UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SCHOOL OF COMPUTING

SECB4313-01

BIOINFORMATICS MODELING AND SIMULATION

ASSIGNMENT 3 - DISCOVERING A WELL-PERFORMING MODEL CONFIGURATION FOR THE HEART DISEASE DATASET USING HYPERPARAMETER OPTIMIZATION ALGORITHM (GRIDSEARCH AND RANDOM SEARCH)

LECTURER:

DR. AZURAH BINTI A SAMAH

GROUP MEMBERS:

SHAHRIL BIN SAIFUL BAHRI (A20EC0144)

MUHAMMAD AIMAN BIN ABDUL RAZAK (A20EC0082)

NAVINTHRA RAO A/L VENKATAKUMAR (A20EC0104)

1. Summary of combination of hyper parameters from Assignment 2

Based on the previous assignment 2, there are 4 hyperparameters that were selected. The hyperparameters are number of trees (n_estimators), maximum depth of trees (max_depth), minimum samples split (min_samples_split) and minimum samples leaf (min_samples_leaf). Based on the previous experiment or assignment 2, the best combination of hyperparameters that was obtained is from experiment number 10 with n_estimators of 200, max_depth of 10, min_samples_split of 2, min_samples_leaf of 4 with the score of 0.81394 and accuracy of 0.85245. The n_estimators of 200 are set to improve model performance by reducing variance (the model's sensitivity to small changes in the data). The value of 200 is a good starting point for finding a balance between accuracy and efficiency. The max_depth with the value of 10 restricts how deeply each individual tree can grow that captures more complex patterns in the data while avoiding overfitting. The min_samples_split is set to 2 that controls the minimum number of samples required to split a node in the tree where lower values allow for more complex trees with more splits. Finally, the min_samples_leaf is fixed to 4 to allow exploration of data that is needed at the leaf level to achieve good model performance. All these hyperparameters are set to find the best configuration of the model that balances accuracy, complexity, and computational efficiency.

2. Output for Grid Search and Random Search

a) Grid Search

```
Grid Search Best Parameters: {'max_depth': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 200}
Grid Search Best Cross-Validation Accuracy: 0.8139455782312925
Grid Search Test Accuracy: 0.8524590163934426
Grid Search Computational Time: 34.95 seconds
```

Grid Search Best Parameter :

'max_depth': 10
'min_samples_leaf': 4
'min_samples_split': 2
'n_estimators': 200

Grid Search Best Cross-Validation Accuracy: 0.8139455782312925
Grid Search Test Accuracy: 0.8524590163934426

b) Random Search

```
Random Search Best Parameters: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 4, 'max_depth': 10}
Random Search Best Cross-Validation Accuracy: 0.8139455782312925
Random Search Test Accuracy: 0.8524590163934426
Random Search Computational Time: 14.83 seconds
```

Random Search Best Parameter :

'max_depth': 10
'min_samples_leaf': 4
'min_samples_split': 2

'n_estimators': 200

Random Search Best Cross-Validation Accuracy: 0.8139455782312925
Random Search Test Accuracy: 0.8524590163934426

3. Results

Grid Search and Random Search were used to tune the hyperparameters of a machine learning model applied to the heart disease dataset. Grid Search exhaustively searches through a predefined grid of hyperparameters, evaluating the model's performance at each point, while Random Search randomly selects combinations of hyperparameters to evaluate.

In terms of computational time, Grid Search took longer, which is 34.95 seconds, compared to Random Search, which is 14.83 seconds. This is due to the fact that Grid Search evaluates every possible combination within the specified parameter grid, which can be time-consuming, especially for large parameter spaces. On the other hand, Random Search explores a smaller subset of the parameter space, resulting in faster computational time.

4. Discussion

Hyperparameter tuning is essential for maximising model performance in machine learning. By fine-tuning hyperparameters like learning rates, regularisation strengths, or model complexities, we can strike a balance between accuracy and generalisation. This process optimises computational resources, ensuring efficient usage while adapting the model to dataset characteristics. In conclusion, hyperparameter tuning enhances predictive accuracy, enabling models to make more informed decisions and improvements in various real-world scenarios.

5. Appendix

Grid Search

```
# Perform GridSearchCV
start_time_grid = time.time()
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
end_time_grid = time.time()

# Get the best parameters and best score from Grid Search
best_params_grid = grid_search.best_params_
best_score_grid = grid_search.best_score_

# Test the best model from Grid Search on the test set
best_model_grid = grid_search.best_estimator_
y_pred_grid = best_model_grid.predict(X_test)
test_accuracy_grid = accuracy_score(y_test, y_pred_grid)

# Print Grid Search results
print("Grid Search Best Parameters:", best_params_grid)
print("Grid Search Best Cross-Validation Accuracy:", best_score_grid)
print("Grid Search Test Accuracy:", test_accuracy_grid)
print("Grid Search Computational Time: {:.2f} seconds".format(end_time_grid - start_time_grid))
```

Random Search

```
# Perform RandomizedSearchCV
start_time_random = time.time()
random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_grid, n_iter=10, cv=5, scoring='accuracy', random_state=42)
random_search.fit(X_train, y_train)
end_time_random = time.time()

# Get the best parameters and best score from Random Search
best_params_random = random_search.best_params_
best_score_random = random_search.best_score_

# Test the best model from Random Search on the test set
best_model_random = random_search.best_estimator_
y_pred_random = best_model_random.predict(X_test)
test_accuracy_random = accuracy_score(y_test, y_pred_random)

# Print Random Search results
print("Random Search Best Parameters:", best_params_random)
print("Random Search Best Cross-Validation Accuracy:", best_score_random)
print("Random Search Test Accuracy:", test_accuracy_random)
print("Random Search Computational Time: {:.2f} seconds".format(end_time_random - start_time_random))
```