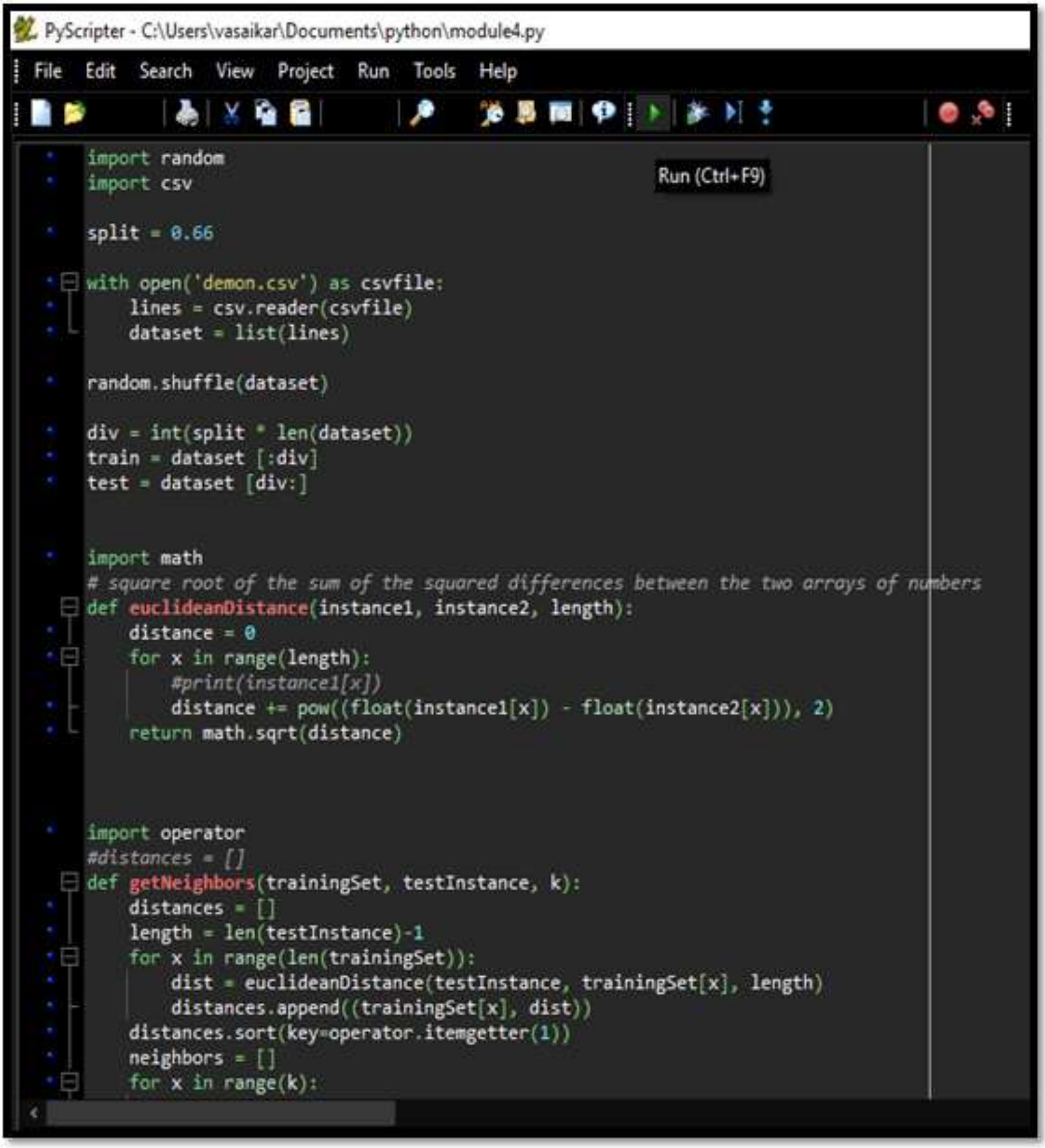
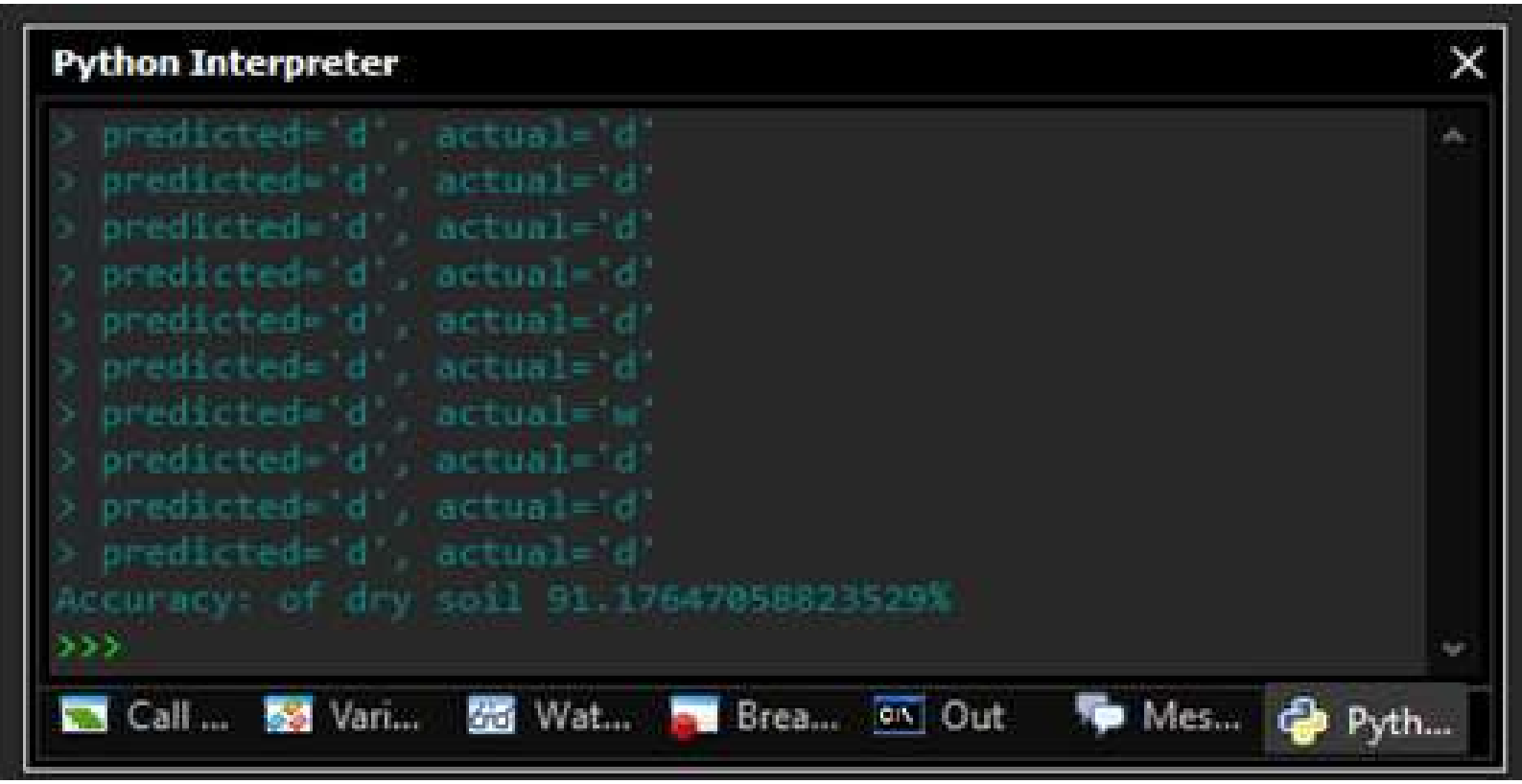
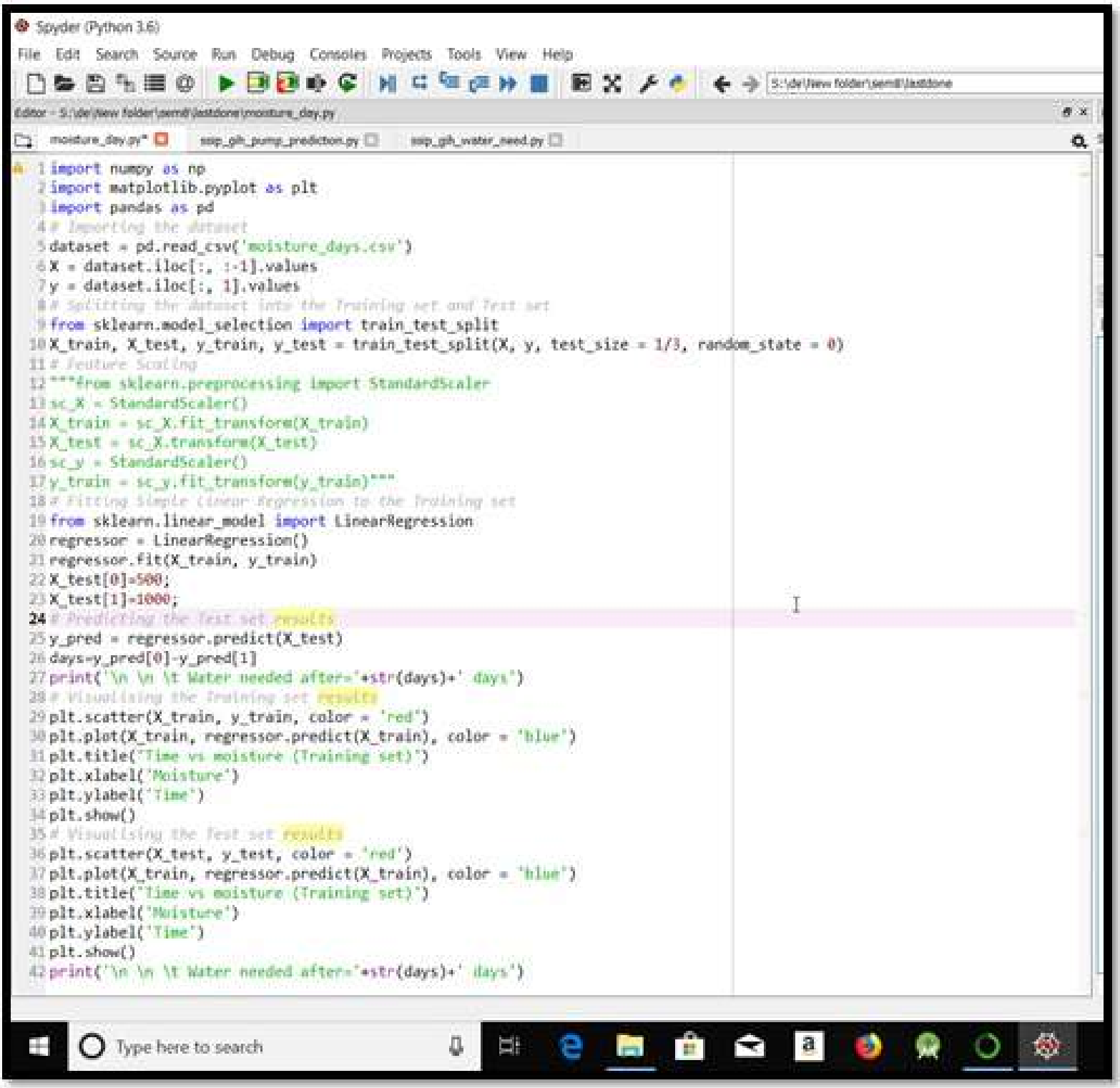
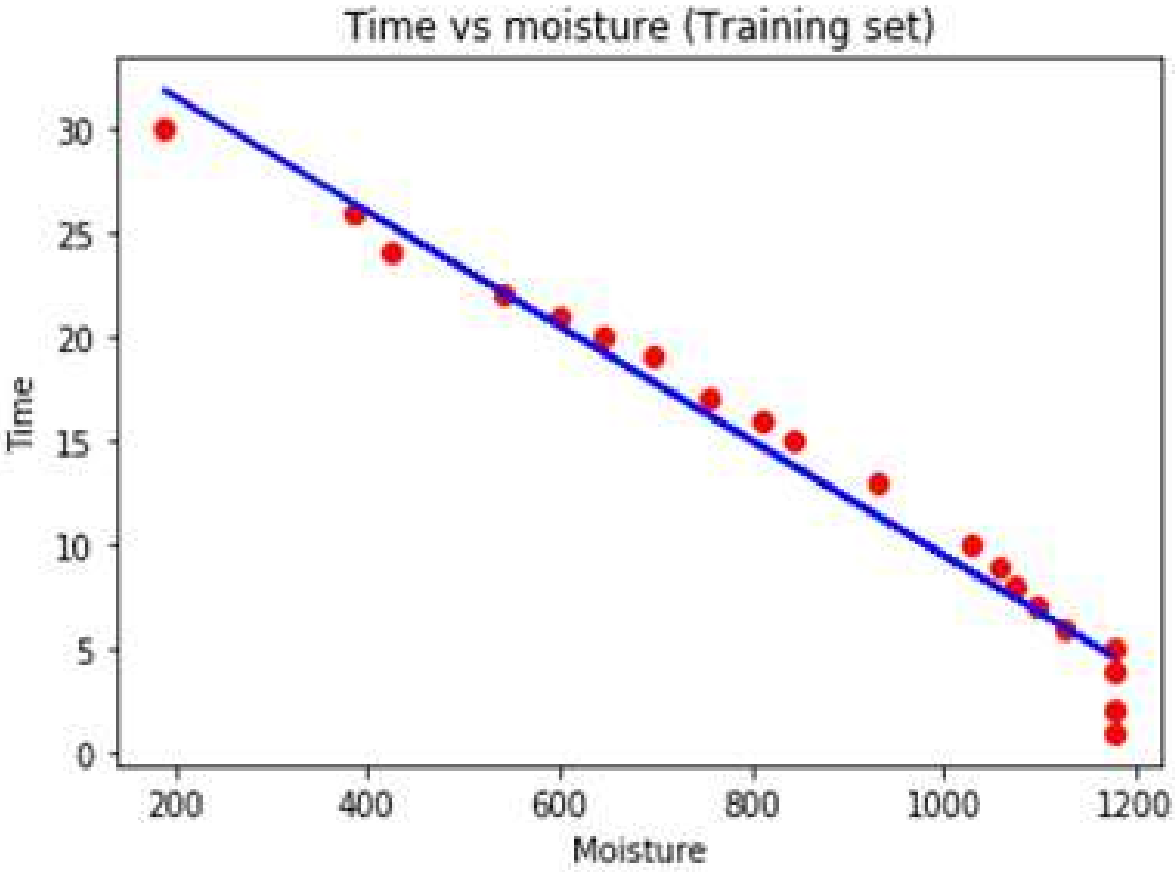
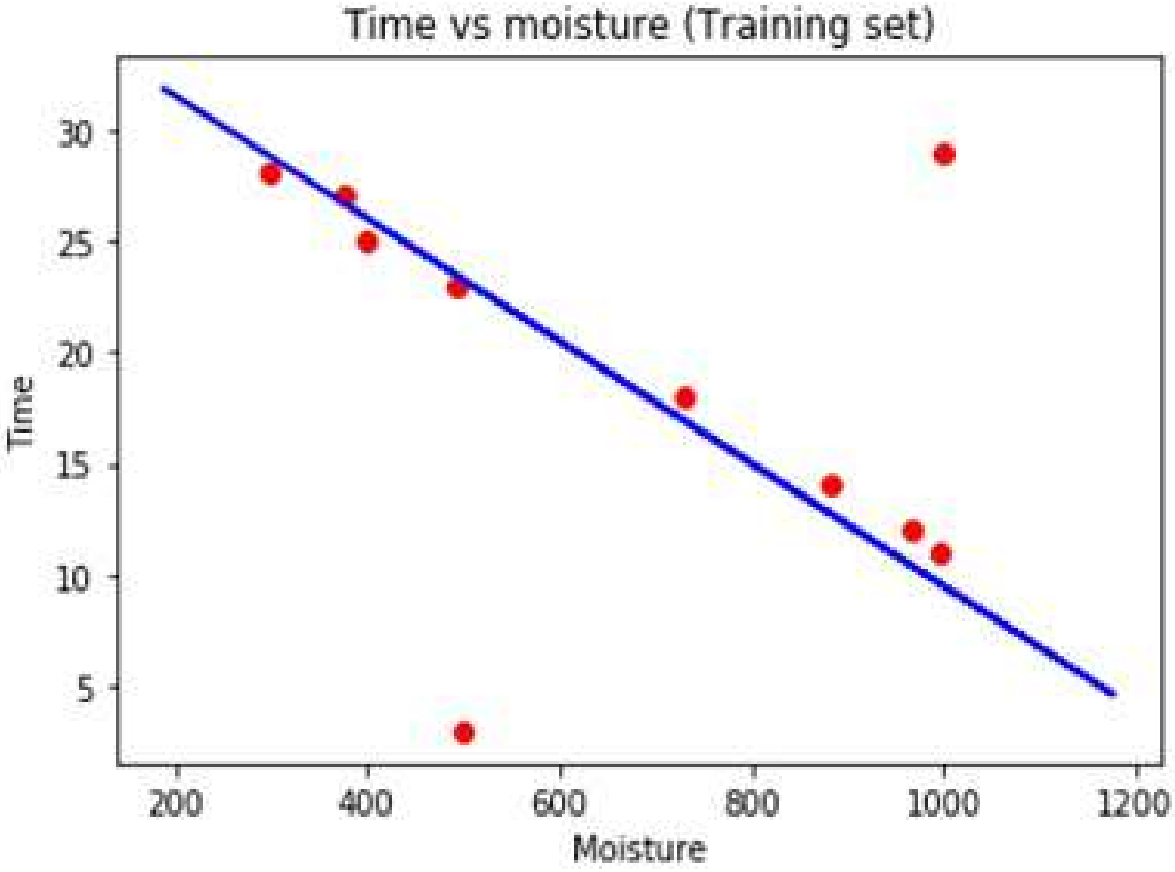

PyScripter - C:\Users\vasaikar\Documents\python\module4.py
File Edit Search View Project Run Tools Help
Run (Ctrl+F9)
import random
import csv
split = 0.66
with open('demon.csv') as csvfile:
lines = csv.reader(csvfile)
dataset = list(lines)
random.shuffle(dataset)
div = int(split * len(dataset))
train = dataset [:div]
test = dataset [div:]
import math
# square root of the sum of the squared differences between the two arrays of numbers
def euclideanDistance(instance1, instance2, length):
distance = 0
for x in range(length):
#print(instance1[x])
distance += pow((float(instance1[x]) - float(instance2[x])), 2)
return math.sqrt(distance)
import operator
#distances = []
def getNeighbors(trainingSet, testInstance, k):
distances = []
length = len(testInstance)-1
for x in range(len(trainingSet)):
dist = euclideanDistance(testInstance, trainingSet[x], length)
distances.append((trainingSet[x], dist))
distances.sort(key=operator.itemgetter(1))
neighbors = []
for x in range(k):

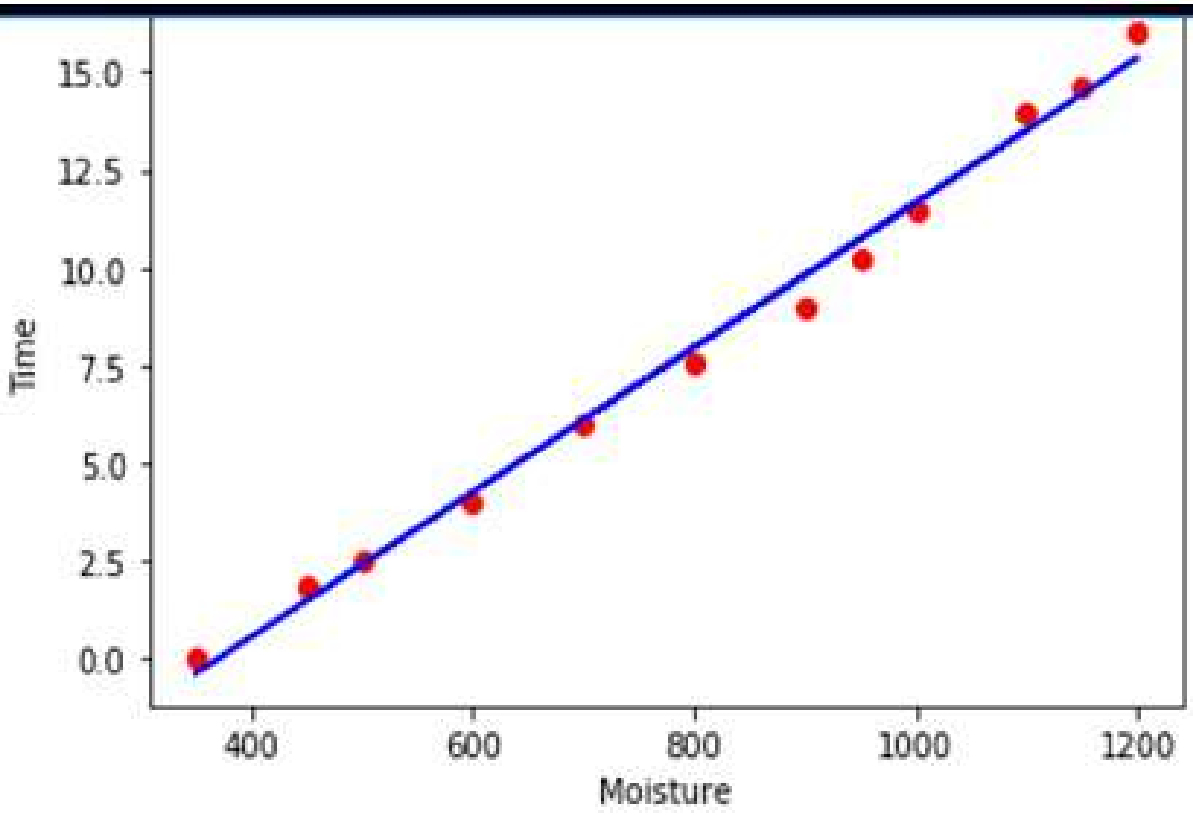

Python Interpreter
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='w'
> predicted='d', actual='d'
> predicted='d', actual='d'
> predicted='d', actual='d'
Accuracy: of dry soil 91.17647058823529%
>>>
Call ...
Vari...
Wat...
Brea...
Out
Mes...
Pyth...

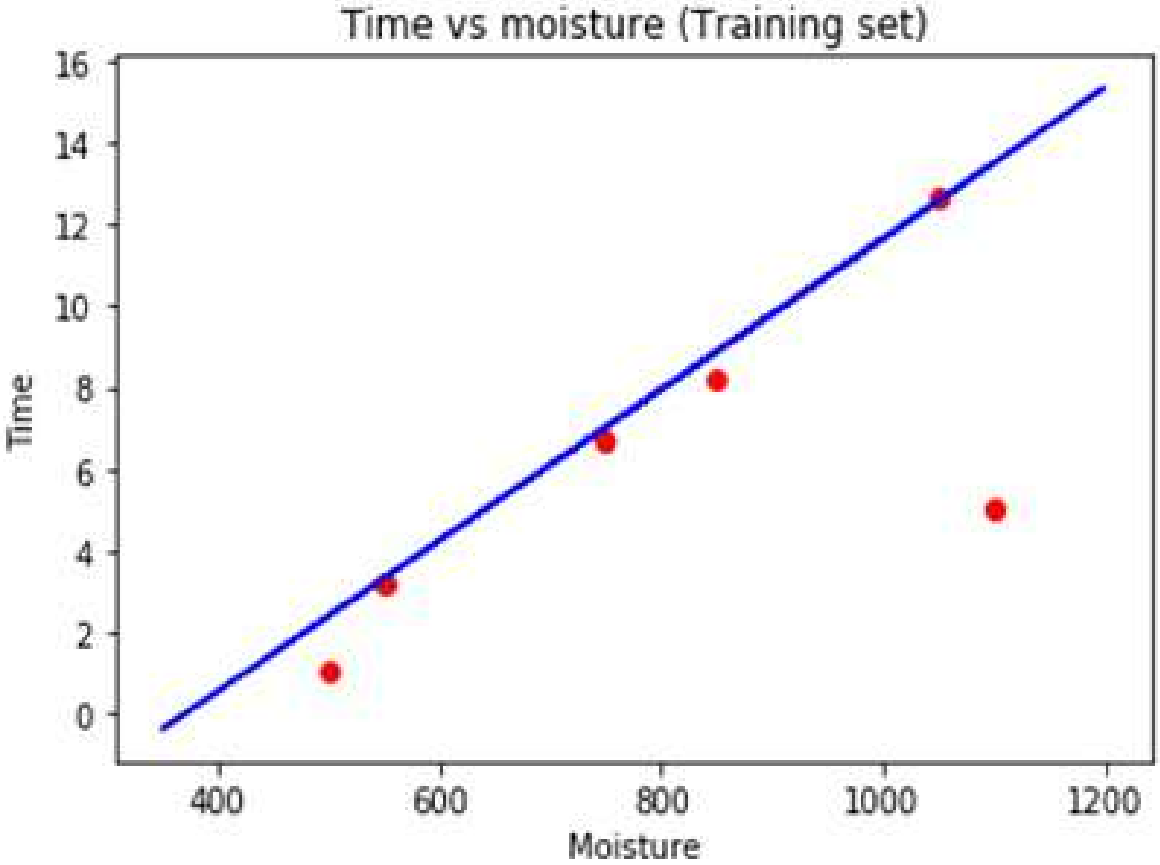

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
moisture_day.py*
soip_glh_pump_prediction.py
soip_glh_water_need.py
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('moisture_days.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
Feature Scaling
"""from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
sc_y = StandardScaler()
y_train = sc_y.fit_transform(y_train)"""
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
X_test[0]=500;
X_test[1]=1000;
# Predicting the Test set results
y_pred = regressor.predict(X_test)
days=y_pred[0]-y_pred[1]
print('\n \n \t Water needed after='+str(days)+' days')
# Visualising the Training set results
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Time vs moisture (Training set)')
plt.xlabel('Moisture')
plt.ylabel('Time')
plt.show()
# Visualising the Test set results
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Time vs moisture (Training set)')
plt.xlabel('Moisture')
plt.ylabel('Time')
plt.show()
print('\n \n \t Water needed after='+str(days)+' days')
Type here to search





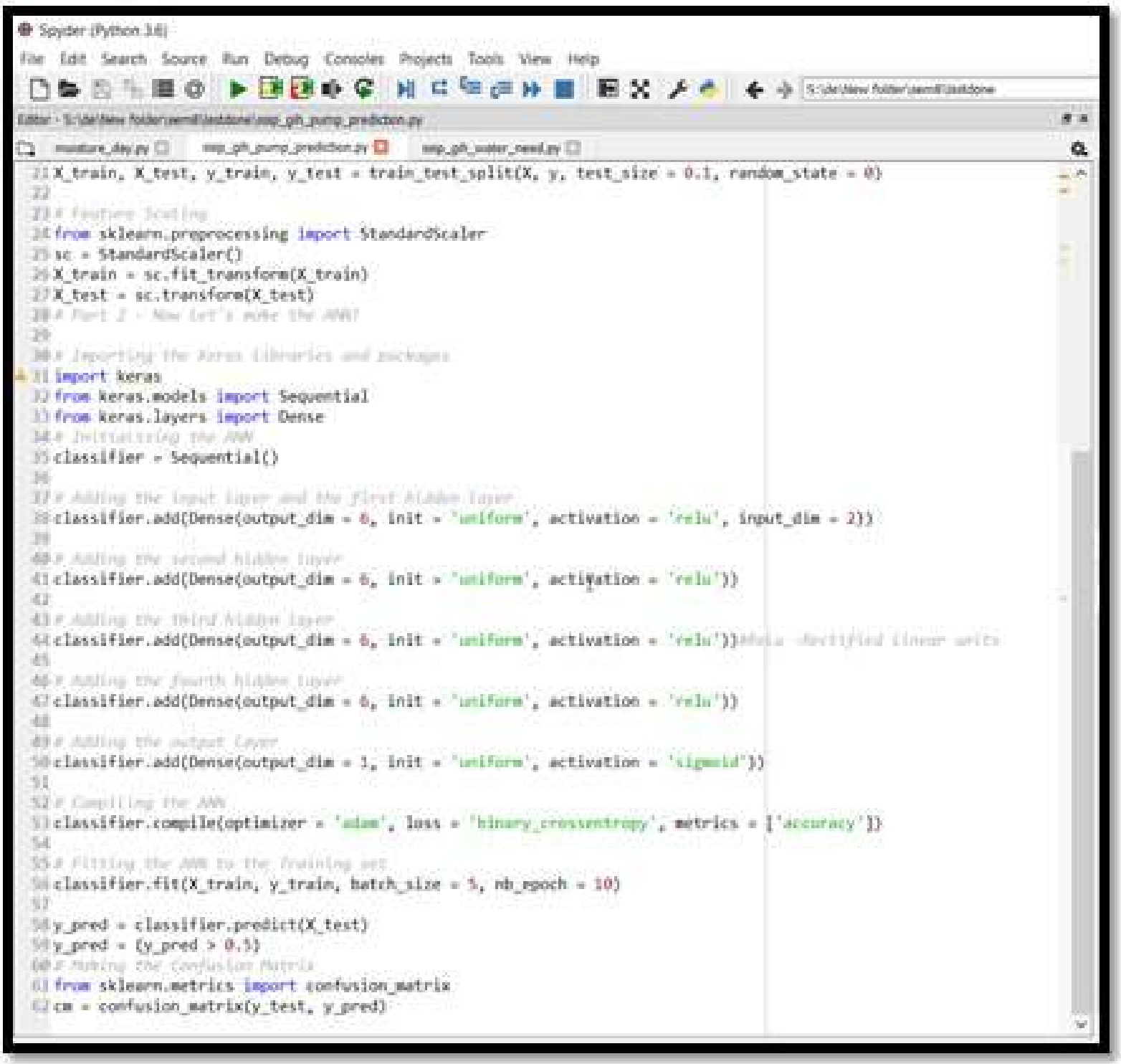
Water needed after=13.769618138810l2 days

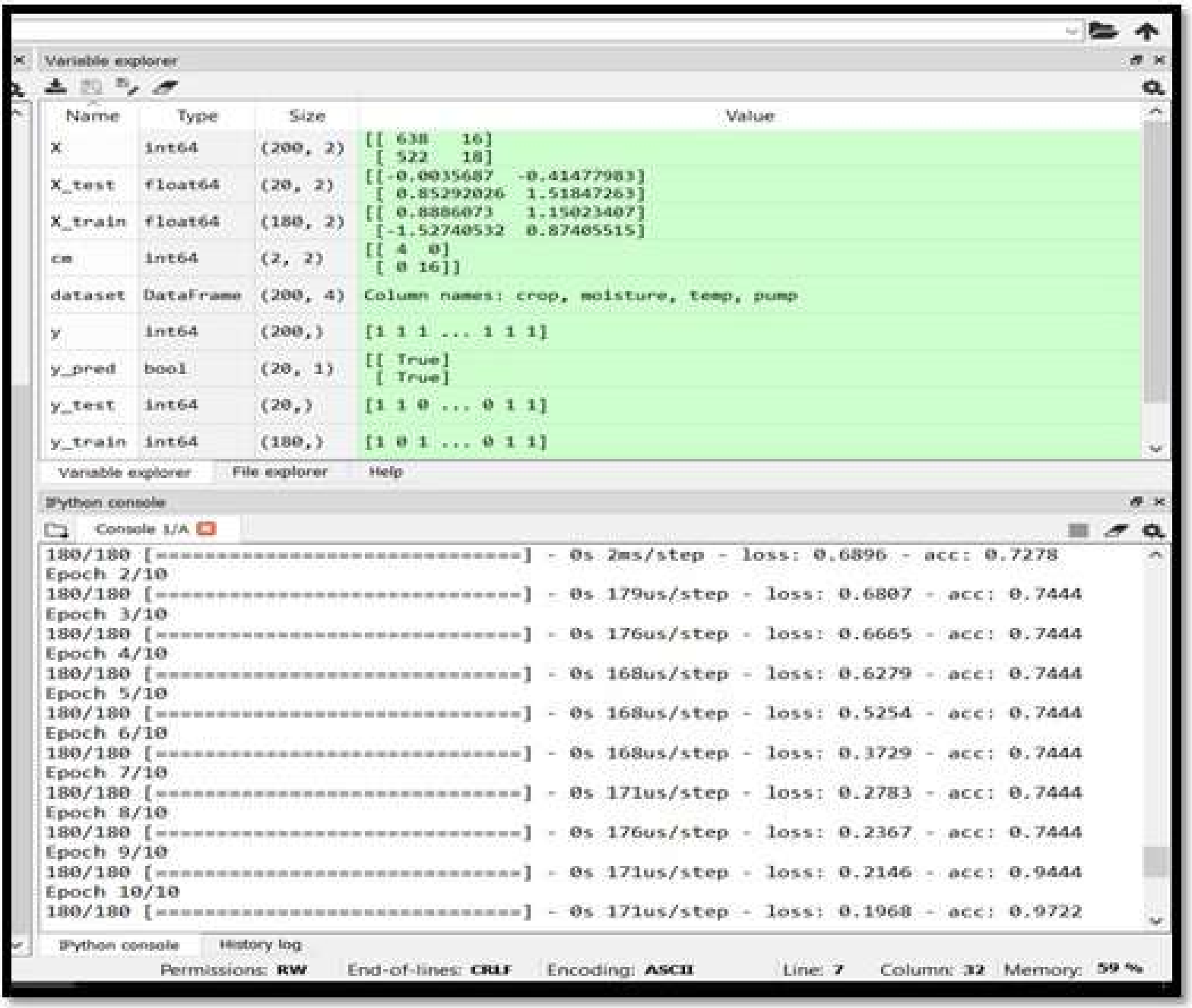




Water needed=2335.2153987167735 liters

Duration of Irrigation=11.120073327222273 Min


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
moisture_day.py
smp_gh_pump_prediction.py
smp_gh_water_need.py
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 0)
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# Part 2 - Now let's make the ANN!
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense
# Initialising the ANN
classifier = Sequential()
# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 2))
# Adding the second hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
# Adding the third hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))#Rela- Rectified linear units
# Adding the fourth hidden layer
classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
# Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 5, nb_epoch = 10)
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)


Variable explorer
Name Type Size Value
X int64 (200, 2) [[638 16] [522 18]]
X_test float64 (20, 2) [[-0.0035687 -0.41477983] [0.85292026 1.51847263]]
X_train float64 (180, 2) [[0.8886073 1.150234071 [-1.52740532 0.87405515]]
cm int64 (2, 2) [[4 0] [0 16]]
dataset DataFrame (200, 4) Column names: crop, moisture, temp, pump
y int64 (200,) [1 1 1 ... 1 1 1]
y_pred bool (20, 1) [[True] [True]]
y_test int64 (20,) [1 1 0 ... 0 1 1]
y_train int64 (180,) [1 0 1 ... 0 1 1]
Variable explorer File explorer Help
IPython console
Console 1/A
180/180 [==========] - 0s 2ms/step - loss: 0.6896 - acc: 0.7278
Epoch 2/10
180/180 [==========] - 0s 179us/step - loss: 0.6807 - acc: 0.7444
Epoch 3/10
180/180 [==========] - 0s 176us/step - loss: 0.6665 - acc: 0.7444
Epoch 4/10
180/180 [==========] - 0s 168us/step - loss: 0.6279 - acc: 0.7444
Epoch 5/10
180/180 [==========] - 0s 168us/step - loss: 0.5254 - acc: 0.7444
Epoch 6/10
180/180 [==========] - 0s 168us/step - loss: 0.3729 - acc: 0.7444
Epoch 7/10
180/180 [==========] - 0s 171us/step - loss: 0.2783 - acc: 0.7444
Epoch 8/10
180/180 [==========] - 0s 176us/step - loss: 0.2367 - acc: 0.7444
Epoch 9/10
180/180 [==========] - 0s 171us/step - loss: 0.2146 - acc: 0.9444
Epoch 10/10
180/180 [==========] - 0s 171us/step - loss: 0.1968 - acc: 0.9722
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 7 Column: 32 Memory: 59 %