Homework Assignment 2

Jarod Klion

December 19th, 2021

|  |  |
| --- | --- |
| OLS Regression | |
| Set | Average R2 |
| training | 0.9538 |
| test | 0.9544 |

Chart, line chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Random Forest Regression | | |
| Number of Trees | Average Training R2 | Average Testing R2 |
| 10 | 0.9111 | 0.4916 |
| 30 | 0.9295 | 0.5204 |
| 100 | 0.9359 | 0.5311 |
| 300 | 0.9376 | 0.5342 |

#!/usr/bin/env python

# coding: utf-8

# In[424]:

import numpy as np

import pandas as pd

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

import statsmodels.api as sm

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import r2\_score

import matplotlib.pyplot as plt

# In[55]:

#Read in the needed data for the assignment

df = pd.read\_csv("abalone.csv", header=None)

X = df.iloc[:, :-1] #Select first 7 columns as predictors

y = df.iloc[:, -1:] #Select last column as the response

df

# In[86]:

#Set up the designated split

X\_train, X\_test, y\_train, y\_test = zip(\*(train\_test\_split(X, y, test\_size=0.1, random\_state = i) for i in range(1,11)))

#zip(\*) unpacks the lists to each variable

# In[171]:

#Perform OLS regression on training and testing set analytically using linalg solver

betas\_train = [np.linalg.solve(X\_train[i].T @ X\_train[i] + 0.0001, X\_train[i].T @ y\_train[i]) for i in range(10)]

betas\_test = [np.linalg.solve(X\_test[i].T @ X\_test[i] + 0.0001, X\_test[i].T @ y\_test[i]) for i in range(10)]

#print(\*betas\_train, sep="\n"\*2)

#Show that these match the regression

reg\_train = [sm.OLS(y\_train[i], X\_train[i]).fit() for i in range(10)]

reg\_test = [sm.OLS(y\_test[i], X\_test[i]).fit() for i in range(10)]

#for i in range(10):

# print("beta value: " + str(betas\_train[i]), "\t", "regression value: " + str(reg\_train[i].params))

# In[514]:

R2\_train = np.mean([reg\_train[i].rsquared for i in range(10)])

R2\_test = np.mean([reg\_test[i].rsquared for i in range(10)])

print("Average training R^2: {}".format(R2\_train), "\nAverage testing R^2: {}".format(R2\_test))

# In[445]:

#Define a function to create regression trees

def run\_tree\_reg(X\_train, X\_test, y\_train, y\_test, depth):

tree\_reg = DecisionTreeRegressor(random\_state = 42, max\_depth = depth).fit(X\_train, y\_train)

#this accuracy measure is the same as R^2

acc\_train = tree\_reg.score(X\_train, y\_train)

acc\_test = tree\_reg.score(X\_test, y\_test)

#print("Tree depth: ", depth)

#print("Training set accuracy: ", round(acc\_train\*100,4), "%")

#print("Testing set accuracy: ", round(acc\_test\*100,4), "%")

return acc\_train, acc\_test

# In[477]:

#Create 7 regression trees for each depth asked

tree\_acc\_train = np.empty(shape= (7,10))

tree\_acc\_test = np.empty(shape= (7,10))

depth = range(1,8)

for i in depth:

for k in range(10):

tree\_acc\_train[i-1][k], tree\_acc\_test[i-1][k] = run\_tree\_reg(X\_train[k], X\_test[k], y\_train[k], y\_test[k], i)

#Calculate the mean for each tree depth

tree\_R2\_train\_mean = [np.mean(tree\_acc\_train[i]) for i in range(7)]

tree\_R2\_test\_mean = [np.mean(tree\_acc\_test[i]) for i in range(7)]

# In[447]:

#Plot train and test R^2 vs the tree depth on the same graph

fig1 = plt.figure(1)

plt.plot(depth, tree\_R2\_train\_mean, "-o", label = "Training Set")

plt.plot(depth, tree\_R2\_test\_mean, "-o", label = "Testing Set")

plt.xlabel("Tree Depth")

plt.ylabel("Average $R^2$")

plt.title("Average $R^2$ vs Regression Tree Depth")

plt.legend()

plt.show()

#fig1.savefig("RegressionTree.jpg")

# In[448]:

#Define a function to create random forest regressions

def run\_rf\_reg(X\_train, X\_test, y\_train, y\_test, trees):

rf\_reg = RandomForestRegressor(random\_state = 42, n\_estimators = trees).fit(X\_train, y\_train)

#this accuracy measure is the same as R^2

acc\_train = rf\_reg.score(X\_train, y\_train)

acc\_test = rf\_reg.score(X\_test, y\_test)

print("# of trees: ", trees)

print("Training set accuracy: ", round(acc\_train\*100,4), "%")

print("Testing set accuracy: ", round(acc\_test\*100,4), "%")

return acc\_train, acc\_test

# In[482]:

#Create a random forest regression for each different tree value (10, 30, 100, 300)

rf\_acc\_train = np.empty(shape= (4,10))

rf\_acc\_test = np.empty(shape= (4,10))

trees = [10, 30, 100, 300]

for i, tree in enumerate(trees):

for k in range(10):

rf\_acc\_train[i][k], rf\_acc\_test[i][k] = run\_rf\_reg(X\_train[k], X\_test[k], y\_train[k].values.ravel(), y\_test[k].values.ravel(), tree)

#Calculate the mean for each different tree count

rf\_R2\_train\_mean = [np.mean(rf\_acc\_train[i]) for i in range(4)]

rf\_R2\_test\_mean = [np.mean(rf\_acc\_test[i]) for i in range(4)]

# In[509]:

print(\*[f"\nNumber of trees: {tree}\n" + f"Average training R^2: {rf\_R2\_train\_mean[i]}\n" + f"Average test R^2: {rf\_R2\_test\_mean[i]}" for i, tree in enumerate(trees)])