**Model Development Phase Template**

|  |  |
| --- | --- |
| Date | 19 June 2025 |
| Team ID | SWTID1749712812 |
| Project Title | Unlocking Silent Signals: Decoding Body Language with Mediapipe |
| Maximum Marks | 4 Marks |

**Initial Model Training Code, Model Validation and Evaluation Report**

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

**Initial Model Training Code:**

#importing and building the **random forest** model

def RandomForest(X\_train, X\_test, y\_train, y\_test):

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

y\_tr = model.predict(X\_train)

print(accuracy\_score(y\_tr, y\_train))

yPred = model.predict(X\_test)

print(accuracy\_score(yPred, y\_test))

# printing the train accuracy and test accuracy respectively

RandomForest(X\_train, X\_test, y\_train, y\_test)

#importing and building the **Decision tree** model

def decisionTree(X\_train, X\_test, y\_train, y\_test):

model = DecisionTreeClassifier()

model.fit(X\_train, y\_train)

y\_tr = model.predict(X\_train)

print(accuracy\_score(y\_tr, y\_train))

yPred = model.predict(X\_test)

print(accuracy\_score(yPred, y\_test))

# printing the train accuracy and test accuracy respectively

decisionTree(X\_train, X\_test, y\_train, y\_test)

#importing and building the **KNN** model

def KNN(X\_train, X\_test, y\_train, y\_test):

model = KNeighborsClassifier()

model.fit(X\_train, y\_train)

y\_tr = model.predict(X\_train)

print(accuracy\_score(y\_tr, y\_train))

yPred = model.predict(X\_test)

print(accuracy\_score(yPred, y\_test))

# printing the train accuracy and test accuracy respectively

KNN(X\_train, X\_test, y\_train, y\_test)

#importing and building the Xg boost model

def XGB(X\_train, X\_test, y\_train, y\_test):

model = GradientBoostingClassifier()

model.fit(X\_train, y\_train)

y\_tr = model.predict(X\_train)

print(accuracy\_score(y\_tr, y\_train))

yPred = model.predict(X\_test)

print(accuracy\_score(yPred, y\_test))

# printing the train accuracy and test accuracy respectively

XGB(X\_train, X\_test, y\_train, y\_test)

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Classification Report** | **F1**  **Scor e** | **Confusion Matrix** |
| Random  Forest | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  0.82 0.80 0.81 75  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 | 83% | confusion\_matrix(y\_test,y\_pred)  array([[60, 15], [14, 80]]) |

**Model Validation and Evaluation Report:**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | 71% | confusion\_matrix(y\_test, y\_pred)  array([[51, 24],  [25, 69]]) |
| KNN | print(classification\_report(y\_test, y\_pred))  precision recall f1-score support  Loan Approved 0.60 0.57 0.59 75  Loan Not Approved 0.67 0.69 0.68 94  accuracy 0.64 169  macro avg 0.63 0.63 0.63 169  weighted avg 0.64 0.64 0.64 169 | 64% | confusion\_matrix(y\_test, y\_pred)  array([[43, 32],  [29, 65]]) |
| 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 | 87% | confusion\_matrix(y\_test, y\_pred)  array([[63, 12],  [10, 84]]) |