Model Optimization and Tuning Phase Template

Date	12 July 2024
Team ID	SWTID1720083491
Project Title	Early Prediction of Chronic Kidney Disease Using Machine Learning
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	# Define the logistic regression model model = LogisticRegression(max_iter=5000) # Define the parameter grid for tuning param_grid = { 'C': [0.001, 0.01, 0.1, 1, 10, 100], # Regularization parameter 'solver': ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga'] # Solvers to try }	# Print the best parameters found print("Best parameters:", best_params) Best parameters: {'C': 100, 'solver': 'lbfgs'}
Decision Tree	# Define the parameter grid for GridSearchCV param_grid = { "max_depth": [2, 3, 4, 5], "min_samples_split": [2, 5, 10], "min_samples_leaf": [1, 2, 4], "max_features": ["auto", "sqrt"] # Different options for feature selection }	# Print the best parameters print("Best parameters:", best_params) Best parameters: {'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}

<table>
<tr>
<td>Random Forest</td>
<td>
<pre>
Define the Random Forest model
rf_model = RandomForestClassifier()

Define the hyperparameter grid
param_grid = {
    'n_estimators': [100, 200, 300], # Number of trees in the forest
    'max_depth': [4, 6, 8], # Maximum depth of each tree
    'min_samples_split': [2, 5, 10], # Minimum samples to split a node
    'min_samples_leaf': [1, 2, 4], # Minimum samples at each leaf node
    'max_features': ['auto', 'sqrt', 'log2'] # Number of features considered at each split
}
</pre>
</td>
<td>
<pre>
Print the best parameters found
print("Best parameters:", best_params)
</pre>
Best parameters: {'n_estimators': 200, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'log2', 'max_depth': 8}
</td>
</tr>
<tr>
<td>SVM</td>
<td>
<pre>
Define the parameter grid for GridSearchCV
param_grid = {
    "C": [0.1, 1, 10], # Regularization parameter
    "kernel": ["linear", "rbf"], # Kernel function
    "gamma": [0.01, 0.1, 1], # Kernel coefficient (for rbf)
}
</pre>
</td>
<td>
<pre>
Print the best parameters
print("Best parameters:", best_params)

Use the best model for prediction
... (Similar to decision tree example)
</pre>
<pre>
Best parameters: {'C': 1, 'gamma': 0.01, 'kernel': 'linear'}
</pre>
</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>Logistic
Regression</td>
<td>
<pre>
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test,y_pred))
</pre>
<pre>
1.0
              precision   recall f1-score   support

           0   1.00   1.00   1.00   27
           1   1.00   1.00   1.00   53

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

[[27 0]
 [0 53]]
</pre>
</td>
</tr>
</table>

Random Forest

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

```
Accuracy: 1.0
Confusion Matrix:
 [[24  0]
 [ 0 56]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        24
           1       1.00      1.00      1.00        56

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

Decision Tree

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

```
1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        27
           1       1.00      1.00      1.00        53

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

[[27  0]
 [ 0 53]]
```

SVM

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

	`` 0.9875 precision recall f1-score support 0 0.96 1.00 0.98 27 1 1.00 0.98 0.99 53 accuracy 0.99 80 macro avg 0.98 0.99 0.99 80 weighted avg 0.99 0.99 0.99 80 [[27 0] [1 52]] ```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Logistic Regression	Gives promising accuracy & performance and does not overfit or underfit even after cv=10. This model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.