Model Optimization and Tuning Phase

Date	15 April 2024
Team ID	Team - 738203
Project Title	Share Price Estimation Of TOP 5 GPU Companies
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

Hyperparameter Tuning Documentation (6 Marks):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Linear Regression</td><td><pre>
# Define the parameter grid to search over for Linear Regression
param_grid_lr = {
    'linearregression__fit_intercept': [True, False],
    'linearregression__copy_X': [True, False]
}

# Create a pipeline with preprocessing (StandardScaler) and Linear Regression
lr_pipeline = Pipeline([
    ('scaler', StandardScaler()), # Add any preprocessing steps here
    ('linearregression', LinearRegression())
])

# Instantiate GridSearchCV to find the best hyperparameters
grid_search_lr = GridSearchCV(estimator=lr_pipeline, param_grid=param_grid_lr, scoring='neg_mean_squared_error', cv=5, verbose=2, n_jobs=-1)

# Fit the grid search to the data
grid_search_lr.fit(x_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters for Linear Regression:", grid_search_lr.best_params_)

# Get the best model
best_lr_model = grid_search_lr.best_estimator_
</pre></td><td><pre>
Fitting 5 folds for each of 4 candidates, totalling 20 fits
Best Hyperparameters for Linear Regression: {'linearregression__copy_X': True, 'linearregression__fit_intercept': True}
</pre></td></tr>
<tr><td>Decision Tree</td><td><pre>
# Define the parameter grid to search over for Decision Tree
param_grid_dt = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the Decision Tree model
dt = DecisionTreeRegressor(random_state=42)

# Instantiate GridSearchCV to find the best hyperparameters
grid_search_dt = GridSearchCV(estimator=dt, param_grid=param_grid_dt, scoring='neg_mean_squared_error', cv=5, verbose=2, n_jobs=-1)

# Fit the grid search to the data
grid_search_dt.fit(x_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters for Decision Tree:", grid_search_dt.best_params_)

# Get the best model
best_dt_model = grid_search_dt.best_estimator_
best_dt_model
</pre></td><td><pre>
Fitting 5 folds for each of 36 candidates, totalling 180 fits
Best Hyperparameters for Decision Tree: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 5}

DecisionTreeRegressor
DecisionTreeRegressor(max_depth=10, min_samples_leaf=2, min_samples_split=5,
                      random_state=42)
</pre></td></tr>
</table>

<table>
<tr>
<td>Extra Tree</td>
<td><pre><code># Define the parameter grid to search over for Extra Trees
param_grid_etr = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the Extra Trees model
etr = ExtraTreesRegressor(random_state=42)

# Instantiate GridSearchCV to find the best hyperparameters
grid_search_etr = GridSearchCV(estimator=etr, param_grid=param_grid_etr, scoring='neg_mean_squared_error', cv=5, verbose=2, n_jobs=-1)

# Fit the grid search to the data
grid_search_etr.fit(x_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters for Extra Trees:", grid_search_etr.best_params_)

# Get the best model
best_etr_model = grid_search_etr.best_estimator_</code></pre></td>
<td><pre><code>Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best Hyperparameters for Extra Trees: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}</code></pre></td>
</tr>
<tr>
<td>Random Forest</td>
<td><pre><code># Define the parameter grid to search over for Random Forest
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Create the Random Forest model
rf = RandomForestRegressor(random_state=42)

# Instantiate GridSearchCV to find the best hyperparameters
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, scoring='neg_mean_squared_error', cv=5, verbose=2, n_jobs=-1)

# Fit the grid search to the data
grid_search_rf.fit(x_train, y_train)

# Print the best hyperparameters
print("Best Hyperparameters for Random Forest:", grid_search_rf.best_params_)

# Get the best model
best_rf_model = grid_search_rf.best_estimator_</code></pre></td>
<td><pre><code>Fitting 5 folds for each of 81 candidates, totalling 405 fits
Best Hyperparameters for Random Forest: {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}</code></pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr>
<th>Model</th>
<th>Optimized Metric</th>
</tr>
<tr>
<td>Linear Regression</td>
<td><pre><code>Hyper-tuned Linear Regression Model:
R-squared (R2) Score: 0.9998367000912862
Mean Absolute Error (MAE): 0.9166458477303411
Mean Squared Error (MSE): 2.9316225652417143
Root Mean Squared Error (RMSE): 1.712198167631806</code></pre></td>
</tr>
<tr>
<td>Decision Tree</td>
<td><pre><code>Decision Tree Model Evaluation:
Mean Squared Error (MSE): 510.72920925764066
Root Mean Squared Error (RMSE): 22.599318778618983
Mean Absolute Error (MAE): 4.297187979147489
R2 Score after hyperparmeter tunning 0.971550896681562</code></pre></td>
</tr>
</table>

Extra Tree	`Extra Trees Model Evaluation:`
	`Mean Squared Error (MSE): 488.7417236390516`
	`Root Mean Squared Error (RMSE): 22.10750378579752`
	`Mean Absolute Error (MAE): 3.006387086974478`
	`R-squared (R2) Score: 0.9727756636201618`

Random Forest	`Random Forest Model Evaluation:`
	`Mean Squared Error (MSE): 527.4793941810102`
	`Root Mean Squared Error (RMSE): 22.966919562296773`
	`Mean Absolute Error (MAE): 4.128952048218431`
	`R-squared (R2) Score: 0.9706178626336749`

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Linear Regression	The Linear Regression model was chosen for its remarkable performance, showcasing exceptional accuracy during the model selection process. Its simplicity coupled with its ability to capture linear relationships between variables efficiently makes it an ideal choice for the project's objectives. By mitigating overfitting through regularization techniques and fine-tuning model parameters, solidifies its position as the optimal choice for meeting the project's requirements.