Model Optimization and Tuning Phase Report

Date	23 April 2024
Team ID	Team-738178
Project Title	Envisioning Success : Predicting University Scores With 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):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Decision Tree</td><td><pre>#Decision Trees:
from sklearn.tree import DecisionTreeRegressor

Define the model
dt = DecisionTreeRegressor()

Define hyperparameters to tune
param_grid = {'max_depth': [None, 5, 10, 20], 'min_samples_split': [2, 5, 10]}

Perform GridSearchCV
grid_search_dt = GridSearchCV(dt, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search_dt.fit(X_train, y_train)

Get the best hyperparameters and model
best_params_dt = grid_search_dt.best_params_
best_model_dt = grid_search_dt.best_estimator_</pre></td><td><pre>print("Decision Tree Performance:")
print(f'Optimal Hyperparameters: {best_params_dt}')
print(f'Mean Squared Error on Test Set: {dt_mse}')

Decision Tree Performance:
Optimal Hyperparameters: {'max_depth': None, 'min_samples_split': 2}
Mean Squared Error on Test Set: 2.9889272727272727</pre></td></tr>
<tr><td>Random

Forest</td><td><pre>#Random Forests:
from sklearn.ensemble import RandomForestRegressor

Define the model
rf = RandomForestRegressor()

Define hyperparameters to tune
param_grid = {'n_estimators': [100, 200, 300], 'max_depth': [None, 5, 10], 'min_samples_split': [2, 5, 10]}

Perform GridSearchCV
grid_search_rf = GridSearchCV(rf, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search_rf.fit(X_train, y_train)

Get the best hyperparameters and model
best_params_rf = grid_search_rf.best_params_
best_model_rf = grid_search_rf.best_estimator_</pre></td><td><pre>print("Random Forest Performance:")
print(f'Optimal Hyperparameters: {best_params_rf}')
print(f'Mean Squared Error on Test Set: {rf_mse}')

Random Forest Performance:
Optimal Hyperparameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 300}
Mean Squared Error on Test Set: 1.637340784825918</pre></td></tr>
</table>

SVR	-	-
Linear Regression	-	-
Lasso Regression	`from sklearn.linear_model import Lasso` `from sklearn.model_selection import GridSearchCV` `lasso_reg = Lasso()` `param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10]}` `grid_search_lasso = GridSearchCV(lasso_reg, param_grid, cv=5, scoring='neg_mean_squared_error')` `grid_search_lasso.fit(X_train, y_train)` `best_params_lasso = grid_search_lasso.best_params_`	`print("Lasso Regression Performance:")` `print(f'Optimal Hyperparameters: {best_params_lasso}')` `print(f'Mean Squared Error on Test Set: {lasso_mse}')` `Lasso Regression Performance:` `Optimal Hyperparameters: {'alpha': 1}` `Mean Squared Error on Test Set: 28.893569757635724`

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	`# Printing results` `print("Prediction Evaluation using Decision Tree:")` `print("MAE:", dt_mae)` `print("MSE:", dt_mse)` `print("RMSE:", dt_rmse)` `print("R-squared:", dt_r2)` `Prediction Evaluation using Decision Tree:` `MAE: 0.7953636363636363` `MSE: 3.165202727272727` `RMSE: 1.7791016629953238` `R-squared: 0.941231324579233` `# Printing actual and predicted values` `print("Actual value:", y_actual)` `print("Predicted value:", y_pred_dt[0])` `Actual value: 100` `Predicted value: 100.0`

<table>
<tr><td>Random Forest</td><td>

```
# Printing results
print("Prediction Evaluation using Random Forest:")
print("MAE:", rf_mae)
print("MSE:", rf_mse)
print("RMSE:", rf_rmse)
print("R-squared:", rf_r2)
print("\n")
```

```
Prediction Evaluation using Random Forest:
MAE: 0.5947518939393951
MSE: 1.6870365632197004
RMSE: 1.2988597165281939
R-squared: 0.9686766021801541
```

```
# Printing actual and predicted values
print("Actual value:", y_actual)
print("Predicted value:", y_pred_rf[0])
```

```
Actual value: 100
Predicted value: 99.42905833333333
```

</td></tr>
<tr><td>SVR</td><td>

```
# Printing results
print("Prediction Evaluation using SVR:")
print("MAE:", svr_mae)
print("MSE:", svr_mse)
print("RMSE:", svr_rmse)
print("R-squared:", svr_r2)
print("\n")
```

```
Prediction Evaluation using SVR:
MAE: 1.7292341972937126
MSE: 26.883723937063873
RMSE: 5.184951681266073
R-squared: 0.50084687070893
```

```
# Printing actual and predicted values
print("Actual value:", y_actual)
print("Predicted value:", y_pred_svr[0])
```

```
Actual value: 100
Predicted value: 60.02460149545989
```

</td></tr>
<tr><td>Linear Regression</td><td>

```
# Printing results
print("Prediction Evaluation using Linear Regression:")
print("MAE:", lr_mae)
print("MSE:", lr_mse)
print("RMSE:", lr_rmse)
print("R-squared:", lr_r2)
print("\n")
```

```
Prediction Evaluation using Linear Regression:
MAE: 2.6657340636132827
MSE: 28.917809410716295
RMSE: 5.377528187812342
R-squared: 0.4630797766933825
```

```
# Printing actual and predicted values
print("Actual value:", y_actual)
print("Predicted value:", y_pred_lr[0])
```

```
Actual value: 100
Predicted value: [63.33166471]
```

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

Lasso Regression	``` # Printing results print("Prediction Evaluation using Lasso Regression:") print("MAE:", lasso_mae) print("MSE:", lasso_mse) print("RMSE:", lasso_rmse) print("R-squared:", lasso_r2) print("\n") ``` ``` Prediction Evaluation using Lasso Regression: MAE: 2.6604781238340274 MSE: 28.893569757635724 RMSE: 5.3752739239629195 R-squared: 0.4635298370613741 ``` ``` # Printing actual and predicted values print("Actual value:", y_actual) print("Predicted value:", y_pred_lasso[0]) ``` ``` Actual value: 100 Predicted value: 62.96954350809409 ```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The Gradient Boosting 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.