Model Optimization and Tuning Phase Report

Date	23 September 2024
Team ID	LTVIP2024TMID24986
Project Title	Movie Box Office Gross Prediction 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

Linear Regression	`from sklearn.linear_model import LinearRegression`
	`from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score`
	
	`def linear_regression(X_train, X_test, y_train, y_test):`
	`    lr = LinearRegression()`
	`    lr.fit(X_train, y_train)`
	`    y_pred = lr.predict(X_test)`
	`    print("**Linear Regression**")`
	`    print("Mean Absolute Error: ", mean_absolute_error(y_test, y_pred))`
	`    print("Mean Squared Error: ", mean_squared_error(y_test, y_pred))`
	`    print("R2 Score: ", r2_score(y_test, y_pred))`
		Suggested code may be subject to a licence | varchanaiyer/weather_app_wwcode
		`linear_regression(x_train, x_test, y_train, y_test)`
		
		`**Linear Regression**`
		`Mean Absolute Error:  54.37754333076527`
		`Mean Squared Error:  8649.147282950133`
		`R2 Score:  0.7758003459046133`

Ridge regression	`def ridge_regression_with_tuning(X_train, X_test, y_train, y_test):`
	`    param_grid = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}  # Regularization strength`
	
	`    ridge = Ridge()`
	`    grid_search = GridSearchCV(estimator=ridge, param_grid=param_grid, cv=5, scoring='r2')`
	`    grid_search.fit(X_train, y_train)`
	
	`    best_ridge = grid_search.best_estimator_`
	`    y_pred = best_ridge.predict(X_test)`
	
	`    print("Best Ridge Hyperparameters:", grid_search.best_params_)`
		`ridge_regression_with_tuning(x_train, x_test, y_train, y_test)`
		
		`Best Ridge Hyperparameters: {'alpha': 100}`
		`R2 Score:  0.7767821907211369`
		`Mean Absolute Error:  54.220345539504514`
		`Mean Squared Error:  8611.26979174089`

<table>
<tr>
<td>SVM regression</td>
<td><pre>
def svr_with_tuning(X_train, X_test, y_train, y_test):
    param_grid = {
        'C': [0.1, 1, 10, 100],   # Regularization parameter
        'epsilon': [0.001, 0.01, 0.1, 1], # Epsilon-tube within which no penalty is ass
        'kernel': ['linear', 'rbf']   # Type of kernel
    }

    svr = SVR()
    grid_search = GridSearchCV(estimator=svr, param_grid=param_grid, cv=5, scoring='r2')
    grid_search.fit(X_train, y_train)

    best_svr = grid_search.best_estimator_
</pre></td>
<td><pre>
Best SVR Hyperparameters: {'C': 100, 'epsilon': 0.1, 'kernel': 'linear'}
R2 Score: 0.7367468556578113
Mean Absolute Error: 52.433519903226696
Mean Squared Error: 10155.748131291039
</pre></td>
</tr>
<tr>
<td>Random Forest</td>
<td><pre>
def random_forest_regressor_with_tuning(X_train, X_test, y_train, y_test):
    param_grid = {
        'n_estimators': [100, 200, 300],   # Number of trees in the forest
        'max_depth': [10, 20, 30, None],   # Maximum depth of the tree
        'min_samples_split': [2, 5, 10],   # Minimum number of samples required to s
        'min_samples_leaf': [1, 2, 4],   # Minimum number of samples required at a
        'bootstrap': [True, False]   # Whether bootstrap samples are used wher
    }

    rf = RandomForestRegressor()
    grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, scoring='r2')
    grid_search.fit(X_train, y_train)

    best_rf = grid_search.best_estimator_
    y_pred = best_rf.predict(X_test)
</pre></td>
<td><pre>
  return fit_method(estimator, *args, **kwargs)
Best Random Forest Hyperparameters: {'bootstrap': True, 'max_dept
R2 Score: 0.7480565218845108
Mean Absolute Error: 51.16163391161018
Mean Squared Error: 9719.445188227117
</pre></td>
</tr>
<tr>
<td>Decision Tree</td>
<td><pre>
def decision_tree_regressor_with_tuning(X_train, X_test, y_train, y_test):
    dt = DecisionTreeRegressor()
    param_grid = {
        'max_depth': [5, 10, 20],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
    grid_search = GridSearchCV(dt, param_grid, cv=5, scoring='r2')
    grid_search.fit(X_train, y_train)

    best_dt = grid_search.best_estimator_
    y_pred = best_dt.predict(X_test)
</pre></td>
<td><pre>
Decision Tree Regressor
Best Params: {'max_depth': 5, 'min_samples_leaf': 4, '
Mean Absolute Error: 55.310368452997785
Mean Squared Error: 10950.050909112899
R2 Score: 0.7161572643132731
</pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
Linear Regression	R2: 0.7758	R2: 0.7758
SVM Regression	R2: 0.1528	R2: 0.1528
Random Forest Regression	R2: 0.7617	R2: 0.7617
Decision Tree Regressor	R2: 0.5318	R2: 0.5318

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Linear Regression	The Linear Regression model was chosen as the final optimized model because it exhibited the highest R-squared value (0.7758), indicating a strong fit to the data. Additionally, it had a lower MSE (8649.14) and Accuracy (77%) compared to other models, suggesting superior predictive accuracy.