Model Development Phase Template

Date	28 June 2025
Team ID	SWTID1749634408
Project Title	Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management
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:

Random Forest Classifier:

```
# Define the model
rf = RandomForestClassifier(random_state=0)

# Define the hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 130],
    'max_depth': [5, 8, 11, None],
    'max_features': ['sqrt', 'log2'],
    'min_samples_split': [2, 3, 5],
    'min_samples_leaf': [1, 2, 3],
    'criterion': ['gini', 'entropy']
}

# Grid search
grid_rf = GridSearchCV(rf, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_rf.fit(X_train, y_train)

# Best model
best_rf = grid_rf.best_estimator_
y_pred_rf = best_rf.predict(X_test)
```

Decision Tree Classifier:

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(max_depth=4, min_samples_split=5, random_state=0)
dtc.fit(X_train, y_train)

# accuracy score, confusion matrix and classification report of decision tree

y_pred_dtc = dtc.predict(X_test) # Add this line

dtc_acc = accuracy_score(y_test, y_pred_dtc)

print(f"Training Accuracy of Decision Tree Classifier is {accuracy_score(y_train, dtc.predict(X_train))}")
print(f"Test Accuracy of Decision Tree Classifier is {dtc_acc} \n")

print(f"Confusion Matrix :- \n{confusion_matrix(y_test, y_pred_dtc)}\n")
print(f"Classification Report :- \n {classification_report(y_test, y_pred_dtc)}")
```

KNN:

```
# Define the hyperparameter grid
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

# Grid search with 5-fold cross-validation
grid_knn = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, n_jobs=-1, verbose=1)
grid_knn.fit(X_train, y_train)

# Best model from grid search
best_knn = grid_knn.best_estimator_

# Predictions and evaluation
y_pred_knn = best_knn.predict(X_test)

print(" Best Hyperparameters:", grid_knn.best_params_)
print(f" Training Accuracy of KNN: {accuracy_score(y_train, best_knn.predict(X_train))}")
print(f" Test Accuracy of KNN: {accuracy_score(y_test, y_pred_knn)}\n")

print(" Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
print("\n Classification Report:\n", classification_report(y_test, y_pred_knn))
```

XGBoost Classifier:

```
# Define base model
xgb = XGBClassifier(objective='binary:logistic', use_label_encoder=False, eval_metric='logloss', random_state=0)

# Hyperparameter grid
param_grid = {
    'n_estimators': [100, 150],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.1, 0.3, 0.5],
    'subsample': [0.7, 0.9, 1],
    'colsample_bytree': [0.7, 1],
    'gamma': [0, 1, 5]
}

# Grid Search
grid_xgb = GridSearchCV(xgb, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_xgb.fit(X_train, y_train)

# Best model
best_xgb = grid_xgb.best_estimator_
y_pred_xgb = best_xgb.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_xgb.best_params_)
print(f" Training Accuracy: {accuracy_score(y_train, best_xgb.predict(X_train))}")
print(f" Test Accuracy: {accuracy_score(y_test, y_pred_xgb)}\n")
print(" Confusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
print("\n Classification Report:\n", classification_report(y_test, y_pred_xgb))
```

ADABoost Classifier:

```
# Base estimator (a shallow tree)
base_dtc = DecisionTreeClassifier(random_state=0)

# AdaBoost classifier with base decision tree
ada = AdaBoostClassifier(estimator=base_dtc, random_state=0)

# Parameter grid including base estimator hyperparameters
param_grid = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 1],
    'estimator__max_depth': [1, 2, 3, 4],
    'estimator__min_samples_split': [2, 5],
    'estimator__criterion': ['gini', 'entropy']
}

# Grid search
grid_ada = GridSearchCV(ada, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_ada.fit(X_train, y_train)

# Best model
best_ada = grid_ada.best_estimator_
y_pred_ada = best_ada.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_ada.best_params_)
print(f" Training Accuracy of AdaBoost Classifier (Adaptive Boosting): {accuracy_score(y_train, best_ada.predict(X_train))}")
print(f" Test Accuracy of AdaBoost Classifier (Adaptive Boosting): {accuracy_score(y_test, y_pred_ada)}\n")

print(" Confusion Matrix:\n", confusion_matrix(y_test, y_pred_ada))
print("\n Classification Report:\n", classification_report(y_test, y_pred_ada))
```

Gradient Boost:

```
# Define model
gb = GradientBoostingClassifier(random_state=0)

# Define hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'subsample': [0.8, 1.0]
}

# Grid Search
grid_gb = GridSearchCV(gb, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_gb.fit(X_train, y_train)

# Best model
best_gb = grid_gb.best_estimator_
y_pred_gb = best_gb.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_gb.best_params_)
print(f"Training Accuracy of Gradient Boosting Classifier: {accuracy_score(y_train, best_gb.predict(X_train))}")
print(f"Test Accuracy of Gradient Boosting Classifier: {accuracy_score(y_test, y_pred_gb)}\n")

print(" Confusion Matrix:\n", confusion_matrix(y_test, y_pred_gb))
print("\n Classification Report:\n", classification_report(y_test, y_pred_gb))
```

Stochastic Gradient Boosting:

```
# Define the base model
sgb = GradientBoostingClassifier(random_state=0)

# Define the grid of hyperparameters
param_grid = {
    'n_estimators': [100, 150, 200],
    'learning_rate': [0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5],
    'subsample': [0.7, 0.8, 0.9],
    'max_features': [0.5, 0.75, 1.0]
}

# Grid search
grid_sgb = GridSearchCV(sgb, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_sgb.fit(X_train, y_train)
# Best model
best_sgb = grid_sgb.best_estimator_
y_pred_sgb = best_sgb.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_sgb.best_params_)
print(f" Training Accuracy of Stochastic Gradient Boosting (SGBoost): {accuracy_score(y_train, best_sgb.predict(X_train))}")
print(f" Test Accuracy of Stochastic Gradient Boosting (SGBoost): {accuracy_score(y_test, y_pred_sgb)}\n")

print(" Confusion Matrix:\n", confusion_matrix(y_test, y_pred_sgb))
print("\n Classification Report:\n", classification_report(y_test, y_pred_sgb))
```

CAT Boost Classifier:

```
# Base CatBoost model
cat = CatBoostClassifier(verbose=0, random_state=0)

# Define the grid
param_grid = {
    'iterations': [100, 200],
    'learning_rate': [0.01, 0.05, 0.1],
    'depth': [4, 6, 8],
    'l2_leaf_reg': [1, 3, 5]
}

# Grid Search
grid_cat = GridSearchCV(cat, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_cat.fit(X_train, y_train_enc)

# Best model
best_cat = grid_cat.best_estimator_
y_pred_cat = best_cat.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_cat.best_params_)
print(f" Training Accuracy of CatBoost Classifier: {accuracy_score(y_train_enc, best_cat.predict(X_train))}")
print(f"Test Accuracy of CatBoost Classifier: {accuracy_score(y_test_enc, y_pred_cat)}\n")

print(" Confusion Matrix:\n", confusion_matrix(y_test_enc, y_pred_cat))
print("\n Classification Report:\n", classification_report(y_test_enc, y_pred_cat))
```

Extra Trees Classifier:

```
# Base model
etc = ExtraTreesClassifier(random_state=0)

# Hyperparameter grid
param_grid = {
    'n_estimators': [100, 150],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt', 'log2'],
    'criterion': ['gini', 'entropy']
}

# Grid Search
grid_etc = GridSearchCV(etc, param_grid, cv=5, n_jobs=-1, verbose=1)
grid_etc.fit(X_train, y_train_enc)

# Best model
best_etc = grid_etc.best_estimator_
y_pred_etc = best_etc.predict(X_test)

# Evaluation
print(" Best Hyperparameters:", grid_etc.best_params_)
print(f" Training Accuracy of Extra Trees Classifier: {accuracy_score(y_train_enc, best_etc.predict(X_train))}")
print(f" Test Accuracy of Extra Trees Classifier: {accuracy_score(y_test_enc, y_pred_etc)}\n")

print("Confusion Matrix:\n", confusion_matrix(y_test_enc, y_pred_etc))
print("\n Classification Report:\n", classification_report(y_test_enc, y_pred_etc))
```

Model Validation and Evaluation Report:

Model	Classification Report	F1 Score	Confusion Matrix
Random Forest	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
Decision Tree	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
KNN	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]

XGBoost Classifier	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
ADA Boost Classifier	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
Gradient Boost	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
Stochastic Gradient Boosting	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
CAT Boost Classifier	Classification Report: precision recall f1-score support 0 1.00 0.98 0.99 52 1 0.97 1.00 0.98 28 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80	98%	Confusion Matrix: [[51 1] [0 28]]
Extra Trees Classifier	Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 52 1 1.00 1.00 1.00 28 accuracy 1.00 80 macro avg 1.00 1.00 1.00 80 weighted avg 1.00 1.00 1.00 80	100%	Confusion Matrix: [[52 0] [0 28]]