RUMAISA MARYAM

MACHINE LEARNING

ASSIGHNMENT 3

13/04/2020

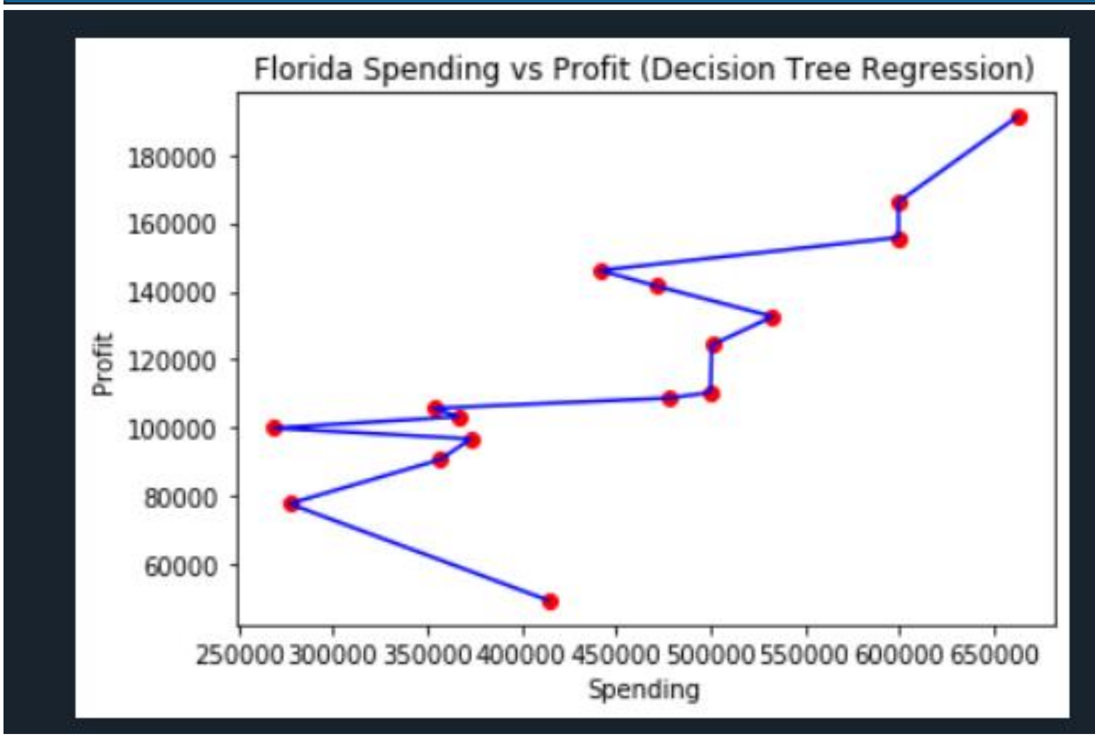
Q1: Take 50 startups of any two countries and find out which country is going to provide best profit in future.

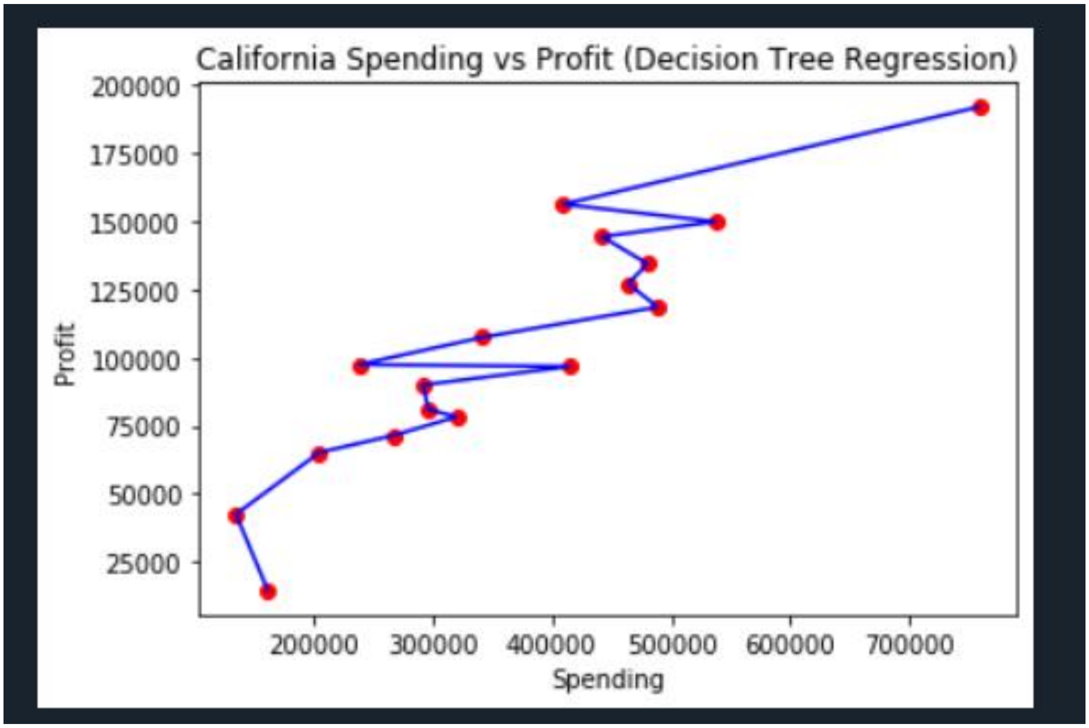
Decision Tree

Ans: *For this dataset I have used decision tree regression to solve the problem since it is a classification problem in which we are asked to predict and compare the profit of two cities.*

```python
#MACHINE LEARNING ASSIGNMENT3
#RUMAISA MARYAM

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('50_startups.csv')
dataset['Sum']=dataset[['R&D Spend', 'Administration', 'Marketing Spend']].sum(axis=1)

Xc = dataset.loc[(dataset.State=='California'),['Sum']]
Yc = dataset.loc[(dataset.State=='California'),['Profit']]


# Splitting the dataset into the Training set and Test set
"""from sklearn.model_selection import train_test_split
Xc_train, Xc_test, Yc_train, Yc_test = train_test_split(Xc, Yc, test_size = 0.3, random_state = 0)"""
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(Xc, Yc)

# Visualising the Decision Tree Regression results (higher resolution)
plt.scatter(Xc, Yc, color = 'red')
plt.plot(Xc, regressor.predict(Xc), color = 'blue')
plt.title('California Spending vs Profit (Decision Tree Regression)')
plt.xlabel('Spending')
plt.ylabel('Profit')
plt.show()

# Predicting a new result with Decision Tree
Z = regressor.predict([[9000000]])
print ("California =" ,Z)

Xf = dataset.loc[(dataset.State=='Florida'),['Sum']]
Yf = dataset.loc[(dataset.State=='Florida'),['Profit']]
```

```python

# Visualising the Decision Tree Regression results (higher resolution)
plt.scatter(Xc, Yc, color = 'red')
plt.plot(Xc, regressor.predict(Xc), color = 'blue')
plt.title('California Spending vs Profit (Decision Tree Regression)')
plt.xlabel('Spending')
plt.ylabel('Profit')
plt.show()

# Predicting a new result with Decision Tree
Z = regressor.predict([[9000000]])
print ("California =" ,Z)

Xf = dataset.loc[(dataset.State=='Florida'),['Sum']]
Yf = dataset.loc[(dataset.State=='Florida'),['Profit']]

# Splitting the dataset into the Training set and Test set
"""from sklearn.model_selection import train_test_split
Xf_train, Xf_test, Yf_train, Yf_test = train_test_split(Xf, Yf, test_size = 0.3, random_state = 0)"""
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(Xf, Yf)

# Visualising the Decision Tree Regression results (higher resolution)
plt.scatter(Xf, Yf, color = 'red')
plt.plot(Xf, regressor.predict(Xf), color = 'blue')
plt.title('Florida Spending vs Profit (Decision Tree Regression)')
plt.xlabel('Spending')
plt.ylabel('Profit')
plt.show()

# Predicting a new result with Decision Tree
Z1= regressor.predict([[9000000]])
print ("Florida = " ,Z1)

print("Hence, the profit of California would be more than Florida.")
```






```
Console 1/A
In [42]: runfile('C:/Users/me/Downloads/50starts.py', wdir='C:/Users/me/Downloads')
California = [191792.06]
Florida =  [191050.39]
Hence, the profit of California would be more than Florida.

In [43]:
```

Q2: Annual temperature between two industries is given. Predict the temperature in 2016 and 2017 using the past data of both country.

Polynomial Regression

Ans: *For the prediction of annual temperature of two industries I have applied both linear and polynomial regression but if we look carefully its evident that polynomial regression produces the best fit line better than linear regression since the loss is less in it. So the temperature of GISTEMP is more than*

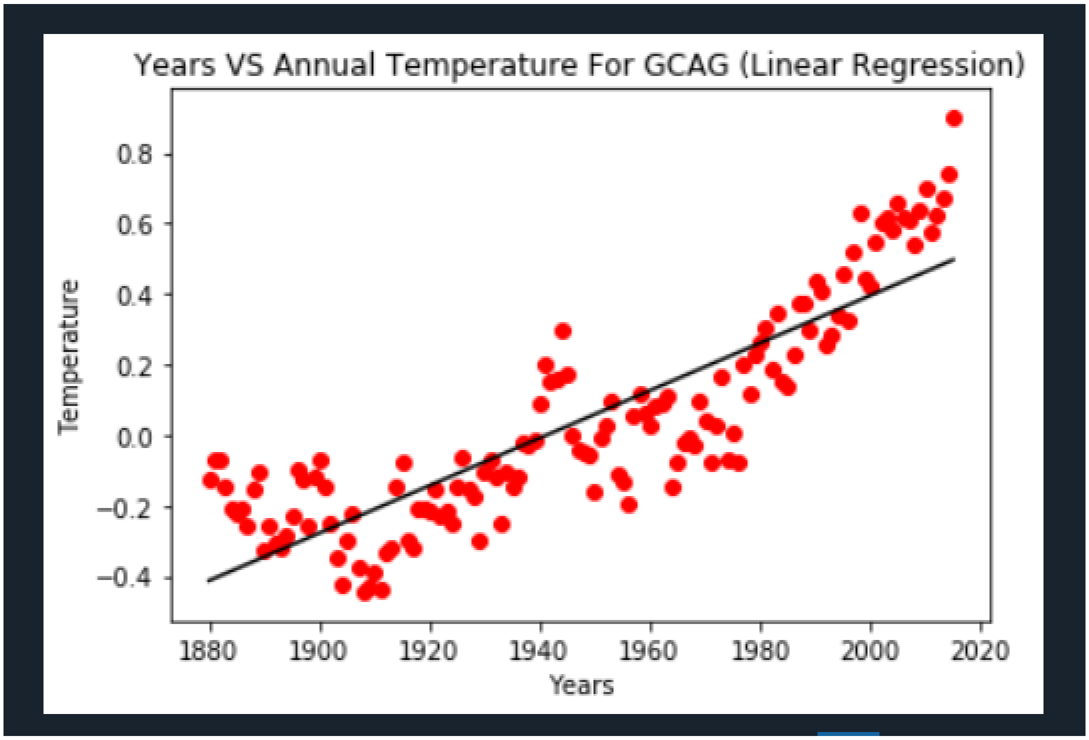
GCAG In 2016 and 2017.

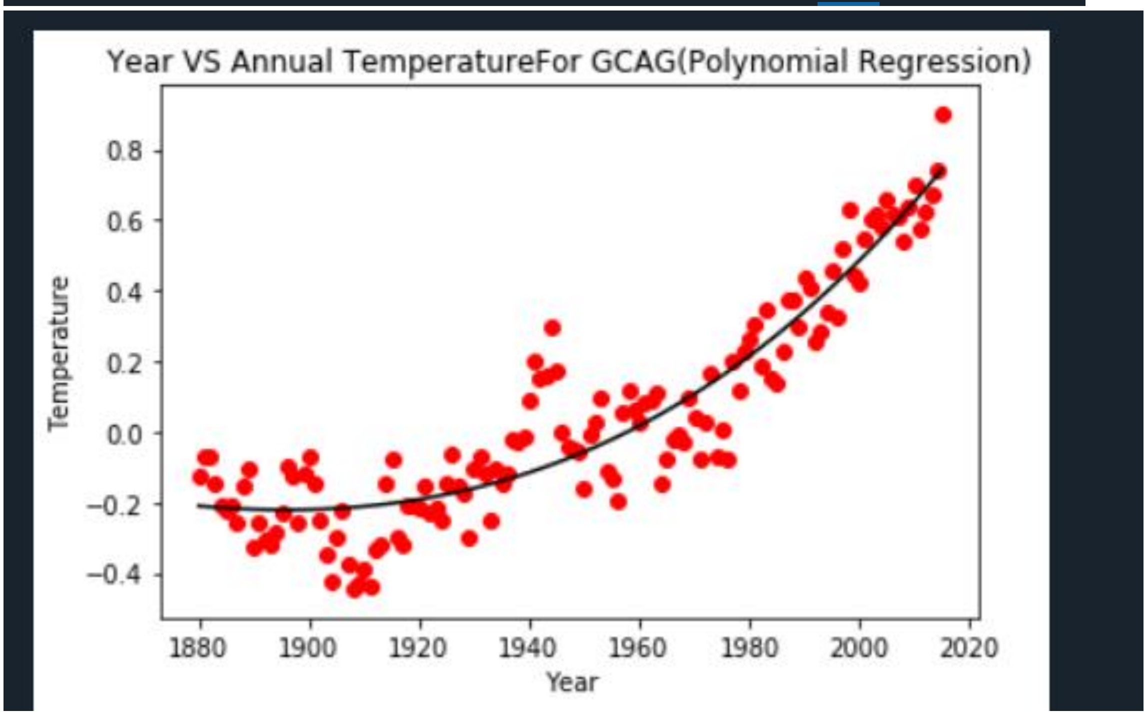
```python
#MACHINE LEARNING ASSIGNMENT3
#RUMAISA MARYAM

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('annual_temp.csv')

#FOR THE FIRST INDUSTRY GCAG
A = dataset.loc[(dataset.Source == 'GCAG'), ['Year']]
B = dataset.loc[(dataset.Source == 'GCAG'), ['Mean']]

# Splitting the dataset into the Training set and Test set
"""from sklearn.cross_validation import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)"""

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
polyreg = PolynomialFeatures(degree = 4)
A_poly = polyreg.fit_transform(A)
polyreg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, reg.predict(A), color = 'Black')
plt.title('Years VS Annual Temperature For GCAG (Linear Regression)')
plt.xlabel('Years')
plt.ylabel('Temperature')
```

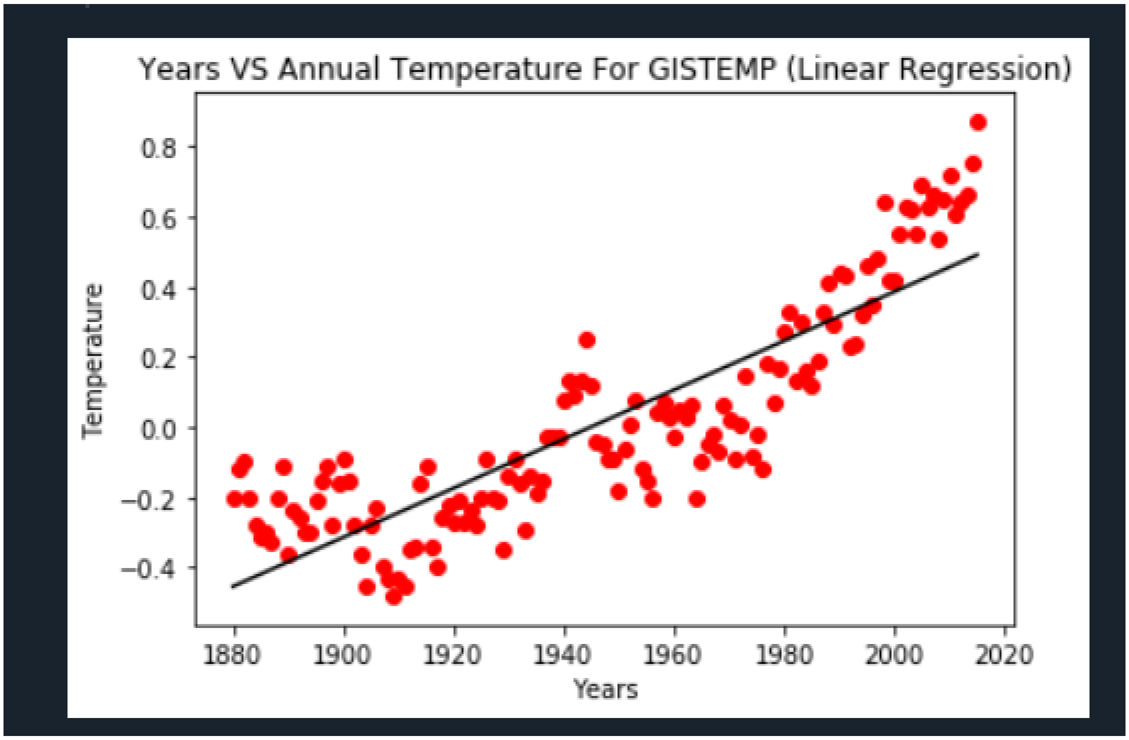
```python
"""from sklearn.cross_validation import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)"""

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
polyreg = PolynomialFeatures(degree = 4)
A_poly = polyreg.fit_transform(A)
polyreg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, reg.predict(A), color = 'Black')
plt.title('Years VS Annual Temperature For GCAG (Linear Regression)')
plt.xlabel('Years')
plt.ylabel('Temperature')
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_reg_2.predict(polyreg.fit_transform(A)), color = 'Black')
plt.title('Year VS Annual TemperatureFor GCAG(Polynomial Regression)')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.show()

# Predicting a new result with Linear Regression
X=reg.predict([[2016]])
X1=reg.predict([[2017]])
print("The result with linear regression for GCAG in 2016 is" , X)
print("The result with linear regression for GCAG in 2017 is" , X1)
```

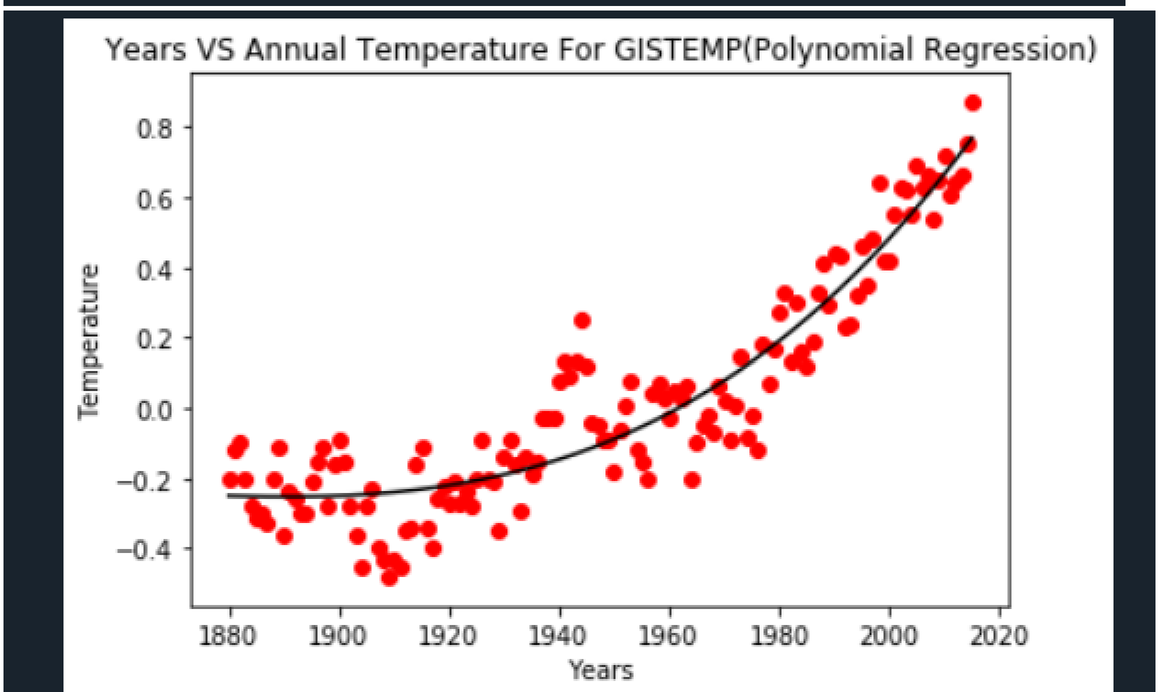
```python
X1=reg.predict([[2017]])
print("The result with linear regression for GCAG in 2016 is" , X)
print("The result with linear regression for GCAG in 2017 is" , X1)

# Predicting a new result with Polynomial Regression
Y=lin_reg_2.predict(polyreg.fit_transform([[2016]]))
Y1=lin_reg_2.predict(polyreg.fit_transform([[2017]]))
print("The result with polynomial regression for GCAG in 2016  is" ,  Y)
print("The result with polynomial regression for GCAG in 2017 is" ,  Y1)


#FOR THE SECOND INDUSTRY GISTEMP
A1 = dataset.loc[(dataset.Source == 'GISTEMP'), ['Year']]
B1 = dataset.loc[(dataset.Source == 'GISTEMP'), ['Mean']]

# Splitting the dataset into the Training set and Test set
"""from sklearn.cross_validation import train_test_split
A1_train, A1_test, B1_train, B1_test = train_test_split(A1, B1, test_size = 0.2, random_state = 0)"""

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(A1, B1)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
polyreg = PolynomialFeatures(degree = 4)
A1_poly = polyreg.fit_transform(A1)
polyreg.fit(A1_poly, B1)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A1_poly, B1)

# Visualising the Linear Regression results
plt.scatter(A1, B1, color = 'red')
plt.plot(A1, reg.predict(A1), color = 'Black')
plt.title('Years VS Annual Temperature For GISTEMP (Linear Regression)')
plt.xlabel('Years')
```

```python
reg.fit(A1, B1)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
polyreg = PolynomialFeatures(degree = 4)
A1_poly = polyreg.fit_transform(A1)
polyreg.fit(A1_poly, B1)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A1_poly, B1)

# Visualising the Linear Regression results
plt.scatter(A1, B1, color = 'red')
plt.plot(A1, reg.predict(A1), color = 'Black')
plt.title('Years VS Annual Temperature For GISTEMP (Linear Regression)')
plt.xlabel('Years')
plt.ylabel('Temperature')
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(A1, B1, color = 'red')
plt.plot(A1, lin_reg_2.predict(polyreg.fit_transform(A1)), color = 'Black')
plt.title('Years VS Annual Temperature For GISTEMP(Polynomial Regression)')
plt.xlabel('Years')
plt.ylabel('Temperature')
plt.show()

# Predicting a new result with Linear Regression
X2=reg.predict([[2016]])
X3=reg.predict([[2017]])
print("The result with linear regression for GCAG in 2016 is" , X2)
print("The result with linear regression for GCAG in 2017 is" , X3)

# Predicting a new result with Polynomial Regression
Y2=lin_reg_2.predict(polyreg.fit_transform([[2016]]))
Y3=lin_reg_2.predict(polyreg.fit_transform([[2017]]))
print("The result with polynomial regression for GCAG in 2016  is" ,  Y2)
print("The result with polynomial regression for GCAG in 2017 is" ,  Y3)
```


Years VS Annual Temperature For GCAG (Linear Regression)
Temperature
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
1880
1900
1920
1940
1960
1980
2000
2020
Years


Year VS Annual TemperatureFor GCAG(Polynomial Regression)
Temperature
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
1880
1900
1920
1940
1960
1980
2000
2020
Year






```
In [43]: runfile('C:/Users/me/Downloads/annualtemp.py', wdir='C:/Users/me/Downloads')
The result with linear regression for GCAG in 2016 is [[0.50298425]]
The result with linear regression for GCAG in 2017 is [[0.50972011]]
The result with polynomial regression for GCAG in 2016  is [[0.76231028]]
The result with polynomial regression for GCAG in 2017 is [[0.78149969]]
The result with linear regression for GCAG in 2016 is [[0.49777778]]
The result with linear regression for GCAG in 2017 is [[0.50477625]]
The result with polynomial regression for GCAG in 2016  is [[0.78885745]]
The result with polynomial regression for GCAG in 2017 is [[0.81039365]]
```

Q3: Data of global production of CO2 of a place is given between 1970s to 2010. Predict the CO2 production for the years 2011, 2012 and 2013 using the old data set.

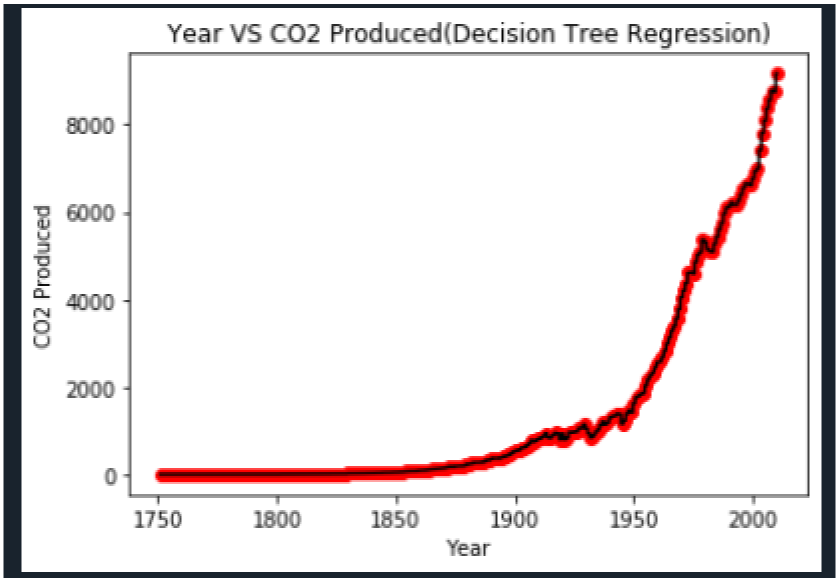
Polynomial Regression

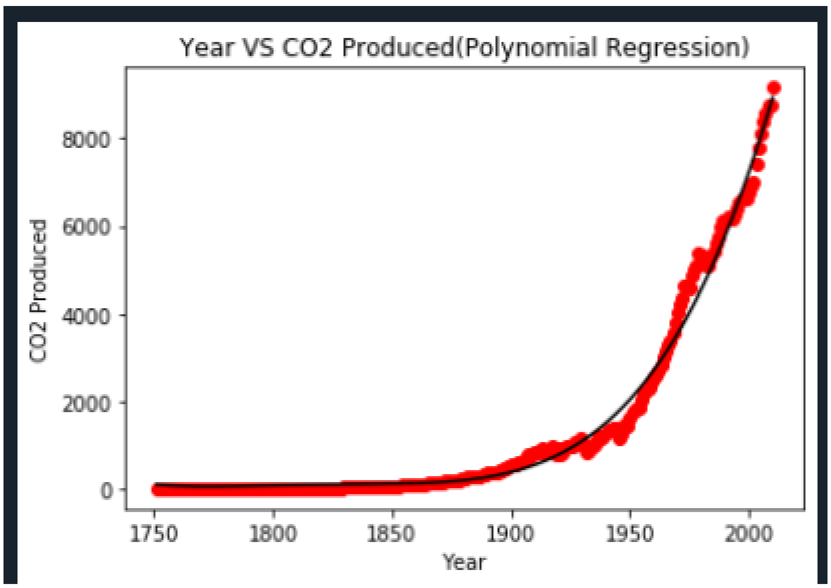
Ans: *For this dataset I have applied all three regression but when seeing the results its evident that* ***polynomial regression*** *is the best form of regression here since it produces the best fit line better than linear regression and decision tree. Also the result for decision tree remains the same for different values of prediction.*

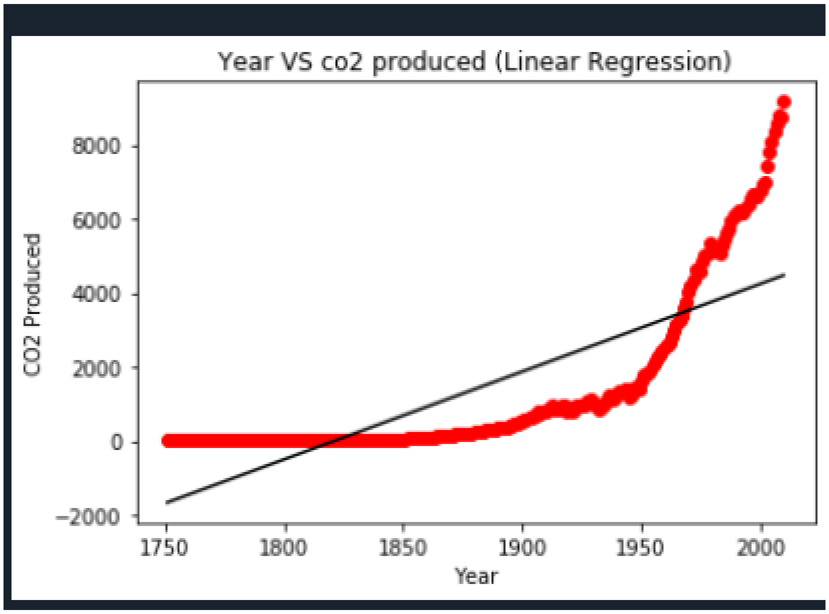
```python
#MACHINE LEARNING ASSIGNMENT3
#RUMAISA MARYAM

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('global_co2.csv')
A = dataset.iloc[:,0:1].values
B = dataset.iloc[:, 1].values


# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)


# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
A_poly = poly_reg.fit_transform(A)
poly_reg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_regressor.predict(A), color = 'Black')
plt.title('Year VS co2 produced (Linear Regression)')
plt.xlabel('Year')
```

```python
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_regressor.predict(A), color = 'Black')
plt.title('Year VS co2 produced (Linear Regression)')
plt.xlabel('Year')
plt.ylabel('CO2 Produced')
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_reg_2.predict(poly_reg.fit_transform(A)), color = 'Black')
plt.title('Year VS co2 produced (Polynomial Regression)')
plt.xlabel('Year')
plt.ylabel('CO2 Produced')
plt.show()

# Visualising the Polynomial Regression results (for higher resolution and smoother curve)
A_grid = np.arange(min(A), max(A), 0.1)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, lin_reg_2.predict(poly_reg.fit_transform(A_grid)), color = 'Black')
plt.title('Year VS CO2 Produced(Polynomial Regression)')
plt.xlabel('Year')
plt.ylabel('CO2 Produced')
plt.show()

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
```

```python
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, regressor.predict(A_grid), color = 'BLACK')
plt.title('Year VS CO2 Produced(Decision Tree Regression)')
plt.xlabel('Year')
plt.ylabel('CO2 Produced')
plt.show()

# Predicting a new result with Linear Regression
X=lin_regressor.predict([[2011]])
X1=lin_regressor.predict([[2012]])
X2=lin_regressor.predict([[2013]])
print("The result with linear regression for co2 produced in  2011 is" , X)
print("The result with linear regression for co2 produced in 2012 is" , X1)
print("The result with linear regression for co2 produced in  2013 is" , X2)

# Predicting a new result with Polynomial Regression
Y=lin_reg_2.predict(poly_reg.fit_transform([[2011]]))
Y1=lin_reg_2.predict(poly_reg.fit_transform([[2012]]))
Y2=lin_reg_2.predict(poly_reg.fit_transform([[2013]]))
print("The result with polynomial regression for co2 produced in 2011  is" ,  Y)
print("The result with polynomial regression for co2 produced in 2012 is" ,  Y1)
print("The result with polynomial regression for co2 produced in 2013 is" ,  Y2)

# Predicting a new result with Decision Tree
Z = regressor.predict([[2011]])
Z1 = regressor.predict([[2012]])
Z2= regressor.predict([[2013]])
print("The result with decision tree for co2 produced in 2011  is" , Z)
print("The result with decision tree for co2 produced in 2012 is" , Z1)
print("The result with decision tree for co2 produced in 2013  is" , Z2)
```


Year VS CO2 Produced(Decision Tree Regression)
CO2 Produced
Year
0
2000
4000
6000
8000
1750
1800
1850
1900
1950
2000


Year VS CO2 Produced(Polynomial Regression)
CO2 Produced
Year
0
2000
4000
6000
8000
1750
1800
1850
1900
1950
2000




```
In [45]: runfile('C:/Users/me/Downloads/globalCO2.py', wdir='C:/Users/me/Downloads')
The result with linear regression for co2 produced in  2011 is [4494.86418176]
The result with linear regression for co2 produced in 2012 is [4518.55824859]
The result with linear regression for co2 produced in  2013 is [4542.25231541]
The result with polynomial regression for co2 produced in 2011  is [9138.92033747]
The result with polynomial regression for co2 produced in 2012 is [9329.39530137]
The result with polynomial regression for co2 produced in 2013 is [9522.85598981]
The result with decision tree for co2 produced in 2011  is [9167.]
The result with decision tree for co2 produced in 2012 is [9167.]
The result with decision tree for co2 produced in 2013  is [9167.]
```

Q4: Housing price according to the ID is assigned to every-house. Perform future analysis where when ID is inserted the housing price is displayed.

Polynomial Regression

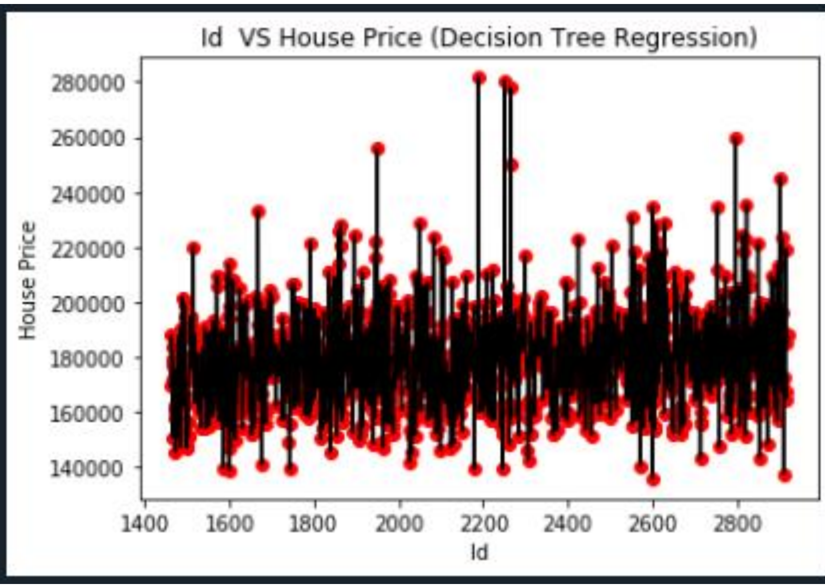
Ans: *For this dataset I have used all forms of regression but observing closely its NOTED that* ***polynomial regression*** *provides the best and accurate result since it produces the best fit line better than linear regression. As for decision tree its observed that the prediction for different set of IDs remain the* *same so its not useful at all here.*

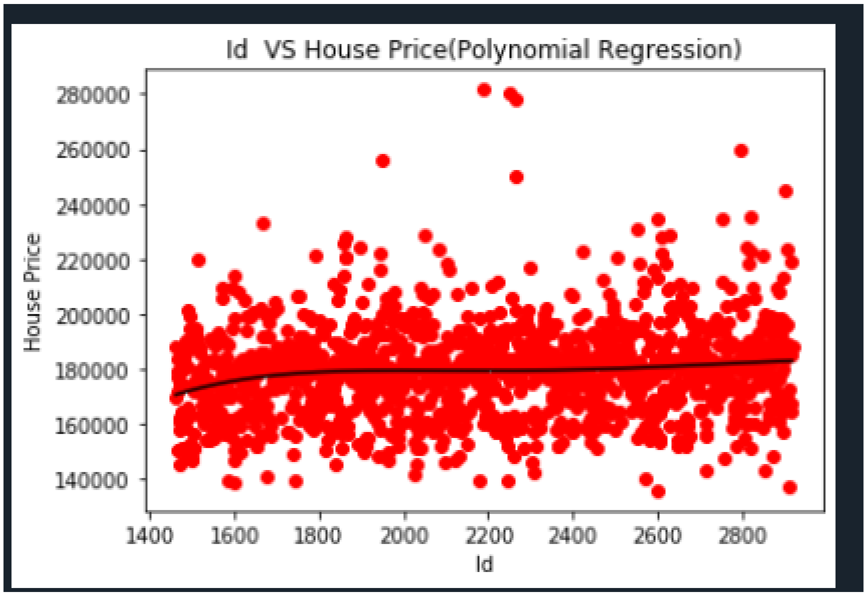
```
#MACHINE LEARNING ASSIGNMENT3
#RUMAISA MARYAM

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the Annual Temperature dataset
dataset = pd.read_csv('housing_price.csv')
A = dataset.iloc[:,0:1].values
B = dataset.iloc[:, 1].values


# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)


# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
A_poly = poly_reg.fit_transform(A)
poly_reg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_regressor.predict(A), color = 'Black')
plt.title('Id VS House Price (Linear Regression)')
plt.xlabel('Id')
```

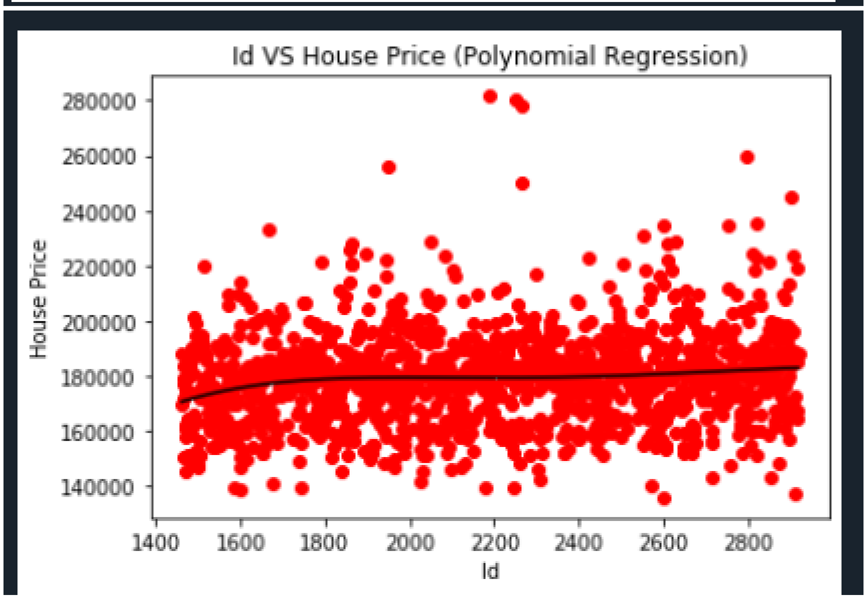
```

# Visualising the Polynomial Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_reg_2.predict(poly_reg.fit_transform(A)), color = 'Black')
plt.title('Id VS House Price (Polynomial Regression)')
plt.xlabel('Id')
plt.ylabel('House Price')
plt.show()

# Visualising the Polynomial Regression results (for higher resolution and smoother curve)
A_grid = np.arange(min(A), max(A), 0.1)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, lin_reg_2.predict(poly_reg.fit_transform(A_grid)), color = 'Black')
plt.title('Id  VS House Price(Polynomial Regression)')
plt.xlabel('Id')
plt.ylabel('House Price')
plt.show()

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, regressor.predict(A_grid), color = 'BLACK')
plt.title('Id  VS House Price (Decision Tree Regression)')
plt.xlabel('Id')
plt.ylabel('House Price')
plt.show()

# Predicting a new result with Linear Regression
X=lin_regressor.predict([[2920]])
```

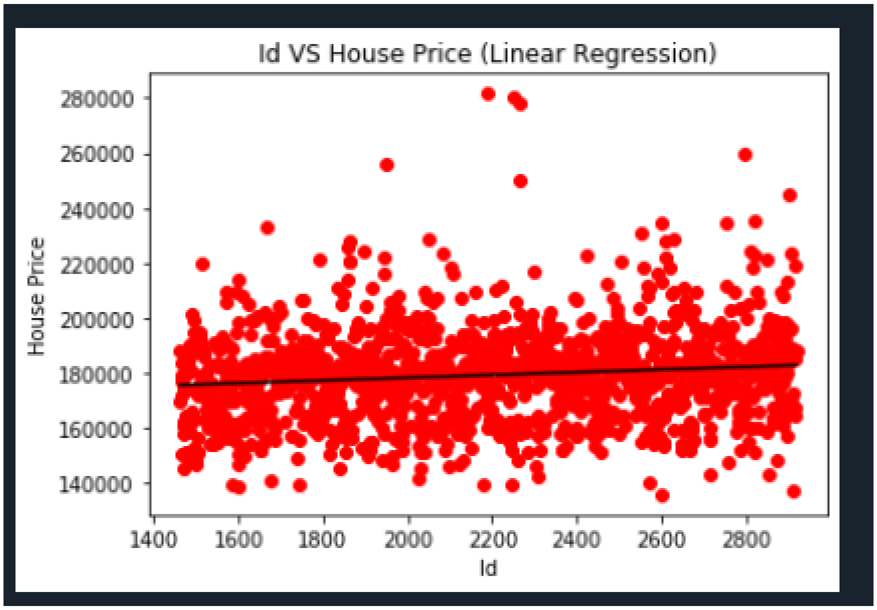
nomial_regression_headweightset.py × ann.py × ann_new.py × polynomial_regression.py × simple_linear_regression.py × Annual_temp.py × houseprice.py* ×

```
plt.xlabel('Id')
plt.ylabel('House Price')
plt.show()

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, regressor.predict(A_grid), color = 'BLACK')
plt.title('Id  VS House Price (Decision Tree Regression)')
plt.xlabel('Id')
plt.ylabel('House Price')
plt.show()

# Predicting a new result with Linear Regression
X=lin_regressor.predict([[2920]])
print("The result with linear regression for house id 2920 is" , X)


# Predicting a new result with Polynomial Regression
Y=lin_reg_2.predict(poly_reg.fit_transform([[2920]]))
print("The result with polynomial regression for house id 2920  is" ,  Y)


# Predicting a new result with Decision Tree
Z = regressor.predict([[2920]])
Z1= regressor.predict([[2929]])
print("The result with decision tree for house id 2920  is" , Z)
print("The result with decision tree for house id 2929  is" , Z1)


```




Id VS House Price(Polynomial Regression)
House Price
280000
260000
240000
220000
200000
180000
160000
140000
1400
1600
1800
2000
2200
2400
2600
2800
Id


Id VS House Price (Polynomial Regression)
House Price
280000
260000
240000
220000
200000
180000
160000
140000
1400
1600
1800
2000
2200
2400
2600
2800
Id




```
In [47]: runfile('C:/Users/me/Downloads/houseprice.py', wdir='C:/Users/me/Downloads')
The result with linear regression for house id 2920 is [182794.79499595]
The result with polynomial regression for house id 2920  is [182886.73493585]
The result with decision tree for house id 2920  is [187741.8667]
The result with decision tree for house id 2929  is [187741.8667]

In [48]:
```

Q5: Data of monthly experience and income distribution of different employs is given. Perform regression.

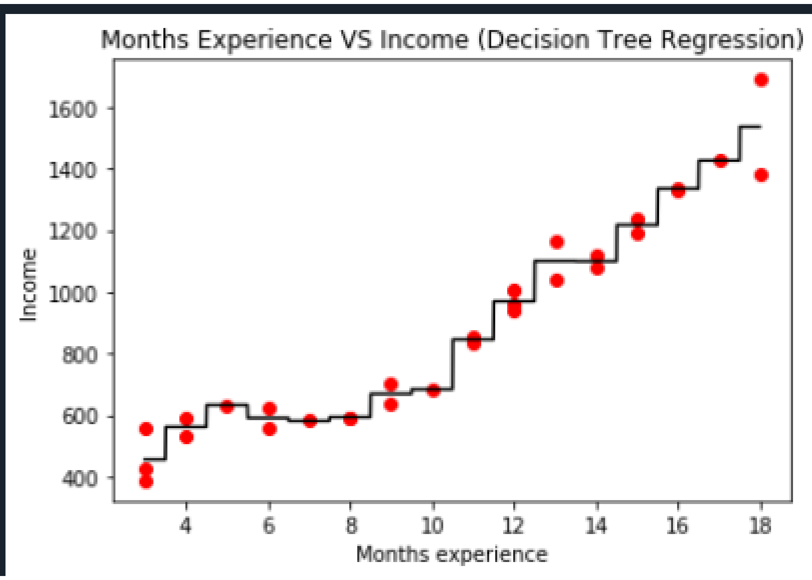
Polynomial Regression

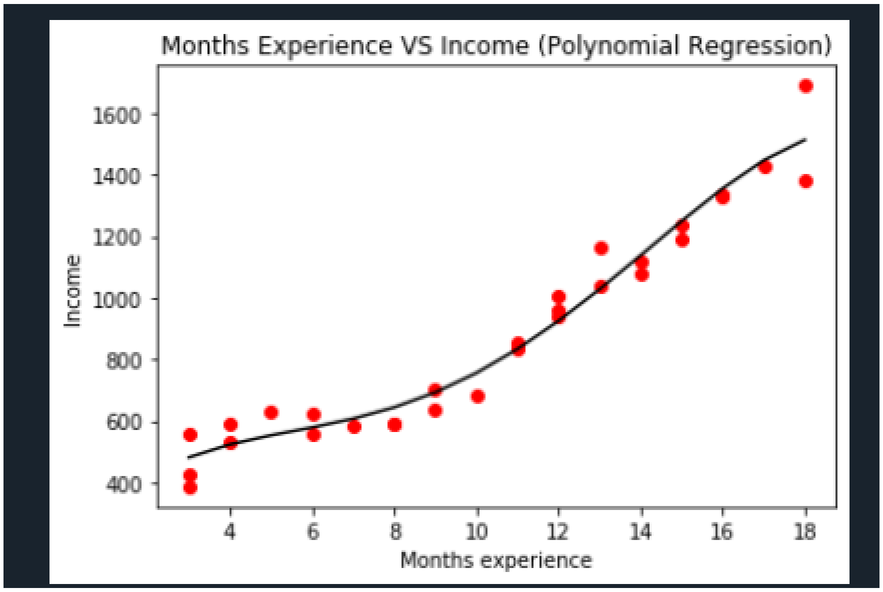
Ans: *For this dataset I have used all three regressions but by observing closely its found out that* ***polynomial regression*** *is the best type of regression since it produces the best fit line better than linear regression. As for decision tree its found out that it produces the same answers for different values,so it cant be used here.*

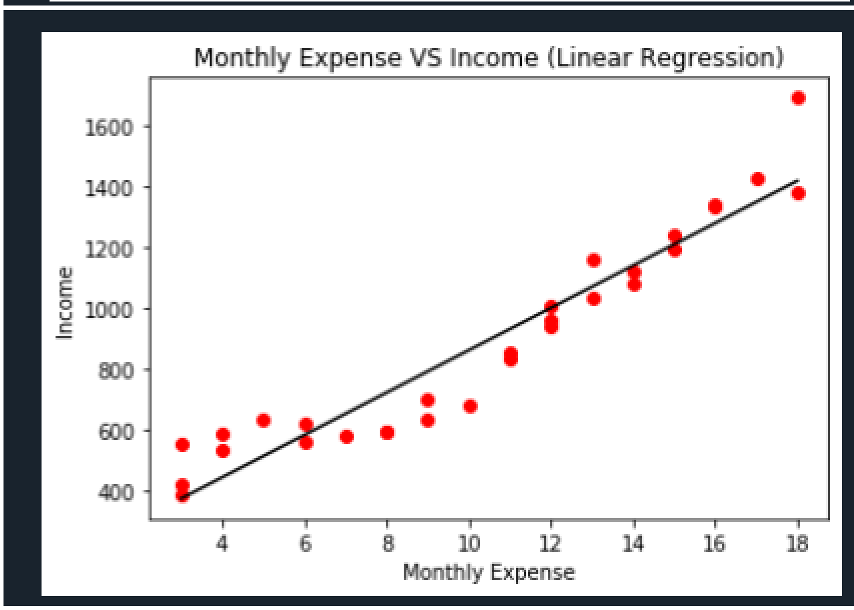
```python
#MACHINE LEARNING ASSIGNMENT3
#RUMAISA MARYAM

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the Annual Temperature dataset
dataset = pd.read_csv('expense.csv')
A = dataset.iloc[:,0:1].values
B = dataset.iloc[:, 1:2].values


# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
A_train, A_test, B_train, B_test = train_test_split(A, B, test_size = 0.2, random_state = 0)


# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(A, B)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 4)
A_poly = poly_reg.fit_transform(A)
poly_reg.fit(A_poly, B)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(A_poly, B)

# Visualising the Linear Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_regressor.predict(A), color = 'Black')
plt.title('Monthly Expense VS Income (Linear Regression)')
plt.xlabel('Monthly Expense')
```

```python

# Visualising the Polynomial Regression results
plt.scatter(A, B, color = 'red')
plt.plot(A, lin_reg_2.predict(poly_reg.fit_transform(A)), color = 'Black')
plt.title('Months Experience VS Income (Polynomial Regression)')
plt.xlabel('Months experience')
plt.ylabel('Income')
plt.show()

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, regressor.predict(A_grid), color = 'BLACK')
plt.title('Months Experience VS Income (Decision Tree Regression)')
plt.xlabel('Months experience')
plt.ylabel('Income')
plt.show()

# Predicting a new result with Linear Regression
X=lin_regressor.predict([[19]])
print("The result with linear regression for 19 months of experince  is" , X)


# Predicting a new result with Polynomial Regression
Y=lin_reg_2.predict(poly_reg.fit_transform([[19]]))
Y1=lin_reg_2.predict(poly_reg.fit_transform([[50]]))
print("The result with polynomial regression for 19 months of experince  is" ,  Y)
print("The result with polynomial regression for 50 months of experince  is" ,  Y1)


# Predicting a new result with Decision Tree
```

```
# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(A, B)

# Visualising the Decision Tree Regression results (higher resolution)
A_grid = np.arange(min(A), max(A), 0.01)
A_grid = A_grid.reshape((len(A_grid), 1))
plt.scatter(A, B, color = 'red')
plt.plot(A_grid, regressor.predict(A_grid), color = 'BLACK')
plt.title('Months Experience VS Income (Decision Tree Regression)')
plt.xlabel('Months experience')
plt.ylabel('Income')
plt.show()

# Predicting a new result with Linear Regression
X=lin_regressor.predict([[19]])
print("The result with linear regression for 19 months of experince  is" , X)


# Predicting a new result with Polynomial Regression
Y=lin_reg_2.predict(poly_reg.fit_transform([[19]]))
Y1=lin_reg_2.predict(poly_reg.fit_transform([[50]]))
print("The result with polynomial regression for 19 months of experince  is" , Y)
print("The result with polynomial regression for 50 months of experince  is" , Y1)


# Predicting a new result with Decision Tree
Z = regressor.predict([[19]])
Z1 = regressor.predict([[50]])
print("The result with decision tree for 19 months of experience is" , Z)
print("The result with decision tree for 50 months of experience is" , Z1)
```








```
In [51]: runfile('C:/Users/me/Downloads/experience.py', wdir='C:/Users/me/Downloads')
The result with linear regression for 19 months of experince  is [[1486.2554603]]
The result with polynomial regression for 19 months of experince  is [[1543.11034559]]
The result with polynomial regression for 50 months of experince  is [[-153767.94019964]]
The result with decision tree for 19 months of experience is [1536.]
The result with decision tree for 50 months of experience is [1536.]

In [52]: |
```