### Results

|  |  |  |
| --- | --- | --- |
| **Model** | **Normal** | **Hyperparameter Tuned** |
| **Decision Tree** | Accuracy:  0.7402597402597403  Precision:  0.7327036599763873  Recall: 0.7402597402597403  F1-score: 0.73433798905497 | Accuracy:  0.8571428571428571  Precision:  0.8633004926108374  Recall: 0.8571428571428571  F1-score: 0.858874458874459 |
| **Random Forest** | Accuracy:  0.8831168831168831  Precision:  0.8823640127987955 Recall: 0.8831168831168831 F1-score:  0.8812085873310362 | Accuracy:  0.8571428571428571  Precision:  0.8554175293305728 Recall: 0.8571428571428571 F1-score:  0.8548104956268221 |
| **Adaboost** | Accuracy:  0.7792207792207793  Precision:  0.7734230055658627 Recall: 0.7792207792207793 F1-score:  0.7726355966562768 | Accuracy:  0.7662337662337663  Precision:  0.7602660992491501  Recall: 0.7662337662337663  F1-score: 0.753482880755608 |

### Findings

After looking at the results, we can conclude that hyperparameter-tuned models may perform better than a normal model as it is optimized. As it involves searching for the best set of hyperparameters, it maximize the model's performance on a validation set. This is true in case of Decision tree where the scores went from being aroung 73-74 to 85-86.

However, it is not 100% necessary that that scores will always be higher as it is seen in my case for the 2 models, Random Forest and Adaboost. There could be several reasons for this such as overfitting. Hyperparameters may be tuned to the specific characteristics of the validation set, and the model is not able to generalize well to new data. Another reason could be that the hyperparameters set are not optimal ones, hence giving poor performance