Assignment 9

Josh Barber

barb4630@mylaurier.ca

215854630

Code:

import numpy as np

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import RandomizedSearchCV

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.ensemble import RandomForestRegressor

california\_housing\_bunch = fetch\_california\_housing()

california\_housing\_X, california\_housing\_y = california\_housing\_bunch.data, california\_housing\_bunch.target

x\_train, x\_test, y\_train, y\_test = train\_test\_split(california\_housing\_X, california\_housing\_y, test\_size=0.2)

# Deafult

rf = RandomForestRegressor(max\_features="sqrt", min\_samples\_split=5, min\_samples\_leaf=2)

rf.fit(x\_train, y\_train)

print("random forest regressor with default parameters")

california\_housing\_y\_pred = rf.predict(x\_test)

print("Mean squared error: %.2f" % mean\_squared\_error(y\_test, california\_housing\_y\_pred))

print("Coefficient of determination: %.2f" % r2\_score(y\_test, california\_housing\_y\_pred))

# Random Search Tuning

n\_estimators = [int(x) for x in np.linspace(start = 600, stop = 2000, num = 15)]

max\_depth = [int(x) for x in np.linspace(10, 80, num = 8)]

max\_depth.append(None)

bootstrap = [True, False]

random\_grid = {'n\_estimators': n\_estimators,

'max\_depth': max\_depth,

'bootstrap': bootstrap}

rf\_random = RandomizedSearchCV(estimator = rf, param\_distributions = random\_grid, n\_iter = 10, cv = 5, n\_jobs = -1)

rf\_random.fit(x\_train, y\_train)

print("random forest regressor with random search tuning")

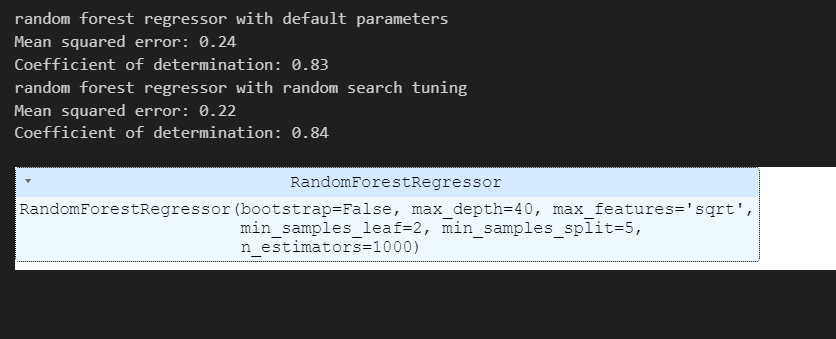
california\_housing\_y\_pred = rf\_random.best\_estimator\_.predict(x\_test)

print("Mean squared error: %.2f" % mean\_squared\_error(y\_test, california\_housing\_y\_pred))

print("Coefficient of determination: %.2f" % r2\_score(y\_test, california\_housing\_y\_pred))

rf\_random.best\_estimator\_

Result:



The random search was able to find a better model with tuned hyper-parameters as compared to the default model generated. Although there is not a significant increase in performance over the default model, nevertheless the potential for performance increase would vary dependent on the model and data.