# -\*- coding: utf-8 -\*-

"""

Created on Sun Jan 19 17:39:33 2020

@author: Dell 5530

"""

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('annual\_temp.csv')

Years = dataset.iloc[::2, 1::2].values # year

x = dataset.iloc[::2, 2].values

# Splitting the dataset into the Training set and Test set

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

Years\_train, Years\_test, x\_train, x\_test = train\_test\_split(Years, x, test\_size = 1/3, random\_state = 0)

# Fitting Simple Linear Regression to the Training set

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(Years\_train, x\_train)

# Predicting the Test set results

Y\_pred = regressor.predict(Years\_test)

# Fitting Polynomial Regression to the dataset

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 5)

Years\_poly = poly\_reg.fit\_transform(Years)

poly\_reg.fit(Years\_poly, x)

lin\_reg\_2 = LinearRegression()

lin\_reg\_2.fit(Years\_poly, x)

# Visualising the Training set results

plt.scatter(Years\_train, x\_train, color = 'yellow')

plt.plot(Years\_train, regressor.predict(Years\_train), color = 'red')

plt.title('Annual Temp Years (Training set)')

plt.xlabel('Years ')

plt.ylabel('MEAN TEMP')

plt.show()

# Visualising the Test set results

plt.scatter(Years\_test, x\_test, color = 'brown')

plt.plot(Years\_train, regressor.predict(Years\_train), color = 'blue')

plt.title('Annual Temp x (Test set)')

plt.xlabel('Years ')

plt.ylabel('Mean Temp')

plt.show()

#WE NEED TO HERE SOMETHING BWFORE THIS

Years\_grid = np.arange(min(Years), max(Years), 0.1)

Years\_grid = Years\_grid.reshape((len(Years\_grid), 1))

plt.scatter(Years\_train, x\_train, color = 'purple')

plt.plot(Years\_grid, lin\_reg\_2.predict(poly\_reg.fit\_transform(Years\_grid)), color = 'black')

plt.title('x Annual Temperature')

plt.xlabel('Years')

plt.ylabel('x Annual Temperature')

plt.show()

print(regressor.predict([[2016]])) # double bracket matrix form MATRIX INPUT

print(regressor.predict([[2017]]))

###################################################################

A = dataset.iloc[1::2, 1::2].values # year

y = dataset.iloc[1::2, 2].values # mean temp

# Splitting the dataset into the Training set and Test set

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

Years\_train, Years\_test, y\_train, y\_test = train\_test\_split(Years, y, test\_size = 1/3, random\_state = 0)

# Fitting Simple Linear Regression to the Training set

from sklearn.linear\_model import LinearRegression

regressor1 = LinearRegression()

regressor1.fit(Years\_train, y\_train)

# Predicting the Test set results

Y\_pred2 = regressor1.predict(Years\_test)

# Fitting Polynomial Regression to the dataset

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 5)

Years\_poly = poly\_reg.fit\_transform(Years)

poly\_reg.fit(Years\_poly, y)

lin\_reg\_3 = LinearRegression()

lin\_reg\_3.fit(Years\_poly, y)

# Visualising the Training set results

plt.scatter(Years\_train, y\_train, color = 'black')

plt.plot(Years\_train, regressor1.predict(Years\_train), color = 'pink')

plt.title('Annual Temp Years (Training set)')

plt.xlabel('Years ')

plt.ylabel('Mean Temperature')

plt.show()

Years\_grid = np.arange(min(Years), max(Years), 0.1)

Years\_grid = Years\_grid.reshape((len(Years\_grid), 1))

plt.scatter(Years\_train, y\_train, color = 'yellow')

plt.plot(Years\_grid, lin\_reg\_3.predict(poly\_reg.fit\_transform(Years\_grid)), color = 'black')

plt.title('y Annual Temperature')

plt.xlabel('Years')

plt.ylabel('y Annual Temperature')

plt.show()

print('temp of y in 2017')

print(regressor1.predict([[2017]]))

print('temp of x in 2017')

print(regressor.predict([[2017]]))

print('temp of y in 2016')

print(regressor1.predict([[2016]]))

print('temp of x in 2016')

print(regressor.predict([[2016]]))

#accracy of x

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(x\_test, Y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(x\_test, Y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(x\_test, Y\_pred)))

#Accuracy of y

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, Y\_pred2))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, Y\_pred2))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, Y\_pred2)))