Model Optimization and Tuning Phase Report

Date	7 July 2024
Team ID	6
Project Title	Abalone Age Prediction
Maximum Marks	10 Marks

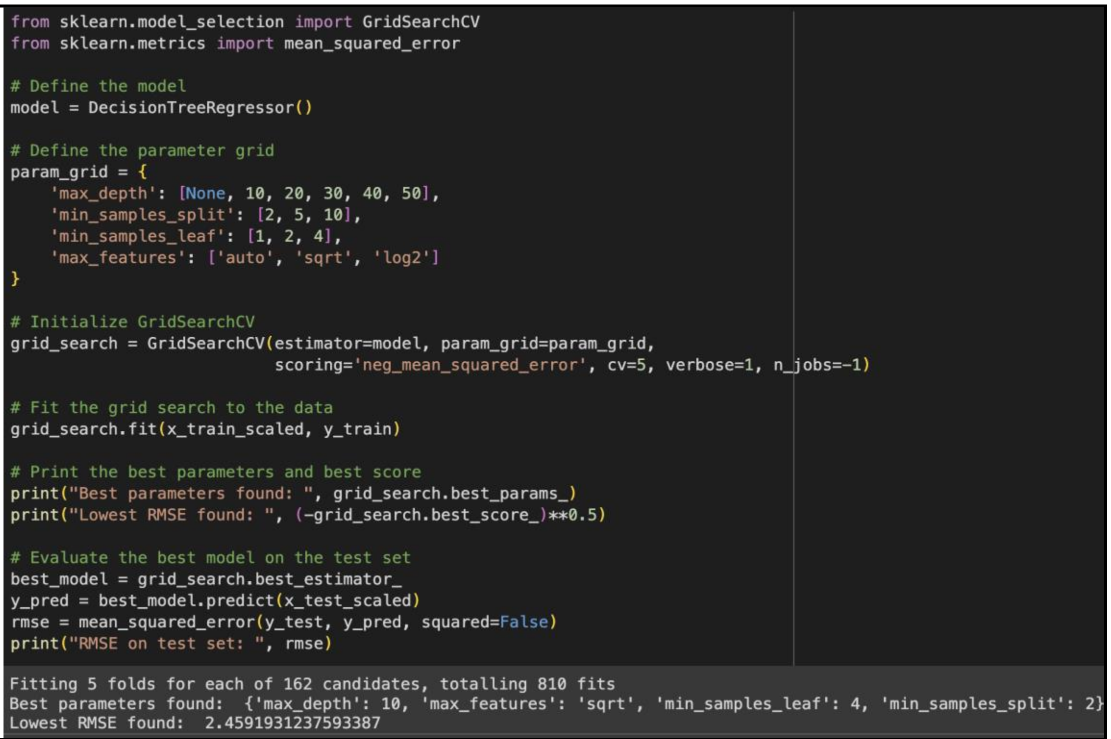
Model Optimization and Tuning Phase

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**

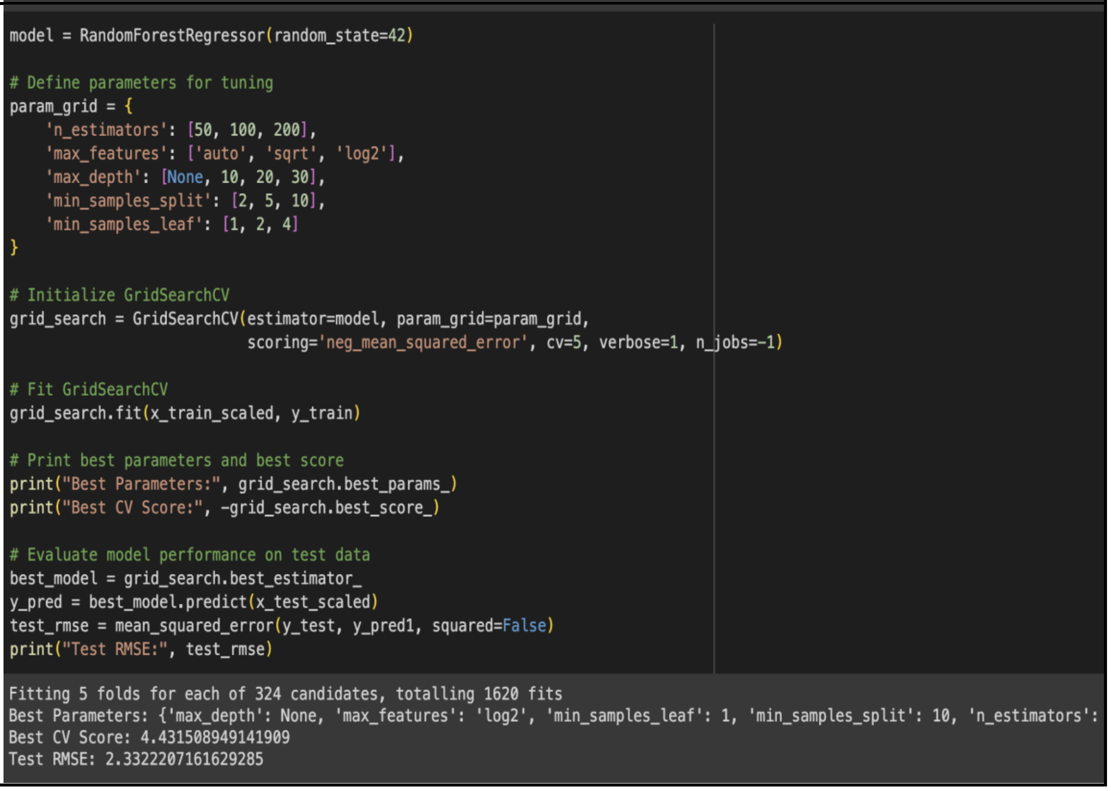
Decision Tree



```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Define the model
model = DecisionTreeRegressor()

# Define the parameter grid
param_grid = {
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=-1)

# Fit the grid search to the data
grid_search.fit(x_train_scaled, y_train)

# Print the best parameters and best score
print("Best parameters found: ", grid_search.best_params_)
print("Lowest RMSE found: ", (-grid_search.best_score_)**0.5)

# Evaluate the best model on the test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(x_test_scaled)
rmse = mean_squared_error(y_test, y_pred, squared=False)
print("RMSE on test set: ", rmse)
```

```
Fitting 5 folds for each of 162 candidates, totalling 810 fits
Best parameters found:  {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2}
Lowest RMSE found:  2.4591931237593387
```

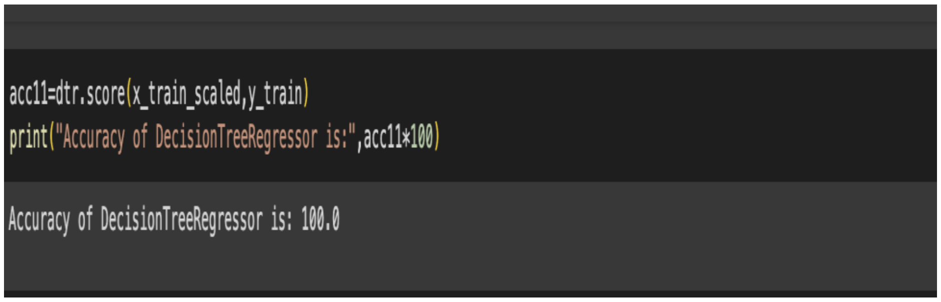
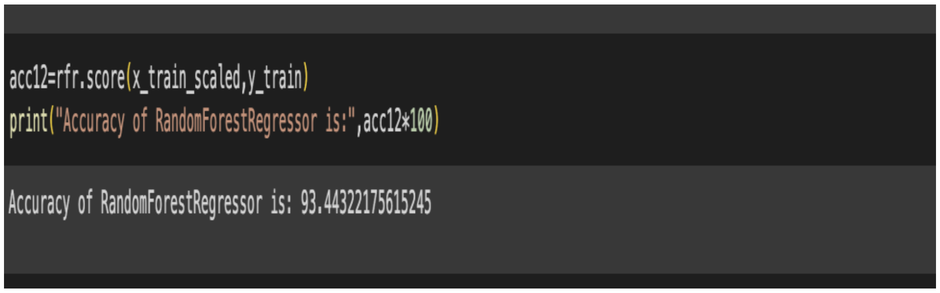
Random Forest



```
model = RandomForestRegressor(random_state=42)

# Define parameters for tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, verbose=1, n_jobs=-1)

# Fit GridSearchCV
grid_search.fit(x_train_scaled, y_train)

# Print best parameters and best score
print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", -grid_search.best_score_)

# Evaluate model performance on test data
best_model = grid_search.best_estimator_
y_pred = best_model.predict(x_test_scaled)
test_rmse = mean_squared_error(y_test, y_pred1, squared=False)
print("Test RMSE:", test_rmse)
```

```
Fitting 5 folds for each of 324 candidates, totalling 1620 fits
Best Parameters: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators':
Best CV Score: 4.431508949141909
Test RMSE: 2.3322207161629285
```

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	 `acc11=dtr.score(x_train_scaled,y_train)` `print("Accuracy of DecisionTreeRegressor is:",acc11*100)` `Accuracy of DecisionTreeRegressor is: 100.0`
Random Forest	 `acc12=rfr.score(x_train_scaled,y_train)` `print("Accuracy of RandomForestRegressor is:",acc12*100)` `Accuracy of RandomForestRegressor is: 93.44322175615245`

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Decision Tree Regressor	The Decision Tree Regressor 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.