Model Optimization and Tuning Phase Template

Date	15 July 2024
Team ID	SWTID1720151584
Project Title	Early Prediction of Chronic Kidney Disease
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr><td>Decision Tree Classifier</td><td>
<pre>
print(classification_report(y_test, y_pred))

Accuracy: 0.95
Precision: 0.9230769230769231
Recall: 0.9230769230769231
F1-Score: 0.9230769230769231
ROC-AUC: 0.9430199430199432
              precision   recall f1-score   support

           0   0.96   0.96   0.96   54
           1   0.92   0.92   0.92   26

    accuracy   0.95   80
   macro avg   0.94   0.94   0.94   80
weighted avg   0.95   0.95   0.95   80
</pre>
</td></tr>
</table>

```
confusion_matrix(y_test,y_pred)
```

```
array([[53,  1],
       [ 1, 25]], dtype=int64)
```

Gradient Boosting Classifier

```
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9625
Precision: 0.96
Recall: 0.9230769230769231
F1-Score: 0.9411764705882353
ROC-AUC: 0.9522792022792023
              precision    recall  f1-score   support

           0       0.96      0.98      0.97        54
           1       0.96      0.92      0.94        26

    accuracy                           0.96        80
   macro avg       0.96      0.95      0.96        80
weighted avg       0.96      0.96      0.96        80
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[53,  1],
       [ 2, 24]], dtype=int64)
```

XG Boost Classifier	<pre>print(classification_report(y_test, y_pred))
	
	Accuracy: 0.95
	Precision: 0.9230769230769231
	Recall: 0.9230769230769231
	F1-Score: 0.9230769230769231
	ROC-AUC: 0.9430199430199432
	              precision recall f1-score support
	
	           0 0.96 0.96 0.96 54
	           1 0.92 0.92 0.92 26
	
	    accuracy 0.95 80
	   macro avg 0.94 0.94 0.94 80
	weighted avg 0.95 0.95 0.95 80
	
	confusion_matrix(y_test,y_pred)
	
	array([[52, 2],
	       [2, 24]], dtype=int64)</pre>

KNN	`print(classification_report(y_test, y_pred))`
	
	`Accuracy: 0.8875`
	`Precision: 0.7575757575757576`
	`Recall: 0.9615384615384616`
	`F1-Score: 0.847457627118644`
	`ROC-AUC: 0.9066951566951568`
	
	`              precision    recall  f1-score   support`
	
	`           0       0.98      0.85      0.91        54`
	`           1       0.76      0.96      0.85        26`
	
	`    accuracy                           0.89        80`
	`   macro avg       0.87      0.91      0.88        80`
	`weighted avg       0.91      0.89      0.89        80`
	
	`confusion_matrix(y_test,y_pred)`
	
	`array([[53,  1],`
	`       [ 1, 25]], dtype=int64)`

<table>
<tr><td>Random Forest Classifier</td><td>
<pre>
print(classification_report(y_test, y_pred))
</pre>
<pre>
Accuracy: 0.975
Precision: 0.9615384615384616
Recall: 0.9615384615384616
F1-Score: 0.9615384615384616
ROC-AUC: 0.9715099715099716
              precision   recall f1-score   support

           0   0.98   0.98   0.98   54
           1   0.96   0.96   0.96   26

    accuracy   0.97   80
   macro avg   0.97   0.97   0.97   80
weighted avg   0.97   0.97   0.97   80
</pre>
<pre>
confusion_matrix(y_test,y_pred)
</pre>
<pre>
array([[53, 1],
       [1, 25]], dtype=int64)
</pre>
</td></tr>
<tr><td>Logistic Regression</td><td>
<pre>
confusion_matrix(y_test,y_pred)
</pre>
<pre>
array([[52, 2],
       [2, 24]], dtype=int64)
</pre>
</td></tr>
</table>

```
print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        54
           1       0.84      1.00      0.91        26

    accuracy                           0.94        80
   macro avg       0.92      0.95      0.93        80
weighted avg       0.95      0.94      0.94        80
```

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

<table>
<tr>
<td>KNN</td>
<td>

Creating a KNeighborsClassifier with initial hyperparameters

```python
knn = KNeighborsClassifier(
    n_neighbors=5,
    weights='uniform',
    algorithm='auto',
    leaf_size=30,
    p=2,
    metric='minkowski',
    n_jobs=None
)
```

</td>
<td>

```python
# Defining the parameter grid for tuning
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11],
    'weights': ['uniform', 'distance'],
    'algorithm': ['auto', 'ball_tree','kd_tree','brute'],
    'leaf_size': [20, 30, 40, 50],
    'p': [1, 2]
}
```

</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>

Creating a LogisticRegression model with initial hyperparameters

```python
mo = LogisticRegression(
    penalty='l2',
    C=1.0,
    solver='lbfgs',
    max_iter=100,
    random_state=42
)
```

</td>
<td>

```python
# Defining the parameter grid for tuning
param_grid = {
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],
    'C': [0.01, 0.1, 1.0, 10, 100],
    'solver': ['lbfgs', 'liblinear', 'saga'],
    'max_iter': [100, 200, 300]
}
```

</td>
</tr>
<tr>
<td>XGBoost Classifier</td>
<td>

Creating an XGBClassifier with initial hyperparameters

```python
xg = xgb.XGBClassifier(
    objective='binary:logistic',
    learning_rate=0.1,
    n_estimators=100,
    max_depth=3,
    min_child_weight=1,
    subsample=1.0,
    colsample_bytree=1.0,
    random_state=42
)
```

</td>
<td>

```python
# Defining the parameter grid for tuning
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.8, 0.9, 1.0],
    'colsample_bytree': [0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2],
    'reg_alpha': [0, 0.01, 0.1],
    'reg_lambda': [1, 1.5, 2]
}
```

</td>
</tr>
</table>

<table>
<tr>
<td>GradientBoostingClassifier</td>
<td>

Creating a GradientBoosting classifier with initial hyperparameters

```
gra = GradientBoostingClassifier(
    loss='deviance',            # Los
    learning_rate=0.1,          # Lea
    n_estimators=100,           # Num
    subsample=1.0,              # Fra
    criterion='friedman_mse',   # Fun
    min_samples_split=2,        # Min
    min_samples_leaf=1,         # Min
    max_depth=3,                # Max
    random_state=42             # Ran
)
```

</td>
<td>

```
# Defining the parameter grid for tuning
param_grid = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.01, 0.1, 0.2],
    'n_estimators': [100, 200, 300],
    'subsample': [0.8, 0.9, 1.0],
    'criterion': ['friedman_mse', 'mse', 'mae'],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_depth': [3, 5, 7],
    'max_features': ['sqrt', 'log2', None]
}
```

</td>
</tr>
<tr>
<td>Decision Tree Classifier</td>
<td>

```
model2 = DecisionTreeClassifier(
    criterion='gini',            #
    splitter='best',             #
    max_depth=None,              #
    min_samples_split=2,         #
    min_samples_leaf=1,          #
    max_features=None,           #
    random_state=42              #
)
```

Creating a DecisionTree classifier with initial hyperparameters

</td>
<td>

```
# Defining the parameter grid for tuning
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2'],
}
```

</td>
</tr>
<tr>
<td>Random Forest Classifier</td>
<td>

```
model1 = RandomForestClassifier(
    n_estimators=100,          # Nu
    max_depth=None,            # Ma
    min_samples_split=2,       # Mi
    min_samples_leaf=1,        # Mi
    max_features='sqrt',       # Th
    bootstrap=True,            # Wh
    random_state=42            # Ra
)
```

Creating a RandomForest classifier with initial hyperparameters

</td>
<td>

```
# Defining the parameter grid for tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', 0.2],
    'bootstrap': [True, False]
}
```

</td>
</tr>
</table>

<table>
<tr>
<td>Ada Boost Classifier</td>
<td>
<pre>ada = AdaBoostClassifier(
    estimator=DecisionTreeClassifier(max_depth=3),
    n_estimators=100,   # Number of weak learne
    learning_rate=0.1,   # Learning rate
    algorithm='SAMME.R',   # Algorithm to use: 'SA
    random_state=42   # Random seed for repro
)</pre>
Creating an AdaBoost classifier with a stronger base estimator
</td>
<td>
<pre># Defining the parameter grid for tuning
param_grid = {
    'estimator__max_depth': [3, 5, 7],
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 1.0],
    'algorithm': ['SAMME', 'SAMME.R']
}</pre>
</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Ada Boost Classifier	The model Ada Booster was selected for its performance high accuracy during hyperparameter tuning .Its ability to handle complex relationships, minimize overfitting, high accuracy justifying the selection as the final model.