

Model Optimization and Tuning Phase

Date	19 June 2025
Team ID	SWTID17449620488
Project Title	Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management
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
Logistic Regression	<pre>#Logistic Regression from sklearn.linear_model import LogisticRegression from sklearn.model_selection import GridSearchCV import warnings from sklearn.exceptions import ConvergenceWarning # Suppress convergence warnings warnings.filterwarnings("ignore", category=ConvergenceWarning) param_grid_lr = { 'penalty': ['l1', 'l2'], 'C': [0.01, 0.1, 1, 10], 'solver': ['liblinear', 'saga'] } lr = LogisticRegression(max_iter=2000) grid_search_lr = GridSearchCV(lr, param_grid_lr, cv=5, scoring='accuracy') grid_search_lr.fit(x_train, y_train) print("Best Logistic Regression Params:", grid_search_lr.best_params_) Best Logistic Regression Params: {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}</pre>	<pre>accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy}") Accuracy: 0.925</pre>

Gradient Boosting	<pre>#Gradient Boosting from sklearn.ensemble import GradientBoostingClassifier param_grid_gb = { 'n_estimators': [100, 200], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 5, 7] } gb = GradientBoostingClassifier(random_state=42) grid_search_gb = GridSearchCV(gb, param_grid_gb, cv=5, scoring='accuracy') grid_search_gb.fit(x_train, y_train) print("Best Gradient Boosting Params:", grid_search_gb.best_params_) Best Gradient Boosting Params: {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 100}</pre>	<pre>accuracy_gbc = accuracy_score(y_test, y_pred_gbc) print(f"Accuracy of Gradient Boosting Classifier: {accuracy_gbc}") Accuracy of Gradient Boosting Classifier: 1.0</pre>
Decision Tree	<pre># Decision Tree from sklearn.tree import DecisionTreeClassifier param_grid_dt = { '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] } dt = DecisionTreeClassifier(random_state=42) grid_search_dt = GridSearchCV(dt, param_grid_dt, cv=5, scoring='accuracy') grid_search_dt.fit(x_train, y_train) print("Best Decision Tree Params:", grid_search_dt.best_params_) Best Decision Tree Params: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}</pre>	<pre>accuracy_dtc = accuracy_score(y_test, y_pred_dtc) print(f"Accuracy of Decision Tree Classifier: {accuracy_dtc}") Accuracy of Decision Tree Classifier: 0.975</pre>
Random Forest	<pre># Random Forest from sklearn.ensemble import RandomForestClassifier param_grid_rf = { 'n_estimators': [100, 200], 'max_depth': [None, 10, 20, 30], 'max_features': ['sqrt', 'log2'], 'bootstrap': [True, False] } rf = RandomForestClassifier(random_state=42) grid_search_rf = GridSearchCV(rf, param_grid_rf, cv=5, scoring='accuracy') grid_search_rf.fit(x_train, y_train) print("Best Random Forest Params:", grid_search_rf.best_params_) Best Random Forest Params: {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'n_estimators': 200}</pre>	<pre>accuracy_rfc = accuracy_score(y_test, y_pred_rfc) print(f"Accuracy of Random Forest Classifier: {accuracy_rfc}") Accuracy of Random Forest Classifier: 1.0</pre>

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Logistic Regression	<pre> from sklearn.metrics import classification_report print("\nClassification Report for Logistic Regression:") print(classification_report(y_test, y_pred)) </pre> <pre> Classification Report for Logistic Regression: precision recall f1-score support 0 0.94 0.94 0.94 54 1 0.88 0.88 0.88 26 accuracy 0.93 macro avg 0.91 weighted avg 0.93 </pre> <p>Confusion Matrix:</p> <pre> [[51 3] [3 23]] </pre>
Gradient Boosting	<pre> from sklearn.metrics import classification_report print("\nClassification Report for Gradient Boosting Classifier:") print(classification_report(y_test, y_pred_gbc)) </pre> <pre> Classification Report for Gradient Boosting Classifier: precision recall f1-score support 0 1.00 1.00 1.00 54 1 1.00 1.00 1.00 26 accuracy 1.00 macro avg 1.00 weighted avg 1.00 </pre> <p>Confusion Matrix of Gradient Boosting Classifier:</p> <pre> [[54 0] [0 26]] </pre>

Decision Tree

```
[ ] from sklearn.metrics import classification_report
print("\nClassification Report for Decision Tree Classifier:")
print(classification_report(y_test, y_pred_dtc))
```



```
Classification Report for Decision Tree Classifier:
              precision    recall  f1-score   support

     0           1.00       0.96       0.98         54
     1           0.93       1.00       0.96         26

 accuracy          0.97         0.97         0.97         80
 macro avg          0.96         0.98         0.97         80
 weighted avg       0.98         0.97         0.98         80
```

```
Confusion Matrix of Decision Tree Classifier:
[[52  2]
 [ 0 26]]
```

Random Forest

```
from sklearn.metrics import classification_report
print("\nClassification Report for Random Forest Classifier:")
print(classification_report(y_test, y_pred_rfc))
```



```
Classification Report for Random Forest Classifier:
              precision    recall  f1-score   support

     0           1.00       1.00       1.00         54
     1           1.00       1.00       1.00         26

 accuracy          1.00         1.00         1.00         80
 macro avg          1.00         1.00         1.00         80
 weighted avg       1.00         1.00         1.00         80
```

```
Confusion Matrix of Random Forest Classifier:
[[54  0]
 [ 0 26]]
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	It achieved the highest validation accuracy among all tuned models. It also handles high-dimensional data well, reduces overfitting through bagging, and was optimized using hyperparameter tuning.