest :   
**N-estimators** : [100, 150, 200, 300]. This denotes the max trees in the forest

**max\_depth**: This defines the maximum depth allowed for each tree. The provided values are [20, 30, 40]. Here, the depths are taken limited because, given the dataset's 50 features, very deep trees might lead to overfitting

**min\_samples\_split**: This sets the minimum number of samples required to split a node. The grid includes [2, 5, 10].

**crietrion :** We have used Gini and Entropy .

Adaboost :

n\_estimators: This parameter sets the number of weak learners to be added to the ensemble. The grid values are [50, 100, 200, 300, 500].  
• learning\_rate: This is the learning rate used to shrink the contribution of each classifier. The candidate values include [0.005, 0.01, 0.05, 0.1]. Adaboost is giving accuracy less than random guess at learning rate of 0.01 , hence we have tried 0.005

K NN :

n\_neighbors: Determines the number of neighbors to use for prediction. The options provided are [3, 5, 7, 9].  
• weights: This specifies the weight function used in predictions. Available settings are 'uniform' (equal weights to all neighbors) and 'distance' (weights are inversely proportional to the distance).  
• metric: The distance metric used for evaluating neighbor closeness. The candidate metrics used are 'euclidean' and 'manhattan'.

These parameter grids can be fed into a grid search (or a related hyperparameter tuning strategy) to identify the best configuration for each machine learning model.

The following is the best hyperparameters as reported when tuned using GridSearchCV

|  |  |
| --- | --- |
| Model | Best HyperParameters |
| Random Forest | max\_depth=30, min\_samples\_split=5, n\_estimators=150, random\_state=40 |
| AdaBoost | learning\_rate=0.01, estimator=DecisionTreeClassifier(random\_state=40), random\_state=40 |
| KNN | metric='euclidean', n\_neighbors=9, weights='distance' |

**3.0 Evaluation**

Two parameters were used for evaluation

* Accuracy: The accuracy of the optimized model was calculated on the test set.
* Confusion Matrix and Classification Report: The confusion matrix and classification report were generated to evaluate the model's performance in terms of precision, recall, and F1-score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Precision | Recall | F1-score |
| Random Forest | 0.655 | 0.70 | 0.66 | 0.60 |
| AdaBoost | 0.485 | 0.49 | 0.48 | 0.49 |
| KNN | 0.745 | 0.75 | 0.74 | 0.73 |

**4.0 Results and Conclusion**

### **4.0.1 Random Forest Performance**

* **Overall Accuracy:** **65.5%**

Class **0** has the highest recall (**97%**), meaning the model correctly identifies most of them. Class **1** recall is lower (**47%**), indicating misclassification with other classes. Class **2** has a very low recall (**12%**), meaning it is often misclassified.

### **4.0.2 AdaBoost Performance**

* **Overall Accuracy:** **48.5%** (Lowest among all models)

**The model p**erforms **poorly in recall and F1-score** across all classes and struggles the most with **class 2** (only 28% recall). Results seem to indicate that AdaBoost might not be handling **high-dimensional and noisy data well**.

### **4.0.3 KNN Performance**

* **Overall Accuracy:** **74.5%** (Best performing model)

**The model p**erforms the **most balanced classification**. Still struggles slightly with class **2**, but overall, it has **better recall and F1-score** than the other models.

To validate the model stability , 5-fold cross validation was used .

Results :

|  |  |
| --- | --- |
| Model | Score |
| Random Forest | 0.648 |
| Adaboost | 0.468 |
| KNN | 0.626 |

**The validation scores indicates that Random Forest and KNN generalize well**, with scores close to test accuracy where as **AdaBoost struggles significantly**, likely due to weak base models in a noisy dataset.