27/03/2024

CSC354 – Assignmen2 – ML – Decision Trees

Fahad Ajmal

FA21-BSE-024

QUESTION 1:

*import* pandas *as* pd

*from* sklearn.model\_selection *import* train\_test\_split, RandomizedSearchCV, GridSearchCV

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.ensemble *import* RandomForestClassifier

*from* sklearn.metrics *import* accuracy\_score

df = pd.read\_csv("datasaurus.csv")

X = df[['x', 'y']]

y = df['dataset']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.2, *random\_state*=42)

baseline\_tree = DecisionTreeClassifier(*random\_state*=42)

baseline\_tree.fit(X\_train, y\_train)

baseline\_tree\_predictions = baseline\_tree.predict(X\_test)

baseline\_tree\_accuracy = accuracy\_score(y\_test, baseline\_tree\_predictions)

print("Baseline Decision Tree (J48) Accuracy:", baseline\_tree\_accuracy)

baseline\_rf = RandomForestClassifier(*random\_state*=42)

baseline\_rf.fit(X\_train, y\_train)

baseline\_rf\_predictions = baseline\_rf.predict(X\_test)

baseline\_rf\_accuracy = accuracy\_score(y\_test, baseline\_rf\_predictions)

print("Baseline Random Forest Accuracy:", baseline\_rf\_accuracy)

param\_dist = {"max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "criterion": ["gini", "entropy"]}

random\_search\_tree = RandomizedSearchCV(DecisionTreeClassifier(*random\_state*=42),

*param\_distributions*=param\_dist,

*n\_iter*=100,

*random\_state*=42)

random\_search\_tree.fit(X\_train, y\_train)

best\_tree = random\_search\_tree.best\_estimator\_

best\_tree\_predictions = best\_tree.predict(X\_test)

best\_tree\_accuracy = accuracy\_score(y\_test, best\_tree\_predictions)

print("Best Decision Tree (J48) Accuracy after Random Search:", best\_tree\_accuracy)

print("Best Decision Tree (J48) Parameters:", random\_search\_tree.best\_params\_)

param\_dist = {"n\_estimators": [10, 50, 100, 200],

              "max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "bootstrap": [True, False],

              "criterion": ["gini", "entropy"]}

random\_search\_rf = RandomizedSearchCV(RandomForestClassifier(*random\_state*=42),

*param\_distributions*=param\_dist,

*n\_iter*=100,

*random\_state*=42)

random\_search\_rf.fit(X\_train, y\_train)

best\_rf = random\_search\_rf.best\_estimator\_

best\_rf\_predictions = best\_rf.predict(X\_test)

best\_rf\_accuracy = accuracy\_score(y\_test, best\_rf\_predictions)

print("Best Random Forest Accuracy after Random Search:", best\_rf\_accuracy)

print("Best Random Forest Parameters:", random\_search\_rf.best\_params\_)

param\_grid = {"max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "criterion": ["gini", "entropy"]}

grid\_search\_tree = GridSearchCV(DecisionTreeClassifier(*random\_state*=42),

*param\_grid*=param\_grid)

grid\_search\_tree.fit(X\_train, y\_train)

best\_tree\_grid = grid\_search\_tree.best\_estimator\_

best\_tree\_grid\_predictions = best\_tree\_grid.predict(X\_test)

best\_tree\_grid\_accuracy = accuracy\_score(y\_test, best\_tree\_grid\_predictions)

print("Best Decision Tree (J48) Accuracy after Grid Search:", best\_tree\_grid\_accuracy)

print("Best Decision Tree (J48) Parameters:", grid\_search\_tree.best\_params\_)

param\_grid = {"n\_estimators": [10, 50, 100, 200],

              "max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "bootstrap": [True, False],

              "criterion": ["gini", "entropy"]}

grid\_search\_rf = GridSearchCV(RandomForestClassifier(*random\_state*=42),

*param\_grid*=param\_grid)

grid\_search\_rf.fit(X\_train, y\_train)

best\_rf\_grid = grid\_search\_rf.best\_estimator\_

best\_rf\_grid\_predictions = best\_rf\_grid.predict(X\_test)

best\_rf\_grid\_accuracy = accuracy\_score(y\_test, best\_rf\_grid\_predictions)

print("Best Random Forest Accuracy after Grid Search:", best\_rf\_grid\_accuracy)

print("Best Random Forest Parameters:", grid\_search\_rf.best\_params\_)

QUESTION 2:

*import* pandas *as* pd

*from* sklearn.model\_selection *import* train\_test\_split, RandomizedSearchCV, GridSearchCV

*from* sklearn.tree *import* DecisionTreeRegressor

*from* sklearn.metrics *import* mean\_squared\_error, mean\_absolute\_error

df = pd.read\_csv("cars-dataset.csv")

df = pd.get\_dummies(df, *columns*=['fuel', 'seller\_type', 'transmission', 'owner'])

X = df.drop(*columns*=['selling\_price'])

y = df['selling\_price']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, *test\_size*=0.2, *random\_state*=42)

baseline\_tree = DecisionTreeRegressor(*random\_state*=42)

baseline\_tree.fit(X\_train, y\_train)

baseline\_tree\_predictions = baseline\_tree.predict(X\_test)

baseline\_tree\_mse = mean\_squared\_error(y\_test, baseline\_tree\_predictions)

baseline\_tree\_mae = mean\_absolute\_error(y\_test, baseline\_tree\_predictions)

print("Baseline Decision Tree Regressor MSE:", baseline\_tree\_mse)

print("Baseline Decision Tree Regressor MAE:", baseline\_tree\_mae)

param\_dist = {"max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "max\_features": ["auto", "sqrt", "log2"]}

random\_search\_tree = RandomizedSearchCV(DecisionTreeRegressor(*random\_state*=42),

*param\_distributions*=param\_dist,

*n\_iter*=100,

*random\_state*=42,

*scoring*='neg\_mean\_squared\_error',

*cv*=5,

*verbose*=1)

random\_search\_tree.fit(X\_train, y\_train)

best\_tree = random\_search\_tree.best\_estimator\_

best\_tree\_predictions = best\_tree.predict(X\_test)

best\_tree\_mse = mean\_squared\_error(y\_test, best\_tree\_predictions)

best\_tree\_mae = mean\_absolute\_error(y\_test, best\_tree\_predictions)

print("Best Decision Tree Regressor MSE after Random Search:", best\_tree\_mse)

print("Best Decision Tree Regressor MAE after Random Search:", best\_tree\_mae)

print("Best Decision Tree Regressor Parameters:", random\_search\_tree.best\_params\_)

param\_grid = {"max\_depth": [3, None],

              "min\_samples\_split": [2, 5, 10],

              "min\_samples\_leaf": [1, 2, 4],

              "max\_features": ["auto", "sqrt", "log2"]}

grid\_search\_tree = GridSearchCV(DecisionTreeRegressor(*random\_state*=42),

*param\_grid*=param\_grid,

*scoring*='neg\_mean\_squared\_error',

*cv*=5,

*verbose*=1)

grid\_search\_tree.fit(X\_train, y\_train)

best\_tree\_grid = grid\_search\_tree.best\_estimator\_

best\_tree\_grid\_predictions = best\_tree\_grid.predict(X\_test)

best\_tree\_grid\_mse = mean\_squared\_error(y\_test, best\_tree\_grid\_predictions)

best\_tree\_grid\_mae = mean\_absolute\_error(y\_test, best\_tree\_grid\_predictions)

print("Best Decision Tree Regressor MSE after Grid Search:", best\_tree\_grid\_mse)

print("Best Decision Tree Regressor MAE after Grid Search:", best\_tree\_grid\_mae)

print("Best Decision Tree Regressor Parameters:", grid\_search\_tree.best\_params\_)

QUESTION 3:

The experience of attempting both Q1 and Q2 provided valuable insights into building and optimizing decision tree-based models for both classification and regression tasks. I gained a deeper understanding of the models.