HW2

Christopher Bloome

7/8/2021

# 7.2 Python

We take the same dataset in 7.2 and try to reproduce the questions using Python. To do this we used the reticulate package in R that enables the use of using python code in markdown chunks

Overall the results were lackluster due to the differences between how R and the Sklearn packages handle variables. The sklearn package expects target variables not to be continuous. The target variables are encoded to integer values and thus the package treats each target variable as a unique class. Since there are 200 observations in the training dataset, the encoding process created 200 different classes for the target variables. Additionally due to this, hyperparameter tuning was not possible. This overall resulted in very poor training performance for both SVM and KNN models, the MARS model faired the best

library(reticulate)

## Warning: package 'reticulate' was built under R version 4.0.5

use\_python("C:/Users/dhair/anaconda3") #should be updated per your lcoal env  
  
#Once installed can be removed  
py\_install("pandas")  
py\_install("scikit-learn")  
py\_install("sklearn-contrib-py-earth")

Transfer training and test dataframes from the R enviornment to the Python enviornemnt

py$train\_set\_x <- r\_to\_py(trainingData$x)

## Warning: Python 'C:/Users/dhair/anaconda3/python.exe' was requested but 'C:/  
## Users/dhair/AppData/Local/r-miniconda/envs/r-reticulate/python.exe' was loaded  
## instead (see reticulate::py\_config() for more information)

py$train\_set\_y <- r\_to\_py(trainingData$y)  
py$test\_set <- r\_to\_py(testData$x)

Import python libraries

import pandas as pd  
import numpy as np  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
import pyearth  
from sklearn.model\_selection import RandomizedSearchCV  
from sklearn.model\_selection import GridSearchCV  
from sklearn import preprocessing  
from sklearn import utils  
from sklearn.metrics import mean\_squared\_error  
from sklearn.metrics import r2\_score  
from sklearn.metrics import classification\_report  
from sklearn import preprocessing

Scale training data and encode the target variable

scaler = preprocessing.StandardScaler().fit(train\_set\_x)  
train\_set\_x\_scaled = scaler.transform(train\_set\_x)  
  
encoder = preprocessing.LabelEncoder()  
train\_set\_y\_encoded = encoder.fit\_transform(train\_set\_y)

KNN model results in scores that do not make sense, due to the fact that there are no similar target classes becuase of the encoding

knn = KNeighborsClassifier(n\_neighbors=19)  
knn\_results = knn.fit(train\_set\_x\_scaled, train\_set\_y\_encoded)  
knn\_pred = knn\_results.predict(train\_set\_x\_scaled)  
print("KNN Score: " + str(knn.score(train\_set\_x\_scaled, train\_set\_y\_encoded)))

## KNN Score: 0.015

print("KNN RMSE: " + str(mean\_squared\_error(train\_set\_y\_encoded, knn\_pred)))

## KNN RMSE: 8153.47

print("KNN R-squared: " + str(r2\_score(train\_set\_y\_encoded, knn\_pred)))

## KNN R-squared: -1.4461021525538138

THe SVM model results in the same issues as the KNN model

svm\_m = SVC(kernel='rbf', gamma="auto")  
svm\_results = svm\_m.fit(train\_set\_x\_scaled, train\_set\_y\_encoded)  
  
print("SVM Score: " + str(svm\_results.score(train\_set\_x\_scaled, train\_set\_y\_encoded)))

## SVM Score: 1.0

svm\_pred = svm\_results.predict(train\_set\_x\_scaled)  
print("SVM RMSE: " + str(mean\_squared\_error(train\_set\_y\_encoded, svm\_pred)))

## SVM RMSE: 0.0

print("SVM R-squared: " + str(r2\_score(train\_set\_y\_encoded, svm\_pred)))

## SVM R-squared: 1.0

The MARS model fairs the best of the 3 given the dataset

mars = pyearth.Earth()  
mars\_results = mars.fit(train\_set\_x\_scaled, train\_set\_y)

## C:\Users\dhair\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pyearth\earth.py:813: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.  
## To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.  
## pruning\_passer.run()  
## C:\Users\dhair\AppData\Local\R-MINI~1\envs\R-RETI~1\lib\site-packages\pyearth\earth.py:1066: FutureWarning: `rcond` parameter will change to the default of machine precision times ``max(M, N)`` where M and N are the input matrix dimensions.  
## To use the future default and silence this warning we advise to pass `rcond=None`, to keep using the old, explicitly pass `rcond=-1`.  
## coef, resid = np.linalg.lstsq(B, weighted\_y[:, i])[0:2]

print(mars\_results.summary())

## Earth Model  
## --------------------------------------  
## Basis Function Pruned Coefficient   
## --------------------------------------  
## (Intercept) No 12.4128   
## h(x0-0.673905) Yes None   
## h(0.673905-x0) No -2.01315   
## x3 No 2.76798   
## h(x1-0.342446) Yes None   
## h(0.342446-x1) No -2.88561   
## x4 Yes None   
## h(x2+0.217285) No 3.5214   
## h(-0.217285-x2) No -5.43924   
## h(x8-1.39745) Yes None   
## h(1.39745-x8) Yes None   
## h(x0+0.462884) Yes None   
## h(-0.462884-x0) No -1.86342   
## h(x2+0.806189) Yes None   
## h(-0.806189-x2) No 5.89439   
## h(x2-0.421124) Yes None   
## h(0.421124-x2) No 4.63714   
## x5 Yes None   
## h(x5+0.452668) Yes None   
## h(-0.452668-x5) No -1.1895   
## h(x5-1.61069) Yes None   
## h(1.61069-x5) No 0.700106   
## h(x5-1.43655) Yes None   
## h(1.43655-x5) Yes None   
## h(x4-0.537899) Yes None   
## h(0.537899-x4) Yes None   
## h(x4+0.720508) No 1.4635   
## h(-0.720508-x4) No -1.3749   
## --------------------------------------  
## MSE: 1.9458, GCV: 2.7251, RSQ: 0.9202, GRSQ: 0.8894