This document explains the logic behind the hyperparameter tuning process performed on the `DecisionTreeClassifier`, `RandomForestClassifier`, and `AdaBoostClassifier` models, how the combinations of hyperparameters were tested, and provides an analysis of the results to determine the best-performing model.

---

1. Hyperparameter Tuning Logic

1.1 Decision Tree Classifier

The hyperparameter grid for the `DecisionTreeClassifier` included:

- `max_depth`: Limits the maximum depth of the tree. Values tested: `[3, 5, 10, None]`.
- `min_samples_split`: Minimum number of samples required to split an internal node. Values tested: `[2, 5, 10]`.
- `criterion`: The function to measure the quality of a split. Values tested: `['gini', 'entropy']`.

Total Combinations:

$4\ (max\_depth) \times 3\ (min\_samples\_split) \times 2\ (criterion)\ =\ 24\ combinations$

Using 5-fold cross-validation, each combination was tested 5 times, resulting in:

$24\ combinations \times 5\ folds\ =\ 120\ fits.$

The grid search identified the best hyperparameters:

- `criterion='entropy'`
- `max_depth=3`
- `min_samples_split=2`

1.2 Random Forest Classifier

The hyperparameter grid for the `RandomForestClassifier` included:

- `n_estimators`: Number of trees in the forest. Values tested: `[50, 100, 200]`.
- `max_depth`: Maximum depth of each tree. Values tested: `[3, 5, 10, None]`.
- `max_features`: Number of features to consider when looking for the best split. Values tested: `['sqrt', 'log2']`.
- `min_samples_split`: Minimum samples required to split an internal node. Values tested: `[2, 5, 10]`.
- `class_weight`: Strategy to handle imbalanced data. Values tested: `[None, 'balanced']`.

Total Combinations:

$3\ (n\_estimators) \times 4\ (max\_depth) \times 2\ (max\_features)$
$\times\ 3\ (min\_samples\_split) \times 2\ (class\_weight)\ =\ 144\ combinations$

Using 5-fold cross-validation, each combination was tested 5 times, resulting in:

$144\ combinations \times 5\ folds\ =\ 720\ fits$

The grid search identified the best hyperparameters:

- `n_estimators=50`
- `max_depth=10`
- `max_features='log2'`
- `min_samples_split=2`
- `class_weight='balanced'`

1.3 AdaBoost Classifier

The hyperparameter grid for the `AdaBoostClassifier` included:

- `n_estimators`: Number of weak learners (iterations). Values tested: `[50, 100, 200]`.
- `learning_rate`: Shrinks the contribution of each weak learner. Values tested: `[0.01, 0.1, 1]`.
- `estimator`: The base learner used for boosting. Decision trees with depths `[1, 2, 3]` were tested.

Total Combinations:

$3\ (n\_estimators) \times 3\ (learning\_rate) \times 3\ (estimator\ depths)\ =\ 27\ combinations$

Using 5-fold cross-validation, each combination was tested 5 times, resulting in:

$27\ combinations \times 5\ folds\ =\ 135\ fits.$

The grid search identified the best hyperparameters:

- `n_estimators=200`
- `learning_rate=1`
- `estimator=DecisionTreeClassifier(max_depth=3)`

---

2. Model Evaluation

Each model was evaluated on the test set using the following metrics:

- **Accuracy**: Percentage of correctly classified samples.
- **Precision**: Proportion of true positives among all positive predictions.
- **Recall**: Proportion of true positives among all actual positives.
- **F1 Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Breakdown of true positives, true negatives, false positives, and false negatives.

2.1 Decision Tree Classifier

- **Confusion Matrix**: [[38, 4], [2, 70]]
- **Metrics**:
  - Accuracy: `0.9474`
  - Precision: `0.9459`
  - Recall: `0.9722`
  - F1 Score: `0.9589`

2.2 Random Forest Classifier

- **Confusion Matrix**: [[39, 3], [3, 69]]
- **Metrics**:
  - Accuracy: `0.9474`
  - Precision: `0.9583`
  - Recall: `0.9583`
  - F1 Score: `0.9583`

2.3 AdaBoost Classifier

- **Confusion Matrix**: [[38, 4], [2, 70]]
- **Metrics**:
  - Accuracy: `0.9474`
  - Precision: `0.9459`
  - Recall: `0.9722`
  - F1 Score: `0.9589`

2.4 Voting Classifier

The Voting Classifier combines all three tuned models using **soft voting** and achieved the following:

- **Confusion Matrix**: [[38, 4], [2, 70]]
- **Metrics**:
  - Accuracy: `0.9474`
  - Precision: `0.9459`
  - Recall: `0.9722`
  - F1 Score: `0.9589`

---

3. Declaring the Winner

Based on the evaluation metrics:

- **Decision Tree** and **AdaBoost** achieved the highest F1 Score (`0.9589`) and performed equally well.
- **Random Forest** achieved a slightly lower F1 Score (`0.9583`), though still highly competitive.

The **Decision Tree Classifier** is the **simplest** model with the best F1 score and is thus the most efficient in this scenario. However, the **Voting Classifier** is recommended for deployment because it combines multiple models, ensuring robustness and stability across predictions.

---

4. Conclusion

This process demonstrates the effectiveness of hyperparameter tuning using grid search and cross-validation. While the individual models performed well, the ensemble model (Voting Classifier) provides a balanced approach by leveraging the strengths of all models.