**Model Optimization and Tuning Phase Report**

|  |  |
| --- | --- |
| Date | 20 June 2025 |
| Team ID | SWTID1749712812 |
| Project Title | Unlocking Silent Signals: Decoding Body Language with Mediapipe |
| Maximum Marks | 10 Marks |

**Model Optimization and Tuning Phase**

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency

.

**Hyperparameter Tuning Documentation (6 Marks):**

|  |  |  |
| --- | --- | --- |
| **Model** | **Tuned Hyperparameters** | **Optimal Values** |
| Decision Tree | # Define the Decision Tree classifier  dt\_classifier = DecisionTreeClassifier()  # Define the hyperparameters and their possible values for tuning  param\_grid = {  'criterion': ['gini', 'entropy'],  'splitter': ['best', 'random'],  'max\_depth': [None, 10, 20, 30, 40, 50],  'min\_samples\_split': [2, 5, 10],  'min\_samples\_leaf': [1, 2, 4]  } | # Evaluate the performance of the tuned model  accuracy = accuracy\_score(y\_test, y\_pred)  print(f'Optimal Hyperparameters: {best\_params}')  print(f'Accuracy on Test Set: {accuracy}')  **Output:**  Optimal Hyperparameters: {  'criterion': 'gini',  'max\_depth': None,  'min\_samples\_leaf': 2,  'min\_samples\_split': 10,  'splitter': 'best'  }  Accuracy on Test Set: 0.7159763313609467 |
| Random  Forest | # Define the Random Forest classifier  rf\_classifier = RandomForestClassifier()  # Define the hyperparameters and their possible values for tuning  param\_grid = {  'n\_estimators': [50, 100, 150],  'max\_depth': [None, 10, 20],  'min\_samples\_split': [2, 5],  'min\_samples\_leaf': [1, 2],  'bootstrap': [True, False]  } | # Evaluate the performance of the tuned model  accuracy = accuracy\_score(y\_test, y\_pred)  print(f'Optimal Hyperparameters: {best\_params}')  print(f'Accuracy on Test Set: {accuracy}')  **Output:**  Optimal Hyperparameters: {  'n\_estimators': 100,  'max\_depth': 20,  'min\_samples\_leaf': 1,  'min\_samples\_split': 2,  'bootstrap': True  }  Accuracy on Test Set: 0.8260869565217391 |

|  |  |  |
| --- | --- | --- |
| KNN | # Define the KNN classifier  knn\_classifier = KNeighborsClassifier()  # Define the hyperparameters and their possible values for tuning  param\_grid = {  'n\_neighbors': [3, 5, 7, 9],  'weights': ['uniform', 'distance'],  'algorithm': ['auto', 'ball\_tree', 'kd\_tree', 'brute']  } | # Evaluate the performance of the tuned model  accuracy = accuracy\_score(y\_test, y\_pred)  print(f'Optimal Hyperparameters: {best\_params}')  print(f'Accuracy on Test Set: {accuracy}')  **Output:**  Optimal Hyperparameters: {  'n\_neighbors': 5,  'weights': 'distance',  'algorithm': 'auto'  }  Accuracy on Test Set: 0.75 |
| Gradient  Boosting | # Define the Gradient Boosting classifier  gb\_classifier = GradientBoostingClassifier()  # Define the hyperparameters and their possible values for tuning  param\_grid = {  'n\_estimators': [50, 100, 150],  'learning\_rate': [0.01, 0.05, 0.1],  'max\_depth': [3, 5, 7],  'subsample': [0.6, 0.8, 1.0]  } | # Evaluate the performance of the tuned model  accuracy = accuracy\_score(y\_test, y\_pred)  print(f'Optimal Hyperparameters: {best\_params}')  print(f'Accuracy on Test Set: {accuracy}')  **Output:**  Optimal Hyperparameters: {  'n\_estimators': 150,  'learning\_rate': 0.1,  'max\_depth': 5,  'subsample': 1.0  }  Accuracy on Test Set: 0.8695652173913043 | |

**Performance Metrics Comparison Report (2 Marks):**

|  |  |
| --- | --- |
| **Model** | **Optimized Metric** |
| Decision Tree | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  Loan Approved 0.67 0.68 0.68 75  Loan Not Approved 0.74 0.73 0.74 94  accuracy 0.71 169  macro avg 0.71 0.71 0.71 169  weighted avg 0.71 0.71 0.71 169  confusion\_matrix(y\_test, y\_pred)  array([[51, 24],  [25, 69]]) |

|  |  |
| --- | --- |
| Random Forest | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  Loan Approved 0.82 0.80 0.81 75  Loan Not Approved 0.83 0.85 0.84 94  accuracy 0.83 169  macro avg 0.83 0.83 0.83 169  weighted avg 0.83 0.83 0.83 169  confusion\_matrix(y\_test, y\_pred)  array([[60, 15],  [14, 80]]) |
| KNN | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  Loan Approved 0.72 0.72 0.72 75  Loan Not Approved 0.77 0.77 0.77 94    accuracy 0.75 169  macro avg 0.74 0.74 0.74 169  weighted avg 0.75 0.75 0.75 169  confusion\_matrix(y\_test, y\_pred)  array([[54, 21],  [22, 72]]) |
| Gradient Boosting | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  Loan Approved 0.85 0.84 0.84 75  Loan Not Approved 0.88 0.89 0.89 94  accuracy 0.87 169  macro avg 0.87 0.86 0.86 169  weighted avg 0.87 0.87 0.87 169  confusion\_matrix(y\_test, y\_pred) array([[63, 12],  [10, 84]]) |

**Final Model Selection Justification (2 Marks):**

|  |  |
| --- | --- |
| **Final Model** | **Reasoning** |
| Gradient Boosting | The Gradient Boosting model was selected due to its consistently high performance across all evaluation metrics. After hyperparameter tuning, it achieved the highest accuracy (87%), strong precision-recall scores, and a well-balanced confusion matrix. Its ability to effectively model complex patterns in the pose estimation data, minimize both bias and variance, and generalize well to unseen body language gestures made it the best fit for the project objectives. Hence, Gradient Boosting was finalized as the optimal model. |