**Practical – 2: Application of Nonlinear Regression in Data Analytics**

**1) Polynomial Regression –**

**Description:** In this Polynomial Regression, we used previous data to fit polynomial.

**Code:**

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

**# data import and split data**

df = pd.read\_csv("FuelConsumptionCo2.csv")

cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION\_COMB','CO2EMISSIONS']]

msk = np.random.rand(len(df)) < 0.8

train = cdf[msk]

test = cdf[~msk]

**# fit two degree polynomial**

from sklearn.preprocessing import PolynomialFeatures

from sklearn import linear\_model

train\_x = np.asanyarray(train[['ENGINESIZE']])

train\_y = np.asanyarray(train[['CO2EMISSIONS']])

test\_x = np.asanyarray(test[['ENGINESIZE']])

test\_y = np.asanyarray(test[['CO2EMISSIONS']])

poly = PolynomialFeatures(degree=3)

train\_x\_poly = poly.fit\_transform(train\_x)

clf = linear\_model.LinearRegression()

train\_y\_ = clf.fit(train\_x\_poly, train\_y)

**# The coefficients**

print ('Coefficients: ', clf.coef\_)

print ('Intercept: ',clf.intercept\_)

**# spot polynomial**

plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS, color=”yellow”)

XX = np.arange(0.0, 10.0, 0.1)

yy = clf.intercept\_[0]+ clf.coef\_[0][1]\*XX + clf.coef\_[0][2]\*np.power(XX, 2) + clf.coef\_[0][3]\*np.power(XX, 3)

plt.plot(XX, yy, '-r' )

plt.xlabel("Engine size")

plt.ylabel("Emission")

**# Evaluation**

from sklearn.metrics import r2\_score

test\_x\_poly = poly.fit\_transform(test\_x)

test\_y\_ = clf.predict(test\_x\_poly)

print("Mean absolute error: %.2f" % np.mean(np.absolute(test\_y\_ - test\_y)))

print("Residual sum of squares (MSE): %.2f" % np.mean((test\_y\_ - test\_y) \*\* 2))

print("R2-score: %.2f" % r2\_score(test\_y\_ , test\_y) )

**Output:**

Coefficients: [[ 0. 27.01570578 4.95461217 -0.53693609]]

Intercept: [132.16625862]

Mean absolute error: 22.78

Residual sum of squares (MSE): 921.19

R2-score: 0.68

