SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

Model Optimization and Tuning Phase

Date	07 July 2024
Team ID	739863
Project Title	BlueBerry Yield Prediction
Maximum Marks	6 Marks

Hyperparameter Tuning Documentation :

Hyperparameter tuning involves adjusting the parameters that govern the training process of machine learning models to optimize their performance. It includes methods such as grid search, random search, and Bayesian optimization. Proper documentation helps in understanding the impact of different hyperparameters, streamlining the tuning process, and replicating results. Clear records of hyperparameter settings and their outcomes are essential for achieving the best model accuracy and efficiency.

Model	Tuned Hyperparameters	Optimal Values

Linear Regression	from sklearn.linear_model import Ridge
	ridge = Ridge()
	parameters = {'alpha': [0.1, 1, 10]} # Example values for regularization strength
	
	ridge_regressor = GridSearchCV(ridge, parameters, scoring='neg_mean_squared_error', cv=5)
	ridge_regressor.fit(x_train, y_train)
	
	best_alpha = ridge_regressor.best_params_['alpha']
	print("Best Alpha:", best_alpha)
	
	# Using the best model found by GridSearchCV
	best_ridge = ridge_regressor.best_estimator_
	best_ridge.fit(x_train, y_train)
	pred_ridge = best_ridge.predict(x_test)
		mae_ridge = mean_absolute_error(y_test, pred_ridge)
		mse_ridge = mean_squared_error(y_test, pred_ridge)
		rmse_ridge = np.sqrt(mse_ridge)
		rsq_ridge = r2_score(y_test, pred_ridge)
		
		print("MAE: %.3f" % mae_ridge)
		print("MSE: %.3f" % mse_ridge)
		print("RMSE: %.3f" % rmse_ridge)
		print("R-Square: %.3f" % rsq_ridge)
		print("Training Accuracy:", best_ridge.score(x_train, y_train))
		print("Testing Accuracy:", best_ridge.score(x_test, y_test))
		
		Best Alpha: 0.1
		MAE: 95.466
		MSE: 14043.502
		RMSE: 118.505
		R-Square: 0.991
		Training Accuracy: 0.991011446378135
		Testing Accuracy: 0.9913088598782471

RandomForest Regressor

```python
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

rf = RandomForestRegressor(random_state=42)

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)

grid_search.fit(x_train, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_

print(f"Best Parameters: {best_params}")
print(f"Best Cross-Validation Score: {best_score:.3f}")

# Train the model with the best parameters
best_rf = grid_search.best_estimator_
pred_rf_train_tu = best_rf.predict(x_train)
pred_rf_tu = best_rf.predict(x_test)
```

```python
mae_rf_train_tu = mean_absolute_error(y_train, pred_rf_train_tu)
mae_rf_tu = mean_absolute_error(y_test, pred_rf_tu)
mse_rf_tu = mean_squared_error(y_test, pred_rf_tu)
rmse_rf_tu = np.sqrt(mse_rf_tu)
rsq_rf_tu = r2_score(y_test, pred_rf_tu)

print("MAE_train: %.3f" % mae_rf_train_tu)
print("MAE: %.3f" % mae_rf_tu)
print("MSE: %.3f" % mse_rf_tu)
print("RMSE: %.3f" % rmse_rf_tu)
print("R-Square: %.3f" % rsq_rf_tu)
print("Training Accuracy: %.3f" % best_rf.score(x_train, y_train))
print("Testing Accuracy: %.3f" % best_rf.score(x_test, y_test))
```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best Parameters: {'bootstrap': True, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Best Cross-Validation Score: 0.986
MAE_train: 41.448
MAE: 110.332
MSE: 19188.170
RMSE: 138.521
R-Square: 0.988
Training Accuracy: 0.998
Testing Accuracy: 0.988
```

DecisionTree Regressor

```python
dt = DecisionTreeRegressor()

param_grid = {
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 5, 10],
    'max_features': ['auto', 'sqrt', 'log2', None]
}

grid_search = GridSearchCV(estimator=dt, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)

grid_search.fit(x_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)

best_dt = grid_search.best_estimator_
pred_dt_tu = best_dt.predict(x_test)
```

```python
mae_dt_tu = mean_absolute_error(y_test, pred_dt_tu)
mse_dt_tu = mean_squared_error(y_test, pred_dt_tu)
rmse_dt_tu = np.sqrt(mse_dt_tu)
rsq_dt_tu = r2_score(y_test, pred_dt_tu)

print("MAE:", mae_dt_tu)
print("MSE:", mse_dt_tu)
print("RMSE:", rmse_dt_tu)
print("R-Squared:", rsq_dt_tu)
print("Training Accuracy:", best_dt.score(x_train, y_train))
print("Testing Accuracy:", best_dt.score(x_test, y_test))
```

```
Best Parameters: {'max_depth': None, 'max_features': None, 'min_samples_leaf': 5, 'min_samples_split': 10}
Best CV Score: -40740.29928310072
MAE: 128.17739583664462
MSE: 30284.679955869266
RMSE: 174.02494061446845
R-Squared: 0.9812576374711801
Training Accuracy: 0.9931849259250838
Testing Accuracy: 0.9812576374711801
```

XGBoost Regressor

```python
xgb = XGBRegressor()
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

grid_search = GridSearchCV(estimator=xgb, param_grid=param_grid,
                           scoring='neg_mean_squared_error', cv=5, verbose=1)

grid_search.fit(x_train, y_train)

print("Best Parameters:", grid_search.best_params_)
print("Best CV Score:", grid_search.best_score_)

best_xgb = grid_search.best_estimator_

pred_xgb_tuned = best_xgb.predict(x_test)
```

```python
mae_xgb_tuned = mean_absolute_error(y_test, pred_xgb_tuned)
mse_xgb_tuned = mean_squared_error(y_test, pred_xgb_tuned)
rmse_xgb_tuned = np.sqrt(mse_xgb_tuned)
rsq_xgb_tuned = r2_score(y_test, pred_xgb_tuned)

print("\nTuned Model Metrics:")
print("MAE: %.3f" % mae_xgb_tuned)
print("MSE: %.3f" % mse_xgb_tuned)
print("RMSE: %.3f" % rmse_xgb_tuned)
print("R-Squared: %.3f" % rsq_xgb_tuned)
print("Training Accuracy:", best_xgb.score(x_train, y_train))
print("Testing Accuracy:", best_xgb.score(x_test, y_test))
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
Best Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 3, 'min_child_weight': 1, 'subsample': 0.6}
Best CV Score: -16626.085239377753

Tuned Model Metrics:
MAE: 94.131
MSE: 14517.358
RMSE: 120.488
R-Squared: 0.991
Training Accuracy: 0.9951537856788809
Testing Accuracy: 0.9910156029061967
```