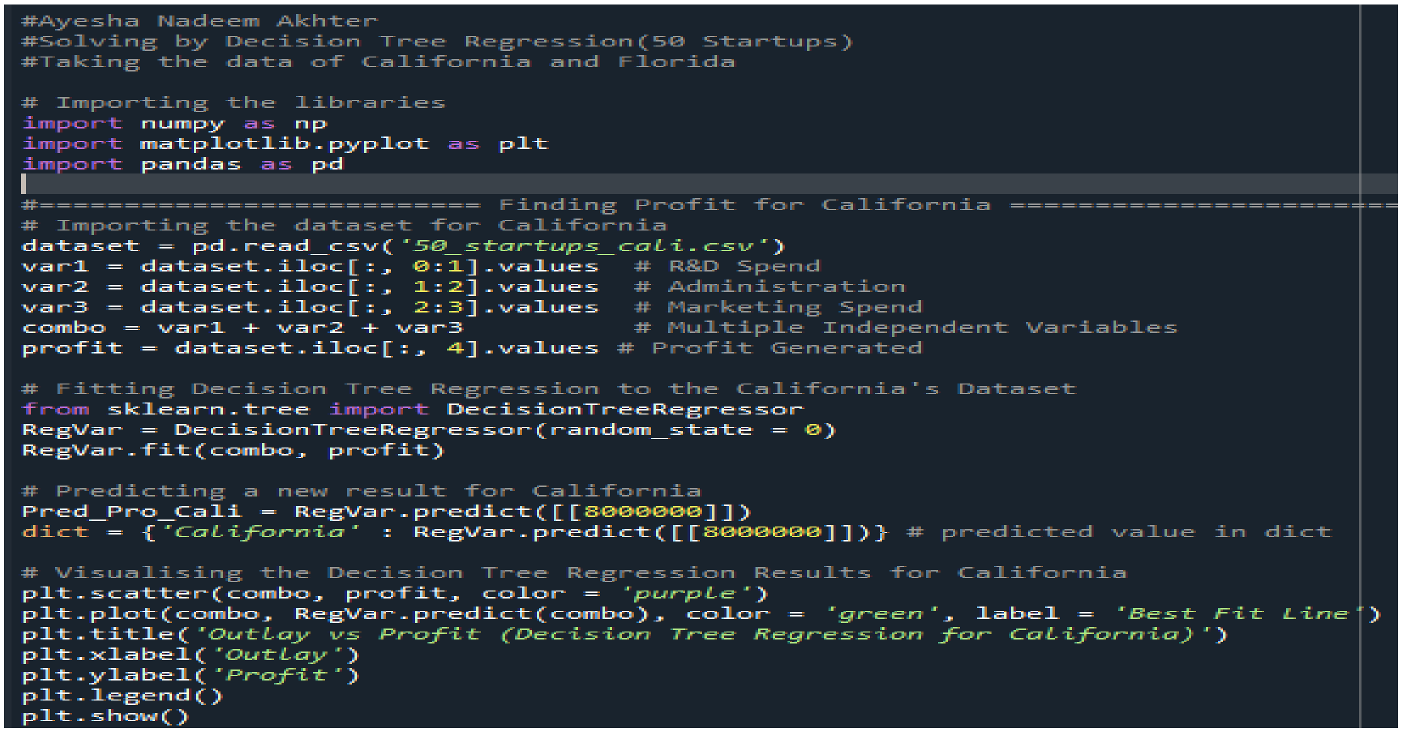
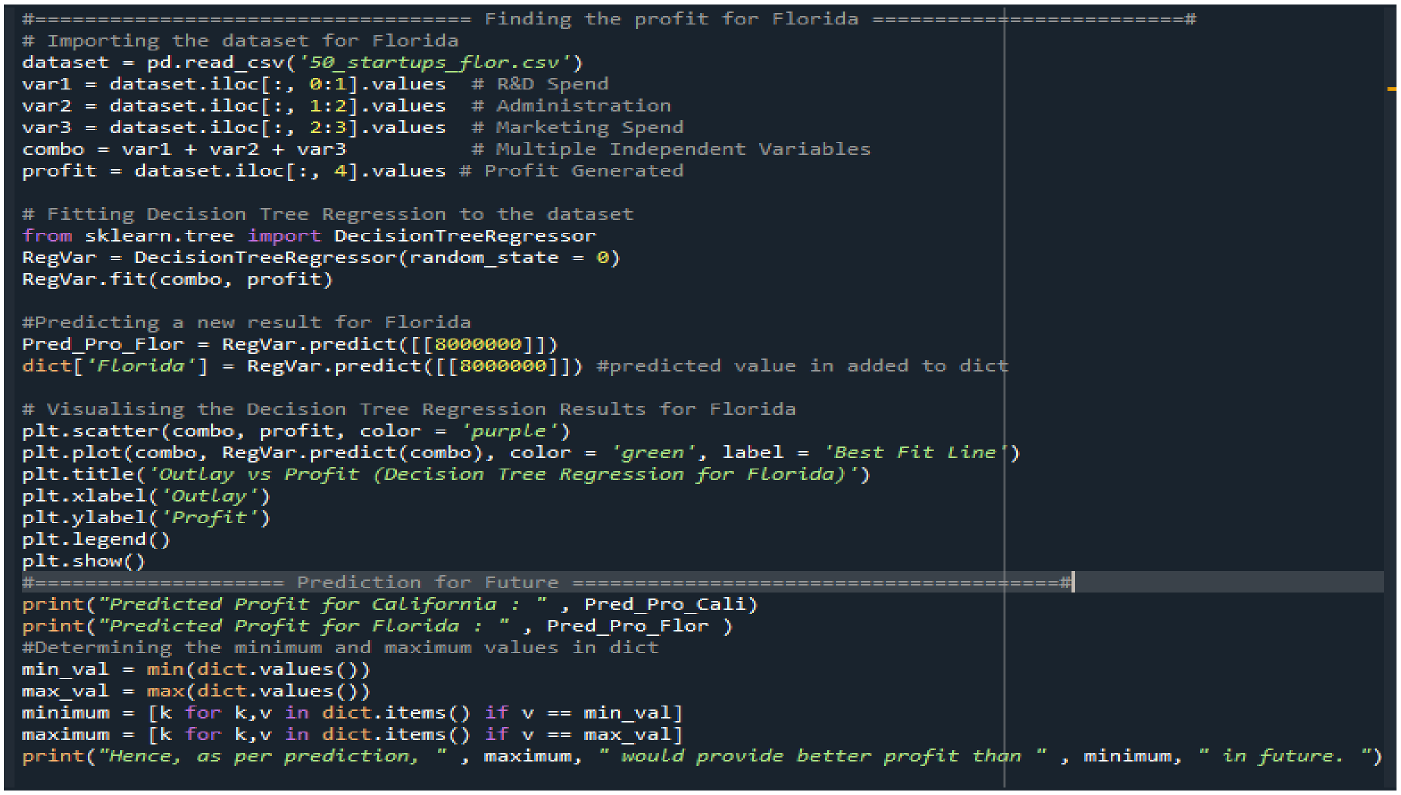
# ASSIGNMENT # 3

## Q.1) TAKE 50 STARTUPS OF ANY TWO COUNTRIES AND FIND OUT WHICH COUNTRY IS GOING TO PROVIDE BEST PROFIT IN FUTURE.

**CODE:**

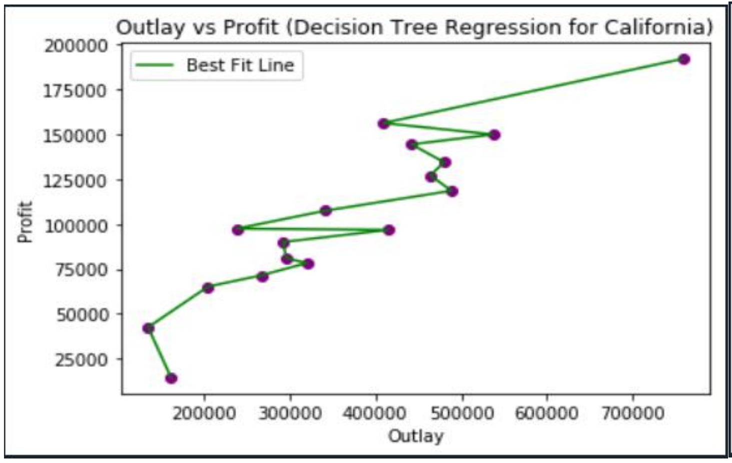


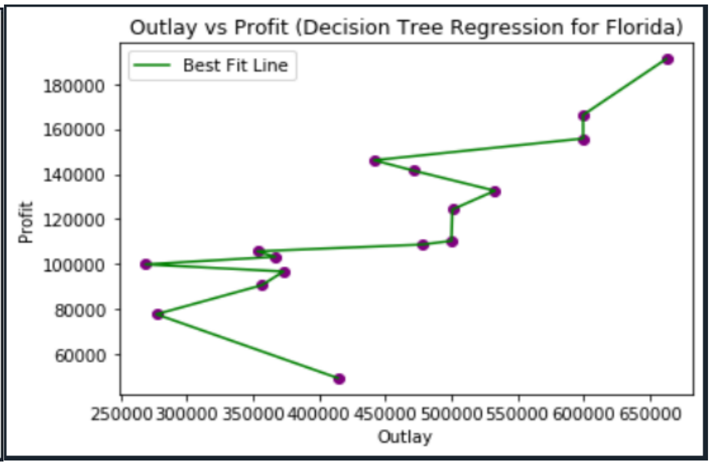
```
#Ayesha Nadeem Akhter
#Solving by Decision Tree Regression(50 Startups)
#Taking the data of California and Florida

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

#========================== Finding Profit for California ========================
# Importing the dataset for California
dataset = pd.read_csv('50_startups_cali.csv')
var1 = dataset.iloc[:, 0:1].values  # R&D Spend
var2 = dataset.iloc[:, 1:2].values  # Administration
var3 = dataset.iloc[:, 2:3].values  # Marketing Spend
combo = var1 + var2 + var3          # Multiple Independent Variables
profit = dataset.iloc[:, 4].values # Profit Generated

# Fitting Decision Tree Regression to the California's Dataset
from sklearn.tree import DecisionTreeRegressor
RegVar = DecisionTreeRegressor(random_state = 0)
RegVar.fit(combo, profit)

# Predicting a new result for California
Pred_Pro_Cali = RegVar.predict([[8000000]])
dict = {'California' : RegVar.predict([[8000000]])} # predicted value in dict

# Visualising the Decision Tree Regression Results for California
plt.scatter(combo, profit, color = 'purple')
plt.plot(combo, RegVar.predict(combo), color = 'green', label = 'Best Fit Line')
plt.title('Outlay vs Profit (Decision Tree Regression for California)')
plt.xlabel('Outlay')
plt.ylabel('Profit')
plt.legend()
plt.show()
```



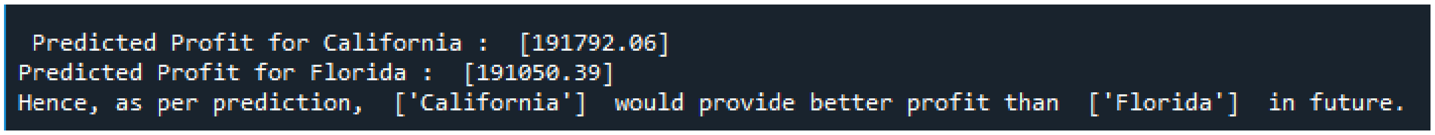
```
#================================== Finding the profit for Florida ========================#
# Importing the dataset for Florida
dataset = pd.read_csv('50_startups_flor.csv')
var1 = dataset.iloc[:, 0:1].values  # R&D Spend
var2 = dataset.iloc[:, 1:2].values  # Administration
var3 = dataset.iloc[:, 2:3].values  # Marketing Spend
combo = var1 + var2 + var3          # Multiple Independent Variables
profit = dataset.iloc[:, 4].values # Profit Generated

# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
RegVar = DecisionTreeRegressor(random_state = 0)
RegVar.fit(combo, profit)

#Predicting a new result for Florida
Pred_Pro_Flor = RegVar.predict([[8000000]])
dict['Florida'] = RegVar.predict([[8000000]]) #predicted value in added to dict

# Visualising the Decision Tree Regression Results for Florida
plt.scatter(combo, profit, color = 'purple')
plt.plot(combo, RegVar.predict(combo), color = 'green', label = 'Best Fit Line')
plt.title('Outlay vs Profit (Decision Tree Regression for Florida)')
plt.xlabel('Outlay')
plt.ylabel('Profit')
plt.legend()
plt.show()
#==================== Prediction for Future ======================================#
print("Predicted Profit for California : " , Pred_Pro_Cali)
print("Predicted Profit for Florida : " , Pred_Pro_Flor )
#Determining the minimum and maximum values in dict
min_val = min(dict.values())
max_val = max(dict.values())
minimum = [k for k,v in dict.items() if v == min_val]
maximum = [k for k,v in dict.items() if v == max_val]
print("Hence, as per prediction, " , maximum, " would provide better profit than " , minimum, " in future. ")
```

## GRAPHS:



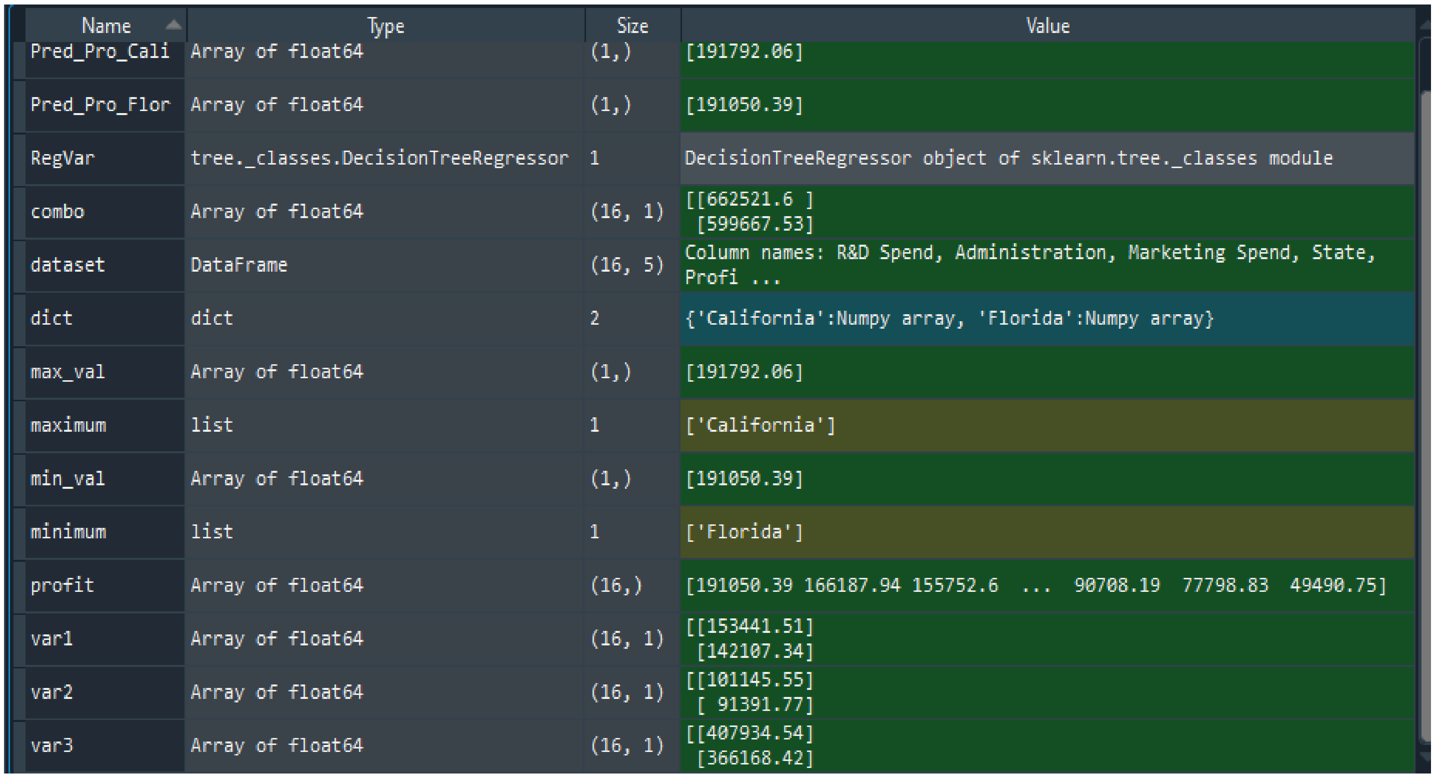


## CONSOLE:



```
 Predicted Profit for California :  [191792.06]
Predicted Profit for Florida :  [191050.39]
Hence, as per prediction,  ['California']  would provide better profit than  ['Florida']  in future.
```

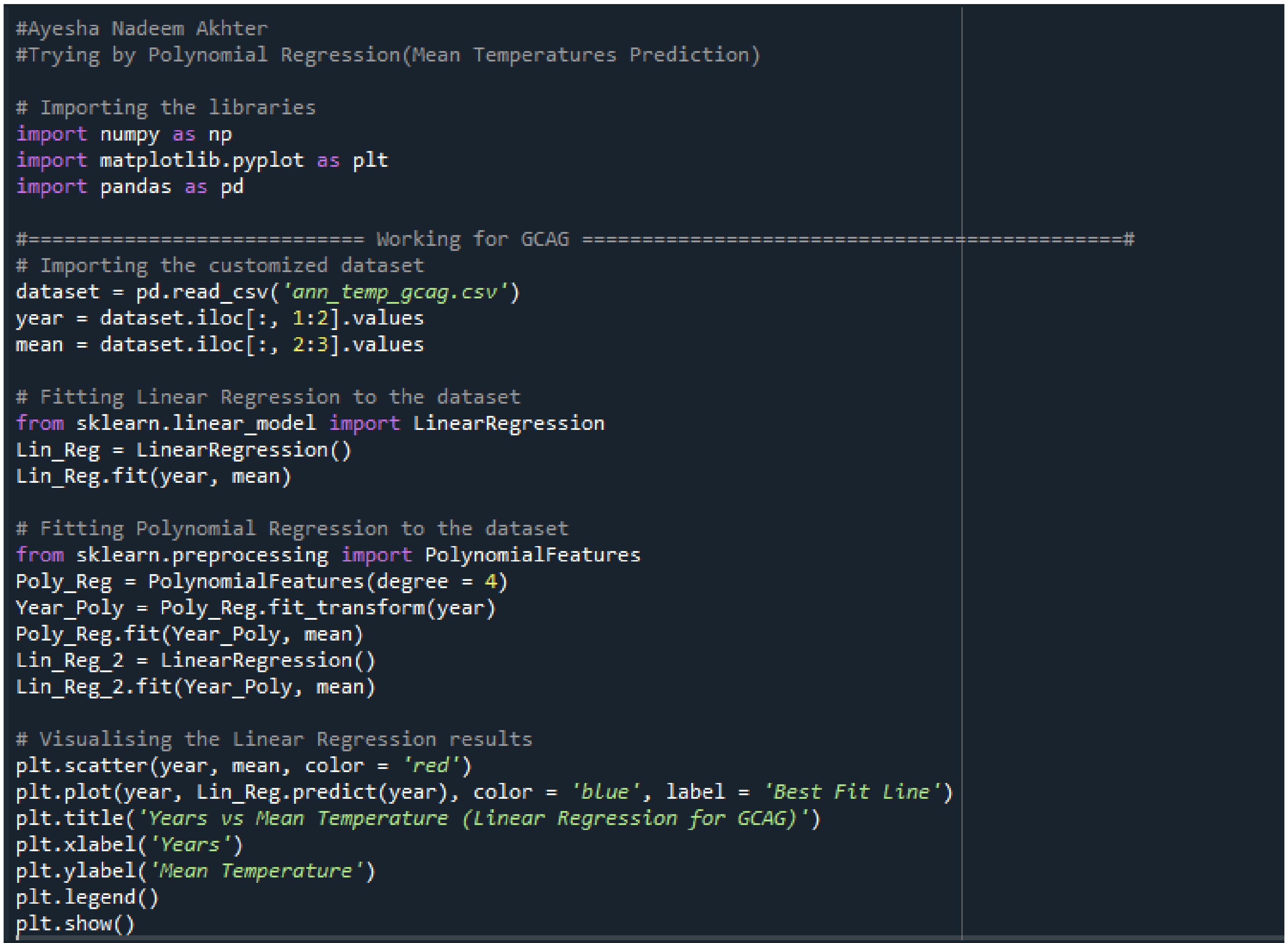
## VARIABLE EXPLORER:



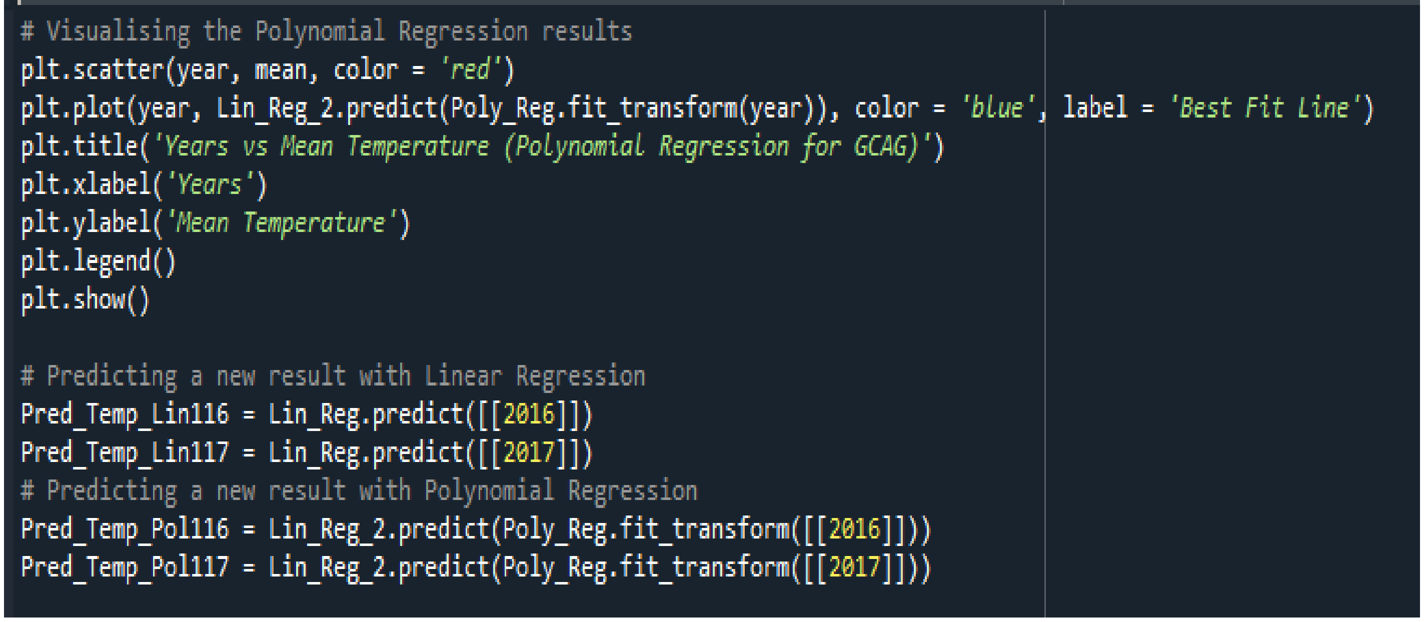
| Name | Type | Size | Value |
|---|---|---|---|
| Pred_Pro_Cali | Array of float64 | (1,) | [191792.06] |
| Pred_Pro_Flor | Array of float64 | (1,) | [191050.39] |
| RegVar | tree._classes.DecisionTreeRegressor | 1 | DecisionTreeRegressor object of sklearn.tree._classes module |
| combo | Array of float64 | (16, 1) | [[662521.6 ]<br>[599667.53] |
| dataset | DataFrame | (16, 5) | Column names: R&D Spend, Administration, Marketing Spend, State, Profi ... |
| dict | dict | 2 | {'California':Numpy array, 'Florida':Numpy array} |
| max_val | Array of float64 | (1,) | [191792.06] |
| maximum | list | 1 | ['California'] |
| min_val | Array of float64 | (1,) | [191050.39] |
| minimum | list | 1 | ['Florida'] |
| profit | Array of float64 | (16,) | [191050.39 166187.94 155752.6  ...  90708.19  77798.83  49490.75] |
| var1 | Array of float64 | (16, 1) | [[153441.51]<br>[142107.34] |
| var2 | Array of float64 | (16, 1) | [[101145.55]<br>[ 91391.77] |
| var3 | Array of float64 | (16, 1) | [[407934.54]<br>[366168.42] |

## Q.2) ANNUAL TEMPERATURE BETWEEN TWO INDUSTRIES IS GIVEN. PREDICT THE TEMPERATURE IN 2016 AND 2017 USING THE PAST DATA OF BOTH COUNTRY.

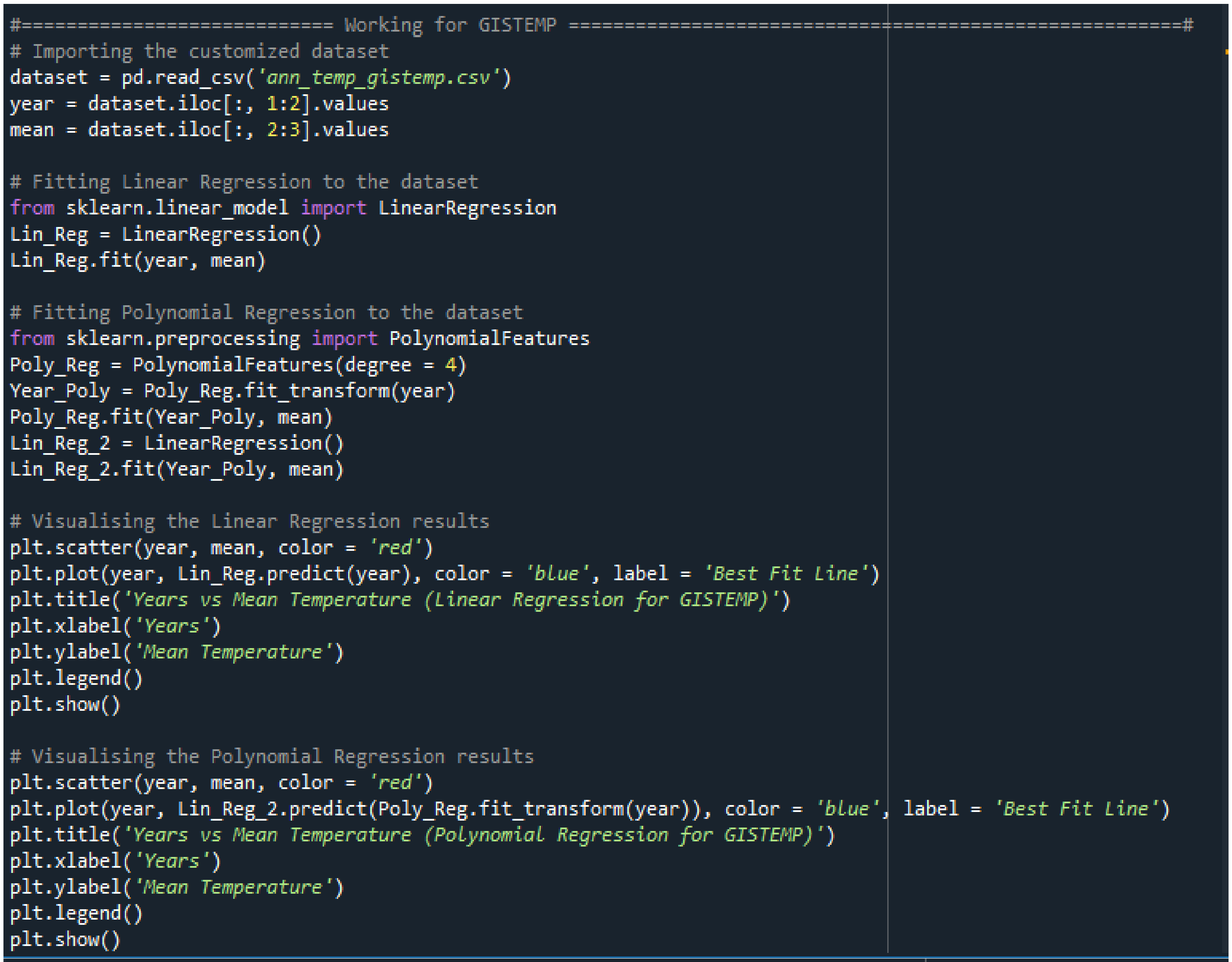
**CODE:**



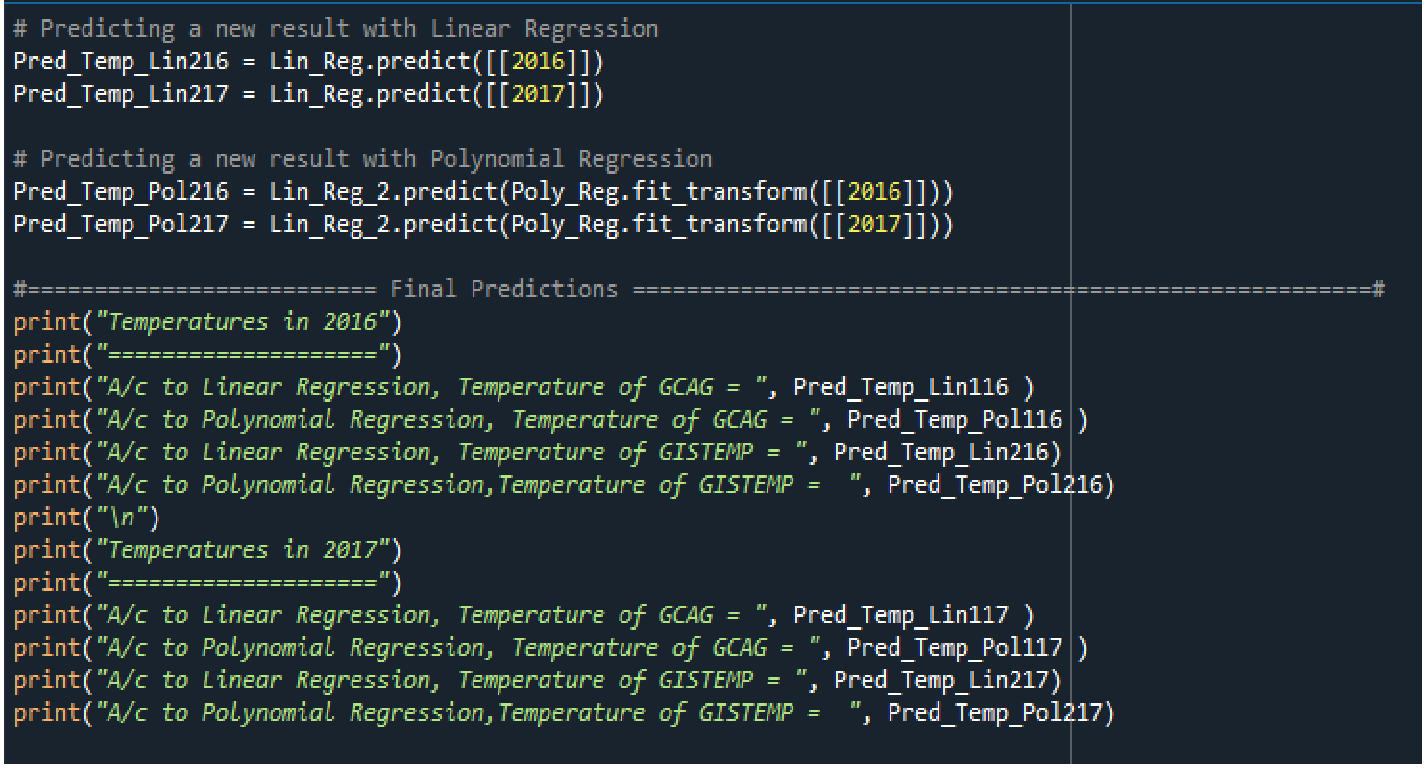
```python
#Ayesha Nadeem Akhter
#Trying by Polynomial Regression(Mean Temperatures Prediction)

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

#============================ Working for GCAG =============================================#
# Importing the customized dataset
dataset = pd.read_csv('ann_temp_gcag.csv')
year = dataset.iloc[:, 1:2].values
mean = dataset.iloc[:, 2:3].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
Lin_Reg = LinearRegression()
Lin_Reg.fit(year, mean)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
Poly_Reg = PolynomialFeatures(degree = 4)
Year_Poly = Poly_Reg.fit_transform(year)
Poly_Reg.fit(Year_Poly, mean)
Lin_Reg_2 = LinearRegression()
Lin_Reg_2.fit(Year_Poly, mean)

# Visualising the Linear Regression results
plt.scatter(year, mean, color = 'red')
plt.plot(year, Lin_Reg.predict(year), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs Mean Temperature (Linear Regression for GCAG)')
plt.xlabel('Years')
plt.ylabel('Mean Temperature')
plt.legend()
plt.show()
```



```python
# Visualising the Polynomial Regression results
plt.scatter(year, mean, color = 'red')
plt.plot(year, Lin_Reg_2.predict(Poly_Reg.fit_transform(year)), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs Mean Temperature (Polynomial Regression for GCAG)')
plt.xlabel('Years')
plt.ylabel('Mean Temperature')
plt.legend()
plt.show()

# Predicting a new result with Linear Regression
Pred_Temp_Lin116 = Lin_Reg.predict([[2016]])
Pred_Temp_Lin117 = Lin_Reg.predict([[2017]])
# Predicting a new result with Polynomial Regression
Pred_Temp_Pol116 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2016]]))
Pred_Temp_Pol117 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2017]]))
```

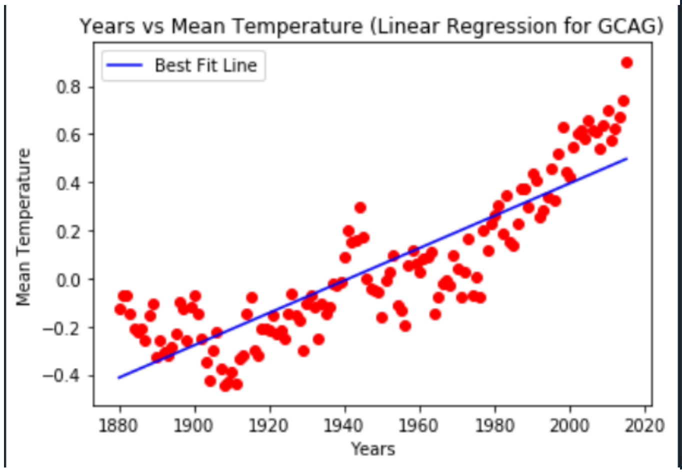


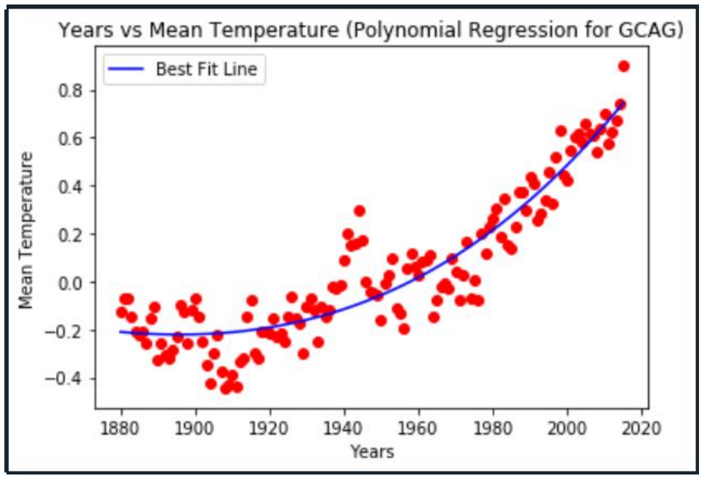
```python
#============================ Working for GISTEMP =======================================================#
# Importing the customized dataset
dataset = pd.read_csv('ann_temp_gistemp.csv')
year = dataset.iloc[:, 1:2].values
mean = dataset.iloc[:, 2:3].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
Lin_Reg = LinearRegression()
Lin_Reg.fit(year, mean)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
Poly_Reg = PolynomialFeatures(degree = 4)
Year_Poly = Poly_Reg.fit_transform(year)
Poly_Reg.fit(Year_Poly, mean)
Lin_Reg_2 = LinearRegression()
Lin_Reg_2.fit(Year_Poly, mean)

# Visualising the Linear Regression results
plt.scatter(year, mean, color = 'red')
plt.plot(year, Lin_Reg.predict(year), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs Mean Temperature (Linear Regression for GISTEMP)')
plt.xlabel('Years')
plt.ylabel('Mean Temperature')
plt.legend()
plt.show()

# Visualising the Polynomial Regression results
plt.scatter(year, mean, color = 'red')
plt.plot(year, Lin_Reg_2.predict(Poly_Reg.fit_transform(year)), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs Mean Temperature (Polynomial Regression for GISTEMP)')
plt.xlabel('Years')
plt.ylabel('Mean Temperature')
plt.legend()
plt.show()
```

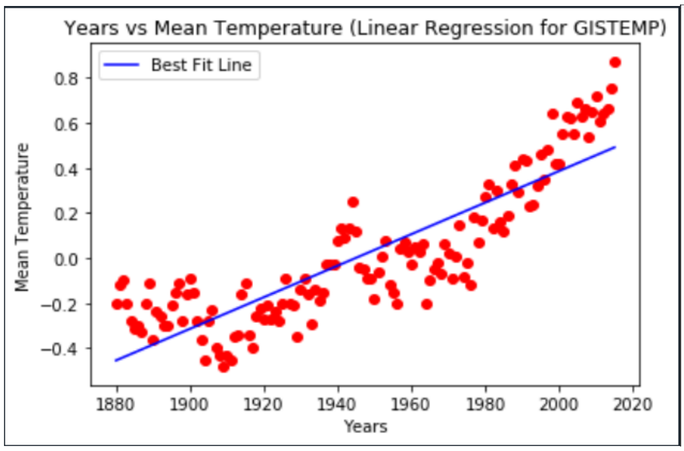


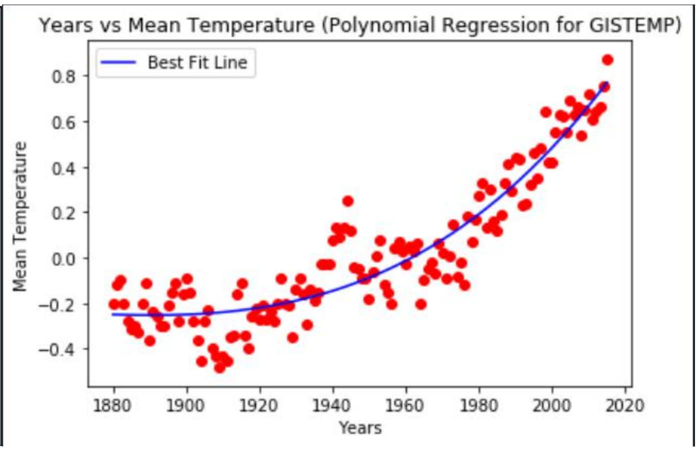
```python
# Predicting a new result with Linear Regression
Pred_Temp_Lin216 = Lin_Reg.predict([[2016]])
Pred_Temp_Lin217 = Lin_Reg.predict([[2017]])

# Predicting a new result with Polynomial Regression
Pred_Temp_Pol216 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2016]]))
Pred_Temp_Pol217 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2017]]))

#========================= Final Predictions ===========================================================#
print("Temperatures in 2016")
print("===================")
print("A/c to Linear Regression, Temperature of GCAG = ", Pred_Temp_Lin116 )
print("A/c to Polynomial Regression, Temperature of GCAG = ", Pred_Temp_Pol116 )
print("A/c to Linear Regression, Temperature of GISTEMP = ", Pred_Temp_Lin216)
print("A/c to Polynomial Regression,Temperature of GISTEMP =  ", Pred_Temp_Pol216)
print("\n")
print("Temperatures in 2017")
print("===================")
print("A/c to Linear Regression, Temperature of GCAG = ", Pred_Temp_Lin117 )
print("A/c to Polynomial Regression, Temperature of GCAG = ", Pred_Temp_Pol117 )
print("A/c to Linear Regression, Temperature of GISTEMP = ", Pred_Temp_Lin217)
print("A/c to Polynomial Regression,Temperature of GISTEMP =  ", Pred_Temp_Pol217)
```

## GRAPHS:

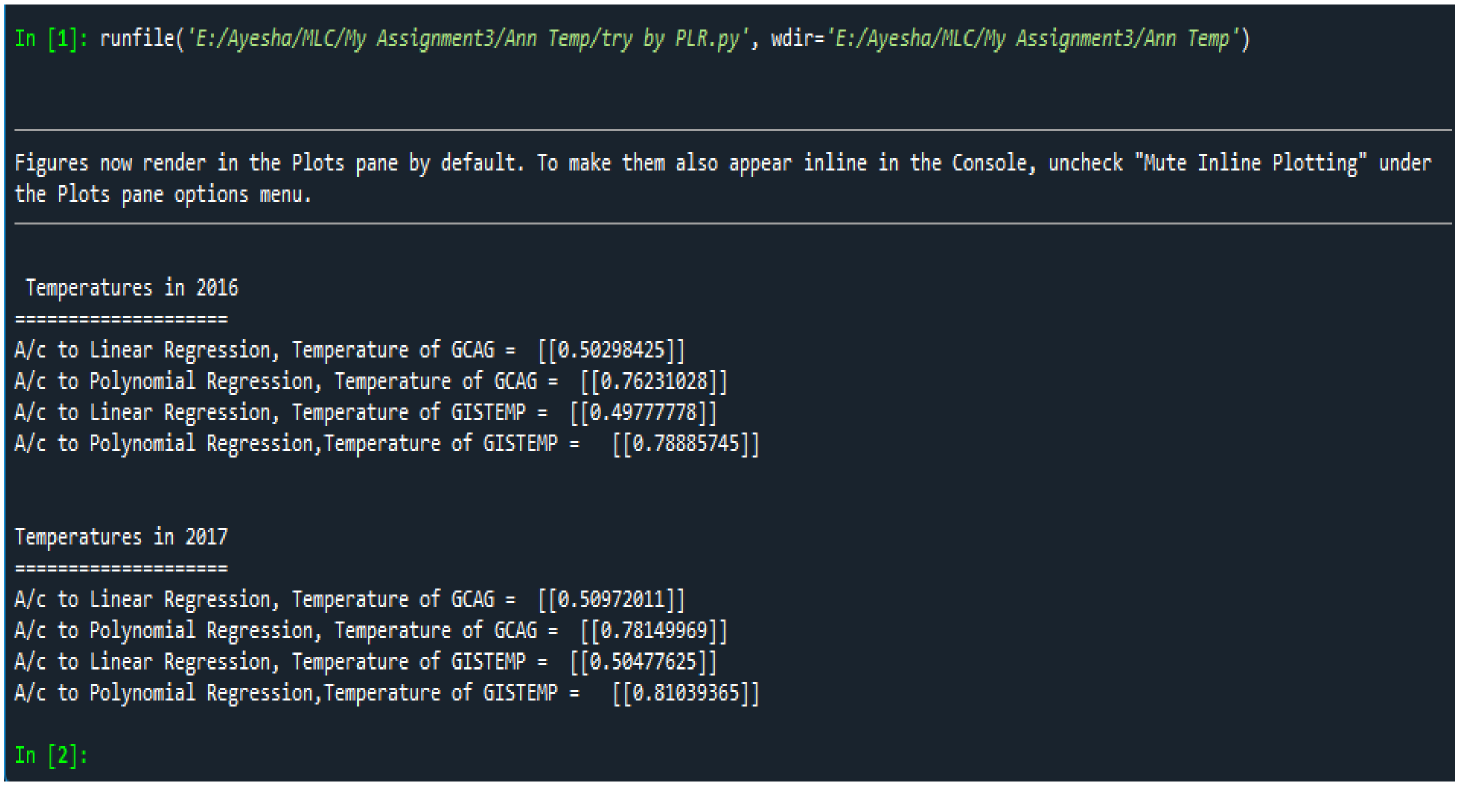






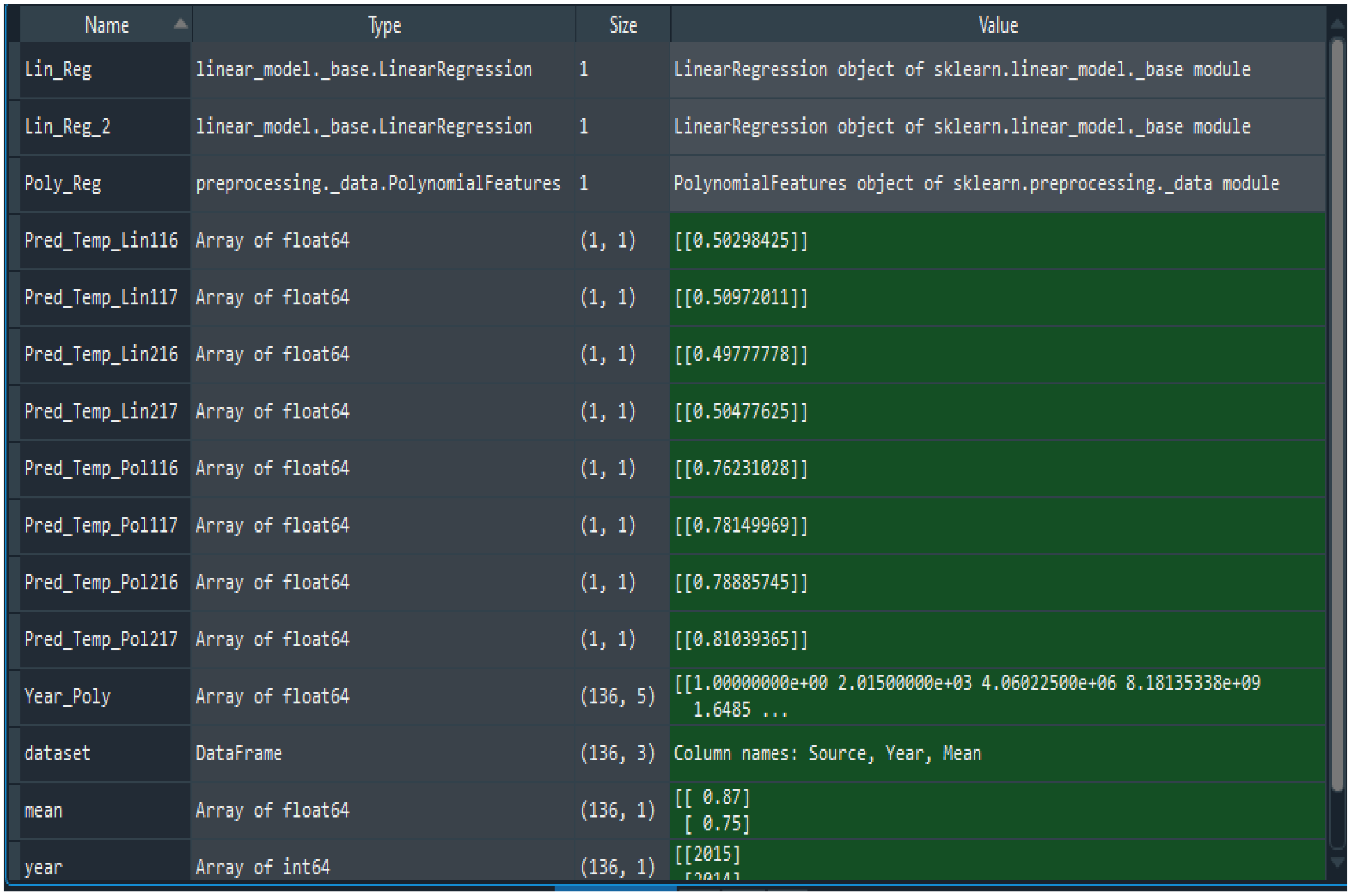


## CONSOLE:



```
In [1]: runfile('E:/Ayesha/MLC/My Assignment3/Ann Temp/try by PLR.py', wdir='E:/Ayesha/MLC/My Assignment3/Ann Temp')


Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under
the Plots pane options menu.


 Temperatures in 2016
====================
A/c to Linear Regression, Temperature of GCAG =  [[0.50298425]]
A/c to Polynomial Regression, Temperature of GCAG =  [[0.76231028]]
A/c to Linear Regression, Temperature of GISTEMP =  [[0.49777778]]
A/c to Polynomial Regression,Temperature of GISTEMP =   [[0.78885745]]


Temperatures in 2017
====================
A/c to Linear Regression, Temperature of GCAG =  [[0.50972011]]
A/c to Polynomial Regression, Temperature of GCAG =  [[0.78149969]]
A/c to Linear Regression, Temperature of GISTEMP =  [[0.50477625]]
A/c to Polynomial Regression,Temperature of GISTEMP =   [[0.81039365]]

In [2]:
```

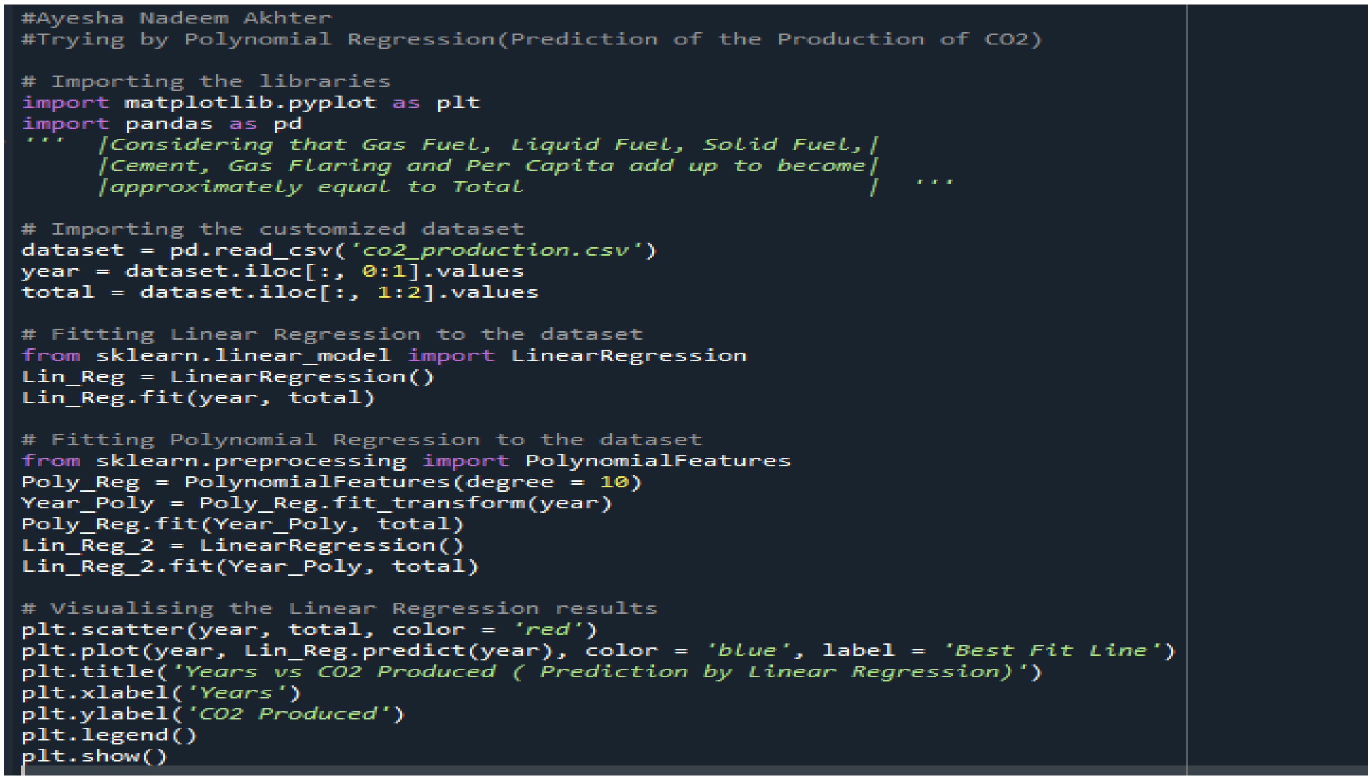
## VARIABLE EXPLORER:



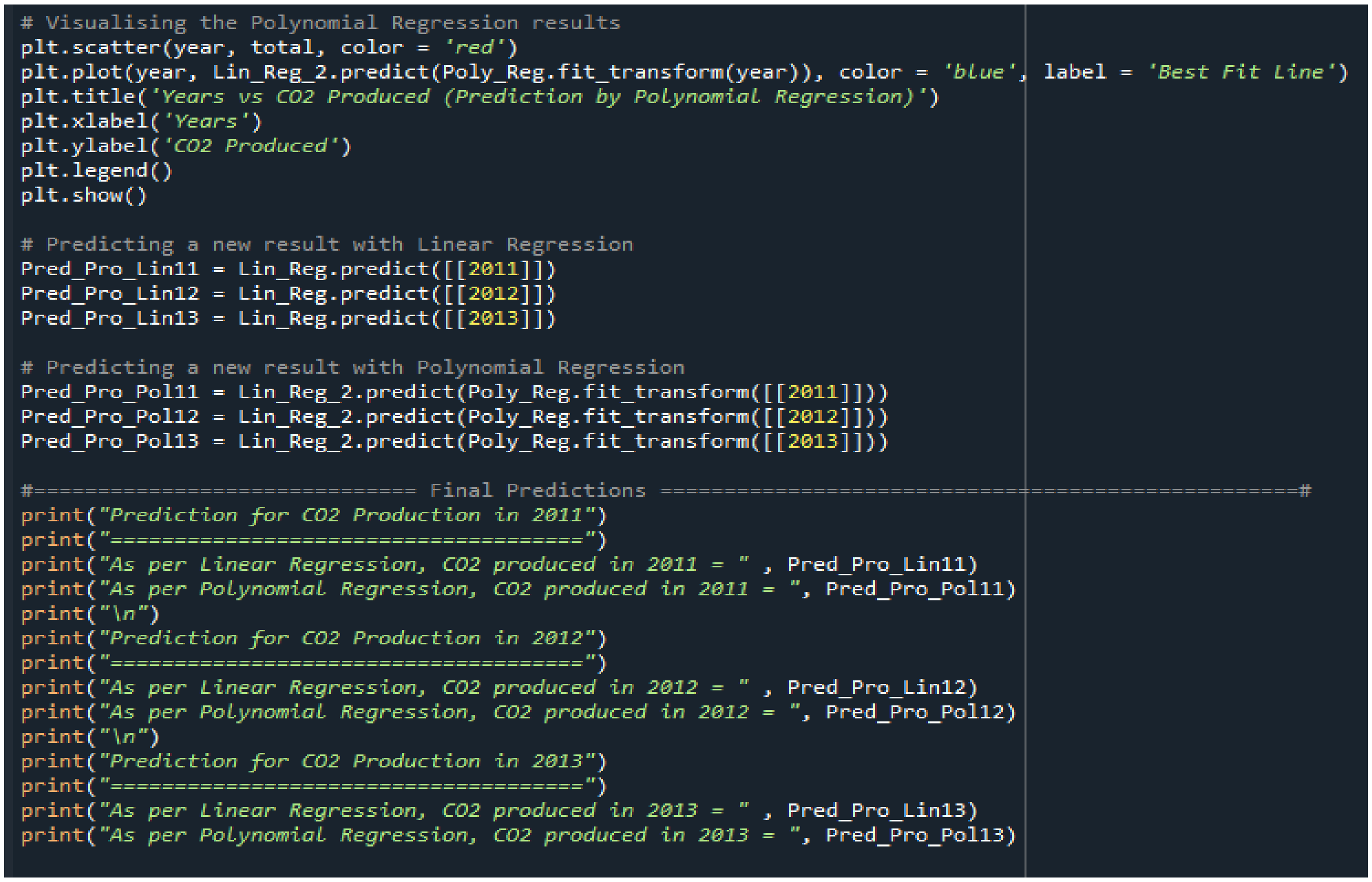
| Name | Type | Size | Value |
|---|---|---|---|
| Lin_Reg | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Lin_Reg_2 | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Poly_Reg | preprocessing._data.PolynomialFeatures | 1 | PolynomialFeatures object of sklearn.preprocessing._data module |
| Pred_Temp_Lin116 | Array of float64 | (1, 1) | [[0.50298425]] |
| Pred_Temp_Lin117 | Array of float64 | (1, 1) | [[0.50972011]] |
| Pred_Temp_Lin216 | Array of float64 | (1, 1) | [[0.49777778]] |
| Pred_Temp_Lin217 | Array of float64 | (1, 1) | [[0.50477625]] |
| Pred_Temp_Pol116 | Array of float64 | (1, 1) | [[0.76231028]] |
| Pred_Temp_Pol117 | Array of float64 | (1, 1) | [[0.78149969]] |
| Pred_Temp_Pol216 | Array of float64 | (1, 1) | [[0.78885745]] |
| Pred_Temp_Pol217 | Array of float64 | (1, 1) | [[0.81039365]] |
| Year_Poly | Array of float64 | (136, 5) | [[1.00000000e+00 2.01500000e+03 4.06022500e+06 8.18135338e+09 1.6485 ... |
| dataset | DataFrame | (136, 3) | Column names: Source, Year, Mean |
| mean | Array of float64 | (136, 1) | [[ 0.87] [ 0.75] |
| year | Array of int64 | (136, 1) | [[2015] |

## Q.3) 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.

**CODE:**

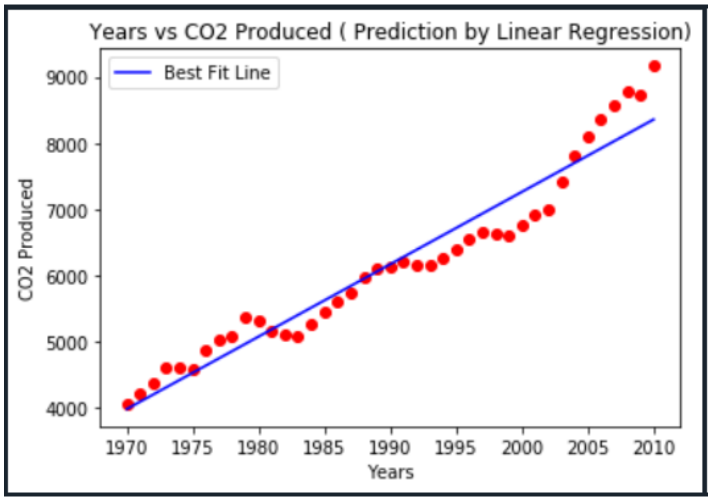


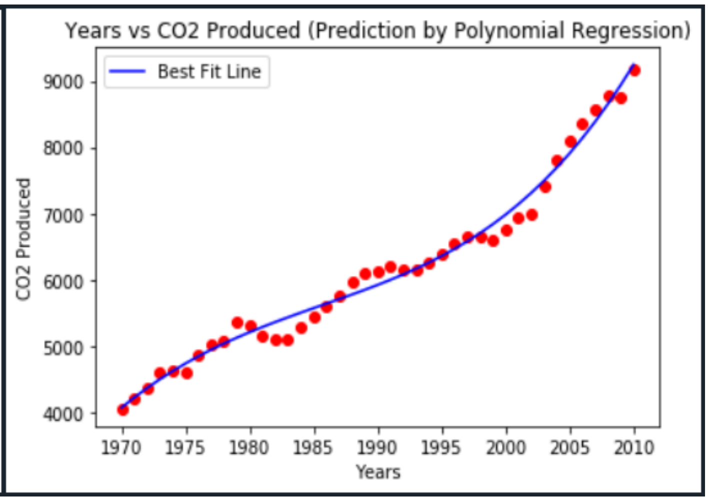
```python
#Ayesha Nadeem Akhter
#Trying by Polynomial Regression(Prediction of the Production of CO2)

# Importing the libraries
import matplotlib.pyplot as plt
import pandas as pd
'''   |Considering that Gas Fuel, Liquid Fuel, Solid Fuel,|
      |Cement, Gas Flaring and Per Capita add up to become|
      |approximately equal to Total                      |   '''

# Importing the customized dataset
dataset = pd.read_csv('co2_production.csv')
year = dataset.iloc[:, 0:1].values
total = dataset.iloc[:, 1:2].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
Lin_Reg = LinearRegression()
Lin_Reg.fit(year, total)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
Poly_Reg = PolynomialFeatures(degree = 10)
Year_Poly = Poly_Reg.fit_transform(year)
Poly_Reg.fit(Year_Poly, total)
Lin_Reg_2 = LinearRegression()
Lin_Reg_2.fit(Year_Poly, total)

# Visualising the Linear Regression results
plt.scatter(year, total, color = 'red')
plt.plot(year, Lin_Reg.predict(year), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs CO2 Produced ( Prediction by Linear Regression)')
plt.xlabel('Years')
plt.ylabel('CO2 Produced')
plt.legend()
plt.show()
```



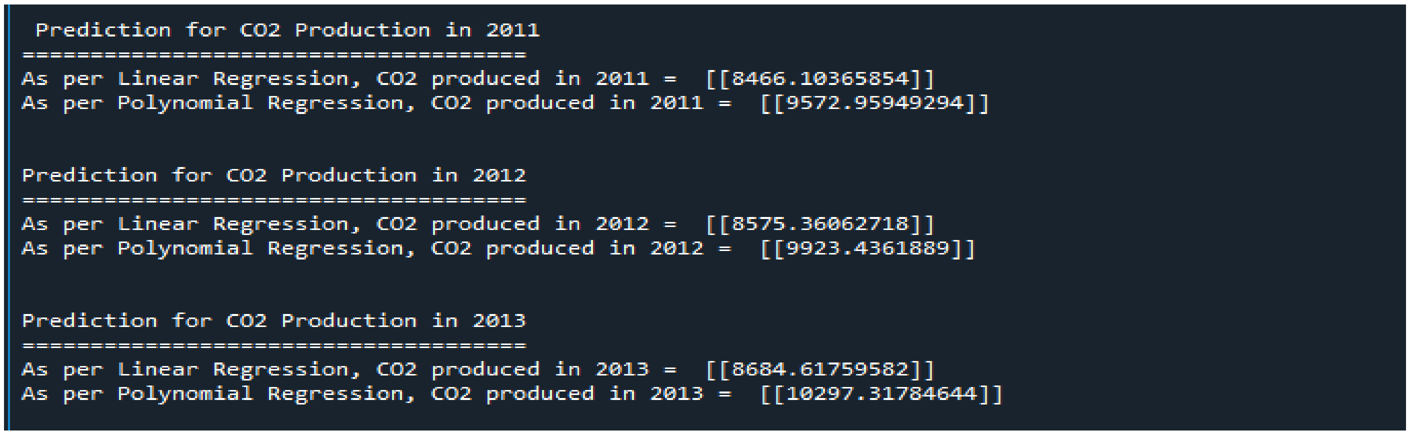
```python
# Visualising the Polynomial Regression results
plt.scatter(year, total, color = 'red')
plt.plot(year, Lin_Reg_2.predict(Poly_Reg.fit_transform(year)), color = 'blue', label = 'Best Fit Line')
plt.title('Years vs CO2 Produced (Prediction by Polynomial Regression)')
plt.xlabel('Years')
plt.ylabel('CO2 Produced')
plt.legend()
plt.show()

# Predicting a new result with Linear Regression
Pred_Pro_Lin11 = Lin_Reg.predict([[2011]])
Pred_Pro_Lin12 = Lin_Reg.predict([[2012]])
Pred_Pro_Lin13 = Lin_Reg.predict([[2013]])

# Predicting a new result with Polynomial Regression
Pred_Pro_Pol11 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2011]]))
Pred_Pro_Pol12 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2012]]))
Pred_Pro_Pol13 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2013]]))

#============================== Final Predictions ==================================================#
print("Prediction for CO2 Production in 2011")
print("====================================")
print("As per Linear Regression, CO2 produced in 2011 = " , Pred_Pro_Lin11)
print("As per Polynomial Regression, CO2 produced in 2011 = ", Pred_Pro_Pol11)
print("\n")
print("Prediction for CO2 Production in 2012")
print("====================================")
print("As per Linear Regression, CO2 produced in 2012 = " , Pred_Pro_Lin12)
print("As per Polynomial Regression, CO2 produced in 2012 = ", Pred_Pro_Pol12)
print("\n")
print("Prediction for CO2 Production in 2013")
print("====================================")
print("As per Linear Regression, CO2 produced in 2013 = " , Pred_Pro_Lin13)
print("As per Polynomial Regression, CO2 produced in 2013 = ", Pred_Pro_Pol13)
```

## GRAPHS:



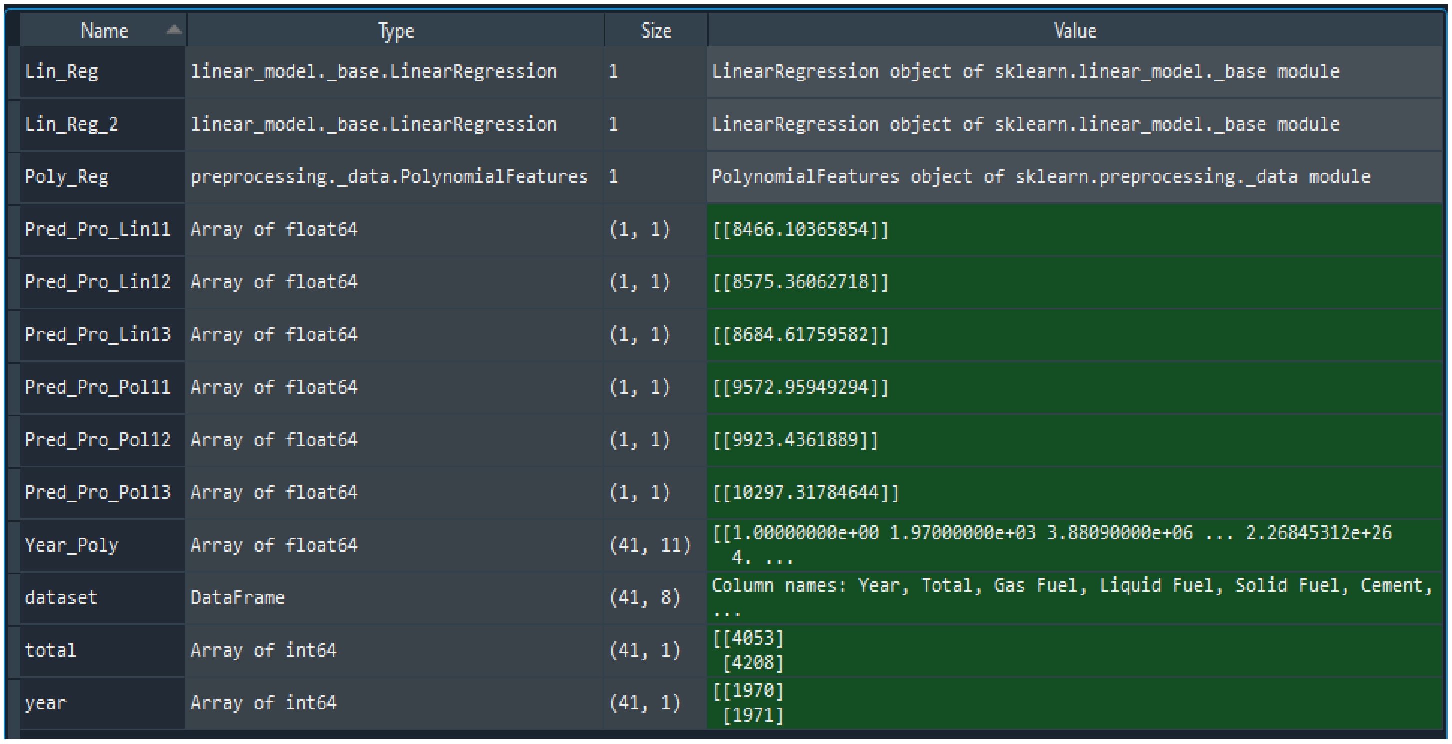


## CONSOLE:



```
 Prediction for CO2 Production in 2011
=====================================
As per Linear Regression, CO2 produced in 2011 =  [[8466.10365854]]
As per Polynomial Regression, CO2 produced in 2011 =  [[9572.95949294]]


Prediction for CO2 Production in 2012
=====================================
As per Linear Regression, CO2 produced in 2012 =  [[8575.36062718]]
As per Polynomial Regression, CO2 produced in 2012 =  [[9923.4361889]]


Prediction for CO2 Production in 2013
=====================================
As per Linear Regression, CO2 produced in 2013 =  [[8684.61759582]]
As per Polynomial Regression, CO2 produced in 2013 =  [[10297.31784644]]
```

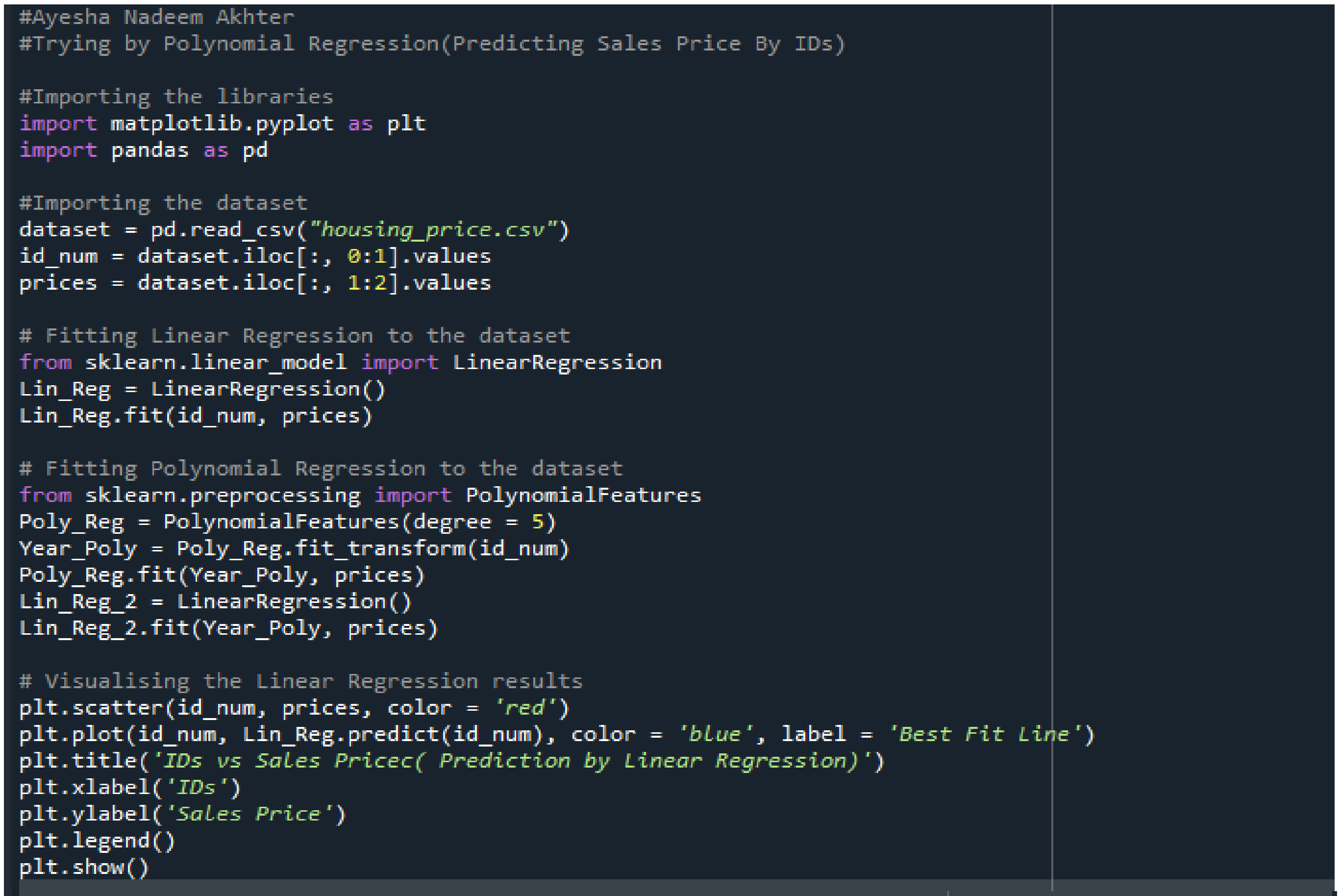
## VARIABLE EXPLORER:



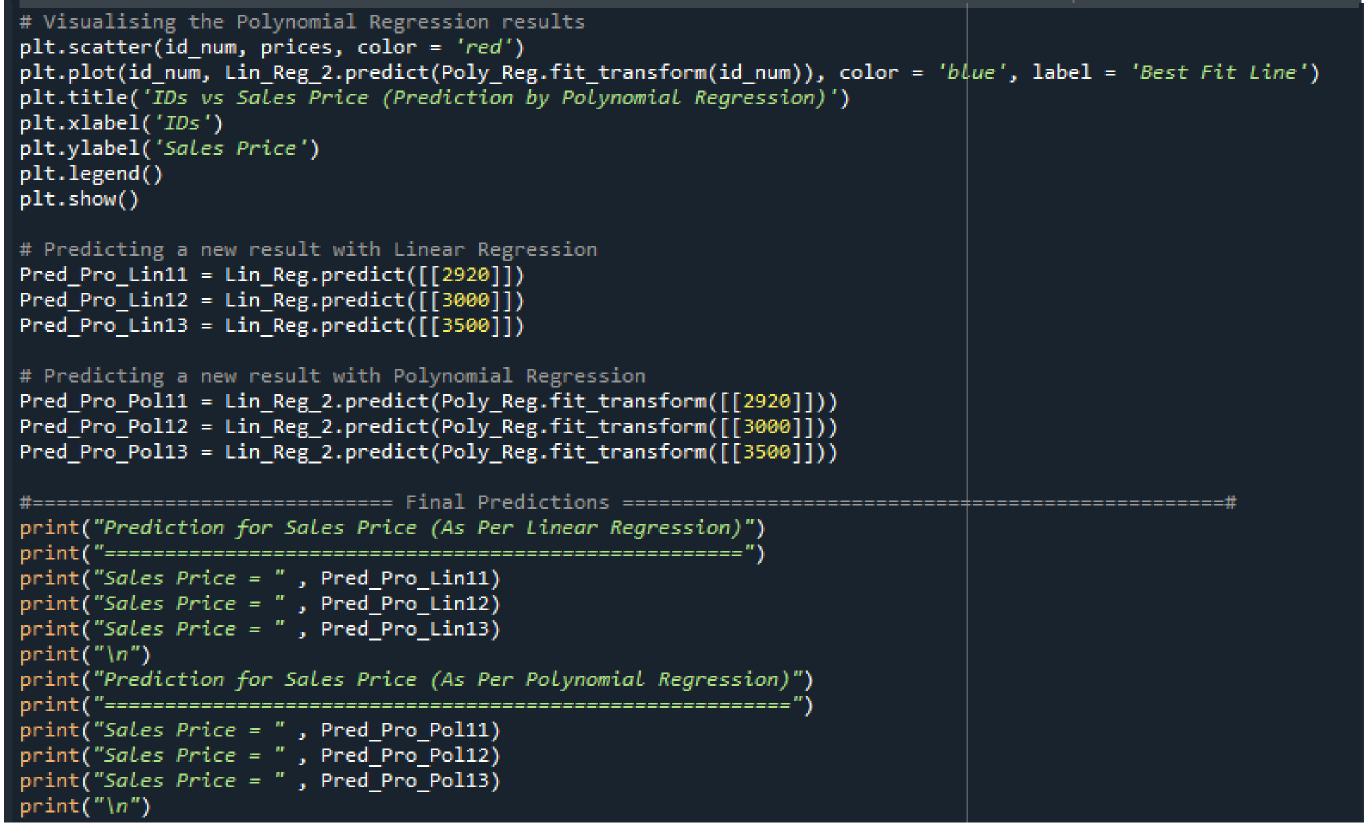
| Name | Type | Size | Value |
|---|---|---|---|
| Lin_Reg | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Lin_Reg_2 | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Poly_Reg | preprocessing._data.PolynomialFeatures | 1 | PolynomialFeatures object of sklearn.preprocessing._data module |
| Pred_Pro_Lin11 | Array of float64 | (1, 1) | [[8466.10365854]] |
| Pred_Pro_Lin12 | Array of float64 | (1, 1) | [[8575.36062718]] |
| Pred_Pro_Lin13 | Array of float64 | (1, 1) | [[8684.61759582]] |
| Pred_Pro_Pol11 | Array of float64 | (1, 1) | [[9572.95949294]] |
| Pred_Pro_Pol12 | Array of float64 | (1, 1) | [[9923.4361889]] |
| Pred_Pro_Pol13 | Array of float64 | (1, 1) | [[10297.31784644]] |
| Year_Poly | Array of float64 | (41, 11) | [[1.00000000e+00 1.97000000e+03 3.88090000e+06 ... 2.26845312e+26 4. ... |
| dataset | DataFrame | (41, 8) | Column names: Year, Total, Gas Fuel, Liquid Fuel, Solid Fuel, Cement, ... |
| total | Array of int64 | (41, 1) | [[4053] [4208] |
| year | Array of int64 | (41, 1) | [[1970] [1971] |

## Q.4) HOUSING PRICE ACCORDING TO THE ID IS ASSIGNED TO EVERY-HOUSE. PERFORM FUTURE ANALYSIS WHERE WHEN ID IS INSERTED THE HOUSING PRICE IS DISPLAYED.

**CODE:**

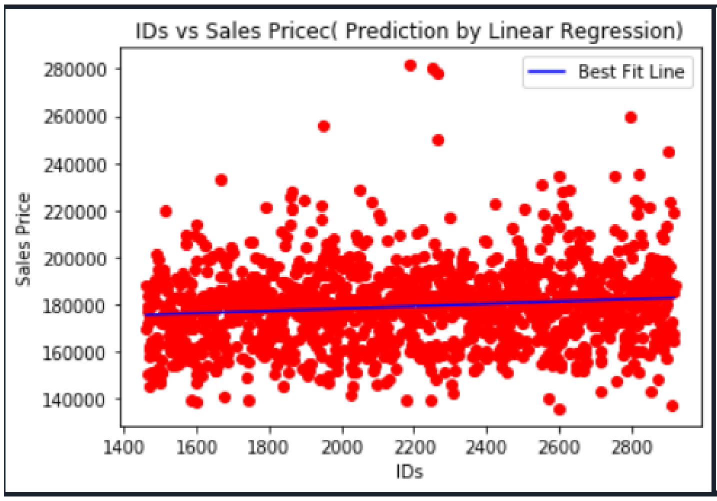


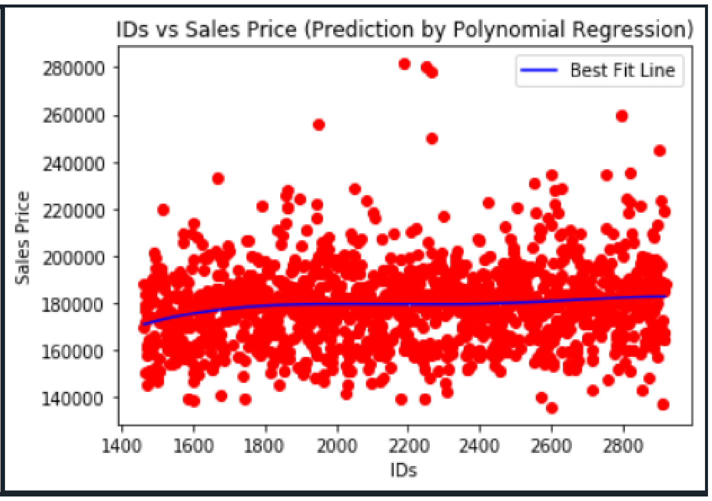
```python
#Ayesha Nadeem Akhter
#Trying by Polynomial Regression(Predicting Sales Price By IDs)

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

#Importing the dataset
dataset = pd.read_csv("housing_price.csv")
id_num = dataset.iloc[:, 0:1].values
prices = dataset.iloc[:, 1:2].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
Lin_Reg = LinearRegression()
Lin_Reg.fit(id_num, prices)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
Poly_Reg = PolynomialFeatures(degree = 5)
Year_Poly = Poly_Reg.fit_transform(id_num)
Poly_Reg.fit(Year_Poly, prices)
Lin_Reg_2 = LinearRegression()
Lin_Reg_2.fit(Year_Poly, prices)

# Visualising the Linear Regression results
plt.scatter(id_num, prices, color = 'red')
plt.plot(id_num, Lin_Reg.predict(id_num), color = 'blue', label = 'Best Fit Line')
plt.title('IDs vs Sales Pricec( Prediction by Linear Regression)')
plt.xlabel('IDs')
plt.ylabel('Sales Price')
plt.legend()
plt.show()
```



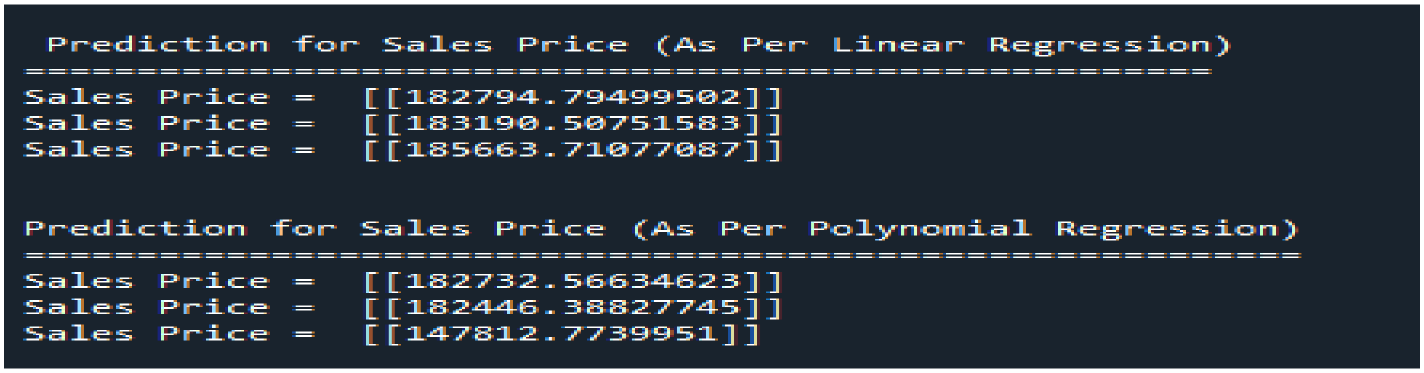
```python
# Visualising the Polynomial Regression results
plt.scatter(id_num, prices, color = 'red')
plt.plot(id_num, Lin_Reg_2.predict(Poly_Reg.fit_transform(id_num)), color = 'blue', label = 'Best Fit Line')
plt.title('IDs vs Sales Price (Prediction by Polynomial Regression)')
plt.xlabel('IDs')
plt.ylabel('Sales Price')
plt.legend()
plt.show()

# Predicting a new result with Linear Regression
Pred_Pro_Lin11 = Lin_Reg.predict([[2920]])
Pred_Pro_Lin12 = Lin_Reg.predict([[3000]])
Pred_Pro_Lin13 = Lin_Reg.predict([[3500]])

# Predicting a new result with Polynomial Regression
Pred_Pro_Pol11 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[2920]]))
Pred_Pro_Pol12 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[3000]]))
Pred_Pro_Pol13 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[3500]]))

#============================== Final Predictions ==================================================#
print("Prediction for Sales Price (As Per Linear Regression)")
print("================================================")
print("Sales Price = " , Pred_Pro_Lin11)
print("Sales Price = " , Pred_Pro_Lin12)
print("Sales Price = " , Pred_Pro_Lin13)
print("\n")
print("Prediction for Sales Price (As Per Polynomial Regression)")
print("====================================================")
print("Sales Price = " , Pred_Pro_Pol11)
print("Sales Price = " , Pred_Pro_Pol12)
print("Sales Price = " , Pred_Pro_Pol13)
print("\n")
```

## GRAPH:



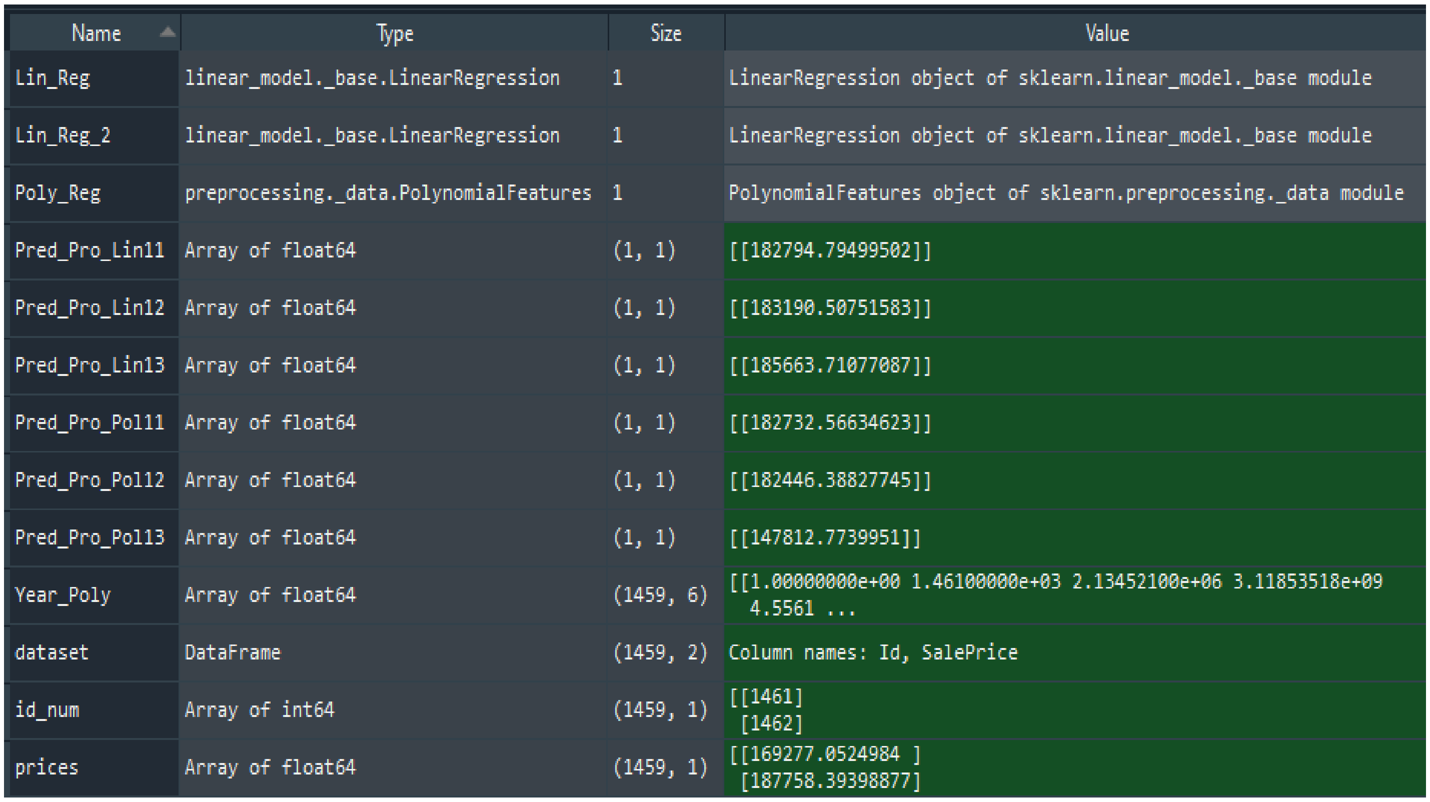


## CONSOLE:



```
 Prediction for Sales Price (As Per Linear Regression)
=====================================================
Sales Price =  [[182794.79499502]]
Sales Price =  [[183190.50751583]]
Sales Price =  [[185663.71077087]]

Prediction for Sales Price (As Per Polynomial Regression)
=========================================================
Sales Price =  [[182732.56634623]]
Sales Price =  [[182446.38827745]]
Sales Price =  [[147812.7739951]]
```

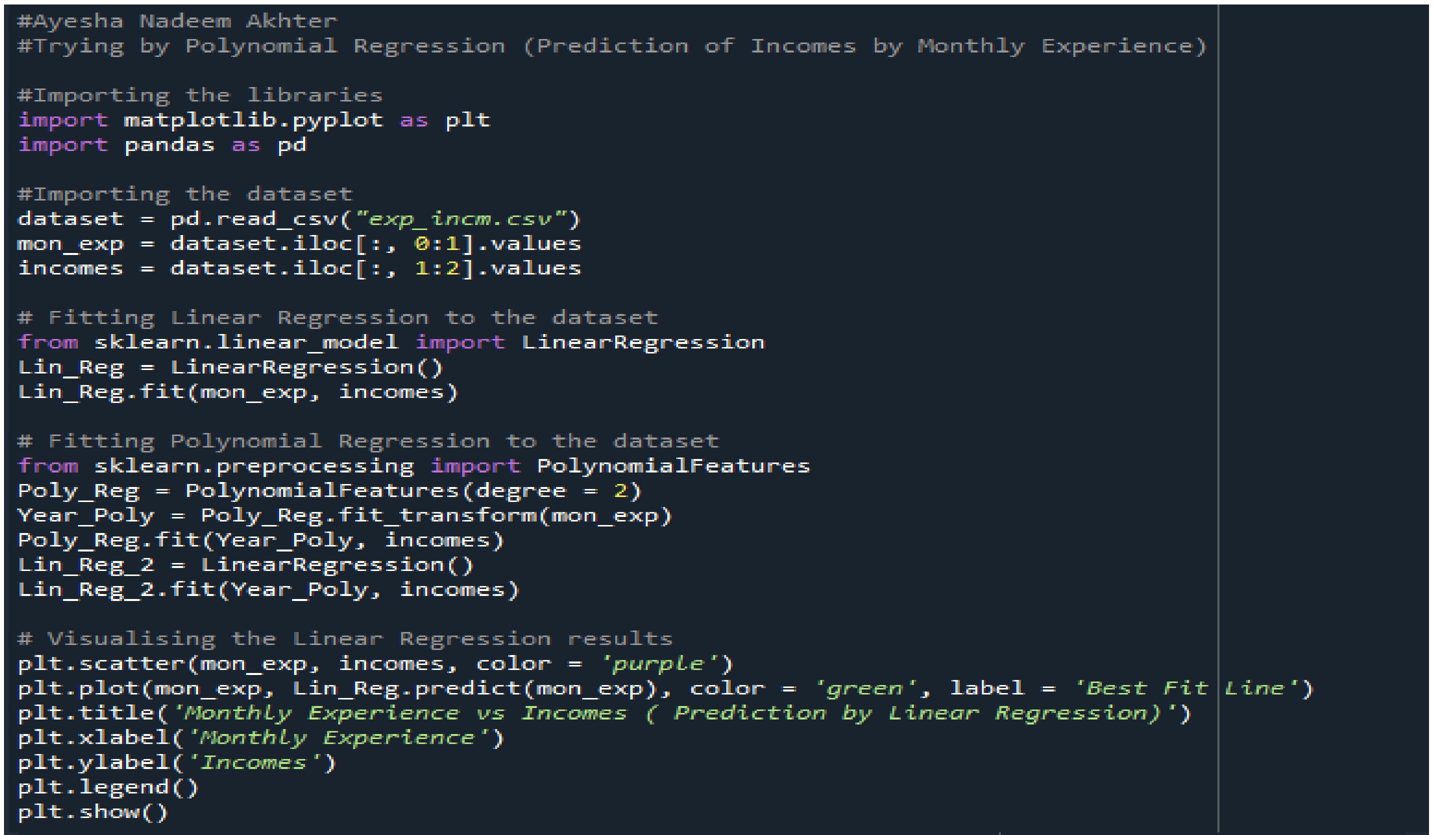
## VARIABLE EXPLORER:



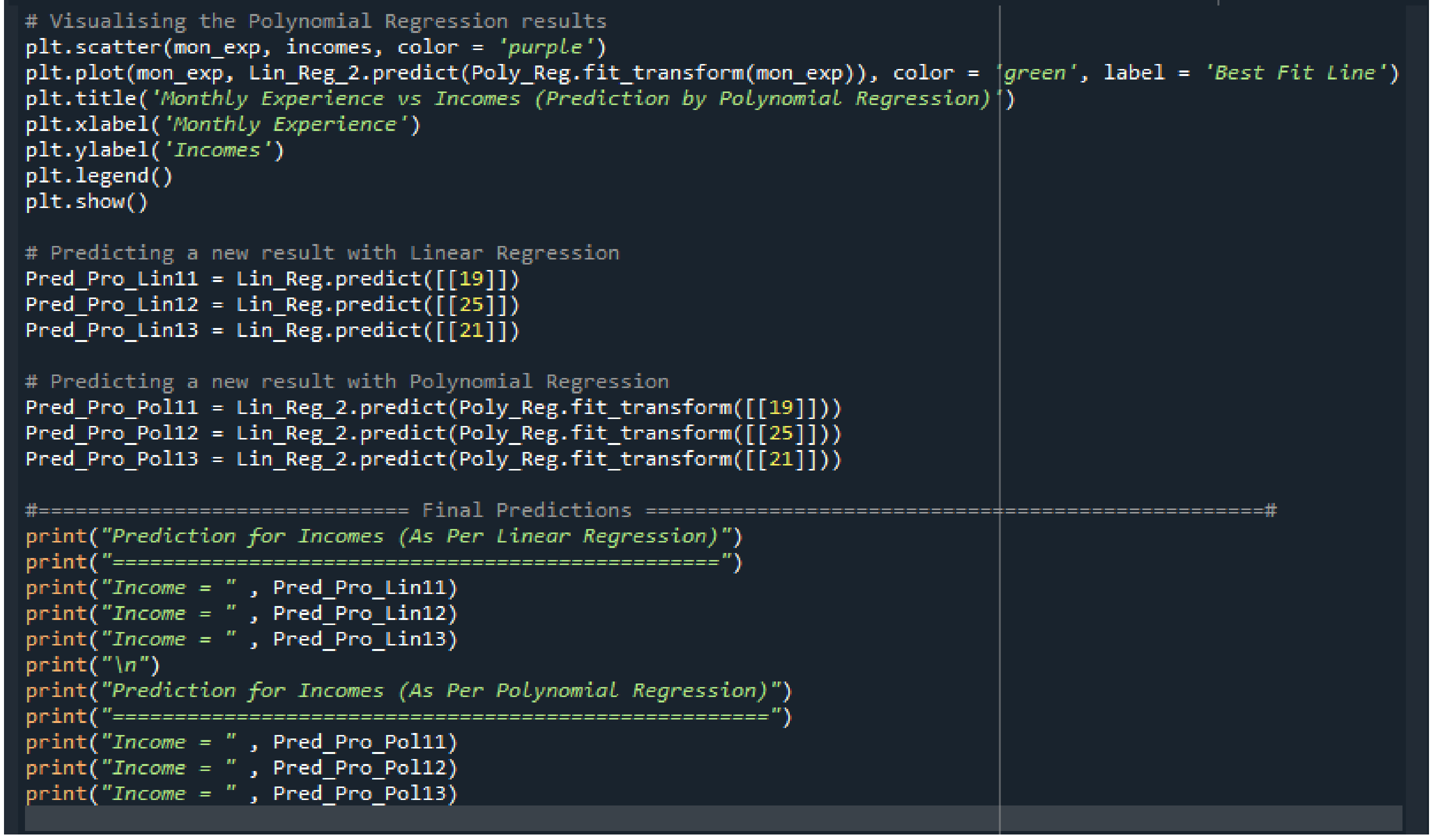
| Name | Type | Size | Value |
|---|---|---|---|
| Lin_Reg | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Lin_Reg_2 | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Poly_Reg | preprocessing._data.PolynomialFeatures | 1 | PolynomialFeatures object of sklearn.preprocessing._data module |
| Pred_Pro_Lin11 | Array of float64 | (1, 1) | [[182794.79499502]] |
| Pred_Pro_Lin12 | Array of float64 | (1, 1) | [[183190.50751583]] |
| Pred_Pro_Lin13 | Array of float64 | (1, 1) | [[185663.71077087]] |
| Pred_Pro_Pol11 | Array of float64 | (1, 1) | [[182732.56634623]] |
| Pred_Pro_Pol12 | Array of float64 | (1, 1) | [[182446.38827745]] |
| Pred_Pro_Pol13 | Array of float64 | (1, 1) | [[147812.7739951]] |
| Year_Poly | Array of float64 | (1459, 6) | [[1.00000000e+00 1.46100000e+03 2.13452100e+06 3.11853518e+09 4.5561 ... |
| dataset | DataFrame | (1459, 2) | Column names: Id, SalePrice |
| id_num | Array of int64 | (1459, 1) | [[1461] [1462] |
| prices | Array of float64 | (1459, 1) | [[169277.0524984 ] [187758.39398877] |

## **Q.5)** DATA OF MONTHLY EXPERIENCE AND INCOME DISTRIBUTION OF DIFFERENT EMPLOYS IS GIVEN. PERFORM REGRESSION.

**CODE:**

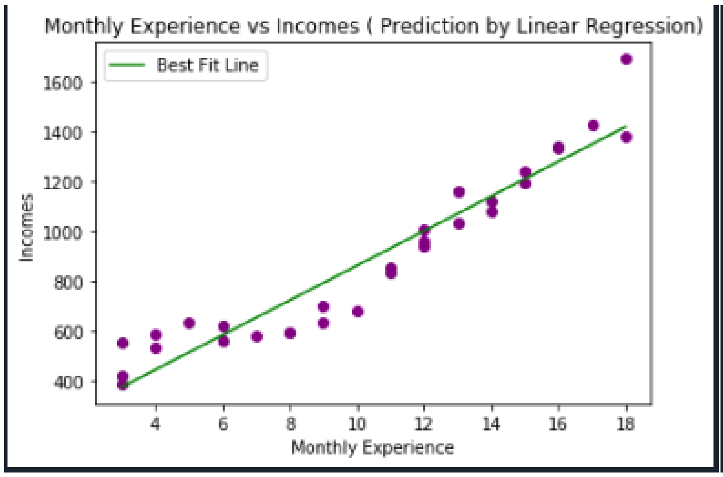


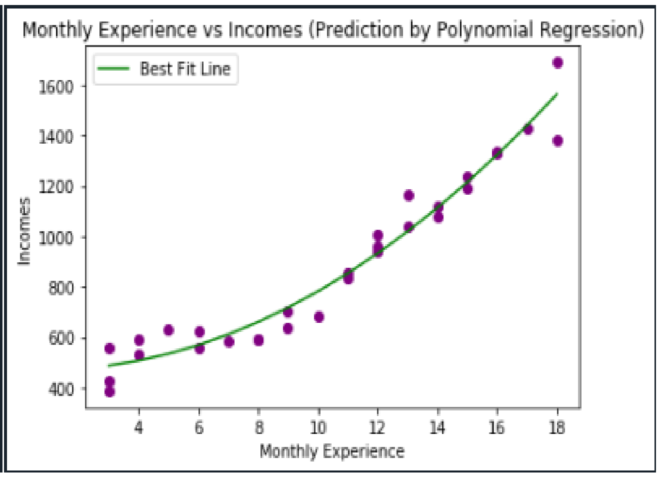
```
#Ayesha Nadeem Akhter
#Trying by Polynomial Regression (Prediction of Incomes by Monthly Experience)

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

#Importing the dataset
dataset = pd.read_csv("exp_incm.csv")
mon_exp = dataset.iloc[:, 0:1].values
incomes = dataset.iloc[:, 1:2].values

# Fitting Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
Lin_Reg = LinearRegression()
Lin_Reg.fit(mon_exp, incomes)

# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
Poly_Reg = PolynomialFeatures(degree = 2)
Year_Poly = Poly_Reg.fit_transform(mon_exp)
Poly_Reg.fit(Year_Poly, incomes)
Lin_Reg_2 = LinearRegression()
Lin_Reg_2.fit(Year_Poly, incomes)

# Visualising the Linear Regression results
plt.scatter(mon_exp, incomes, color = 'purple')
plt.plot(mon_exp, Lin_Reg.predict(mon_exp), color = 'green', label = 'Best Fit Line')
plt.title('Monthly Experience vs Incomes ( Prediction by Linear Regression)')
plt.xlabel('Monthly Experience')
plt.ylabel('Incomes')
plt.legend()
plt.show()
```



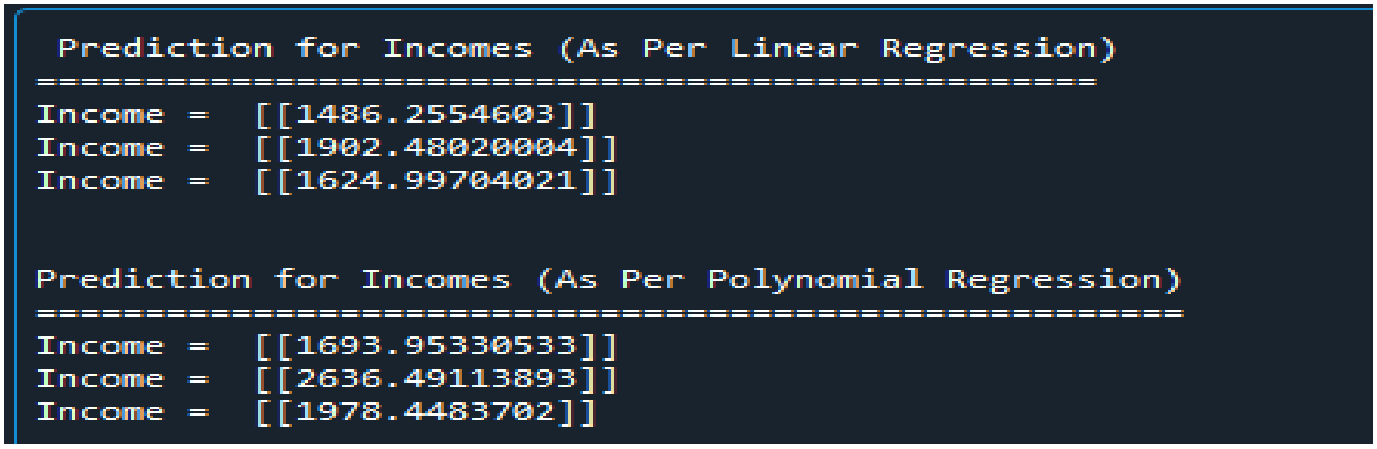
```
# Visualising the Polynomial Regression results
plt.scatter(mon_exp, incomes, color = 'purple')
plt.plot(mon_exp, Lin_Reg_2.predict(Poly_Reg.fit_transform(mon_exp)), color = 'green', label = 'Best Fit Line')
plt.title('Monthly Experience vs Incomes (Prediction by Polynomial Regression)')
plt.xlabel('Monthly Experience')
plt.ylabel('Incomes')
plt.legend()
plt.show()

# Predicting a new result with Linear Regression
Pred_Pro_Lin11 = Lin_Reg.predict([[19]])
Pred_Pro_Lin12 = Lin_Reg.predict([[25]])
Pred_Pro_Lin13 = Lin_Reg.predict([[21]])

# Predicting a new result with Polynomial Regression
Pred_Pro_Pol11 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[19]]))
Pred_Pro_Pol12 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[25]]))
Pred_Pro_Pol13 = Lin_Reg_2.predict(Poly_Reg.fit_transform([[21]]))

#============================== Final Predictions ==================================================#
print("Prediction for Incomes (As Per Linear Regression)")
print("================================================")
print("Income = " , Pred_Pro_Lin11)
print("Income = " , Pred_Pro_Lin12)
print("Income = " , Pred_Pro_Lin13)
print("\n")
print("Prediction for Incomes (As Per Polynomial Regression)")
print("====================================================")
print("Income = " , Pred_Pro_Pol11)
print("Income = " , Pred_Pro_Pol12)
print("Income = " , Pred_Pro_Pol13)
```

## GRAPHS:



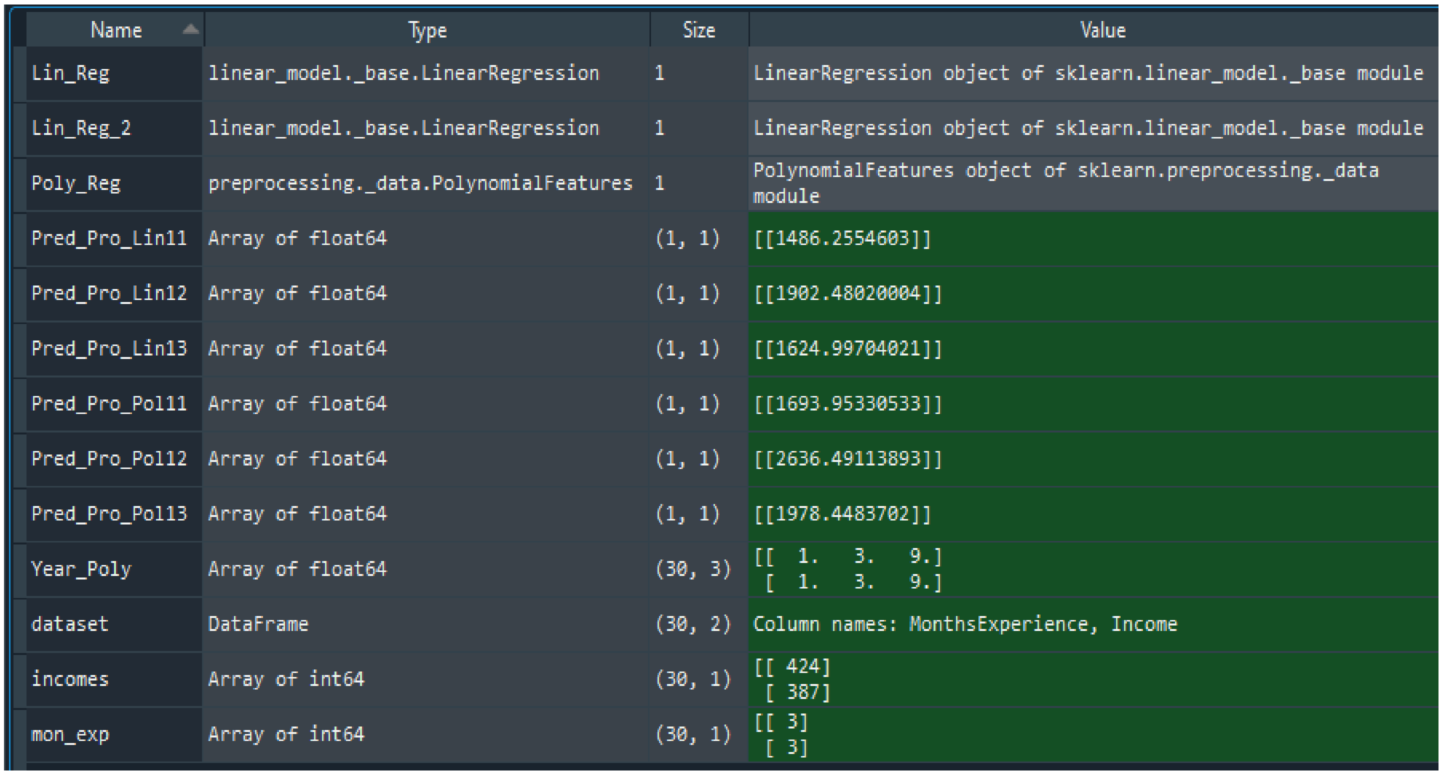


## CONSOLE:



```
 Prediction for Incomes (As Per Linear Regression)
=================================================
Income =  [[1486.2554603]]
Income =  [[1902.48020004]]
Income =  [[1624.99704021]]


Prediction for Incomes (As Per Polynomial Regression)
=====================================================
Income =  [[1693.95330533]]
Income =  [[2636.49113893]]
Income =  [[1978.4483702]]
```

## VARIABLE EXPLORER:



| Name | Type | Size | Value |
|---|---|---|---|
| Lin_Reg | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Lin_Reg_2 | linear_model._base.LinearRegression | 1 | LinearRegression object of sklearn.linear_model._base module |
| Poly_Reg | preprocessing._data.PolynomialFeatures | 1 | PolynomialFeatures object of sklearn.preprocessing._data module |
| Pred_Pro_Lin11 | Array of float64 | (1, 1) | [[1486.2554603]] |
| Pred_Pro_Lin12 | Array of float64 | (1, 1) | [[1902.48020004]] |
| Pred_Pro_Lin13 | Array of float64 | (1, 1) | [[1624.99704021]] |
| Pred_Pro_Pol11 | Array of float64 | (1, 1) | [[1693.95330533]] |
| Pred_Pro_Pol12 | Array of float64 | (1, 1) | [[2636.49113893]] |
| Pred_Pro_Pol13 | Array of float64 | (1, 1) | [[1978.4483702]] |
| Year_Poly | Array of float64 | (30, 3) | [[ 1. 3. 9.] [ 1. 3. 9.] |
| dataset | DataFrame | (30, 2) | Column names: MonthsExperience, Income |
| incomes | Array of int64 | (30, 1) | [[ 424] [ 387] |
| mon_exp | Array of int64 | (30, 1) | [[ 3] [ 3] |

## JUSTIFICATIONS FOR THE CHOICES MADE

### QUESTION # 1 (DECISION TREE REGRESSION)

The problem was regarding classification of the state that would give the best profit in future. Therefore, Decision Tree Regression was, according to me, the best approach for such a prediction.

### QUESTION # 2(POLYNOMIAL REGRESSION)

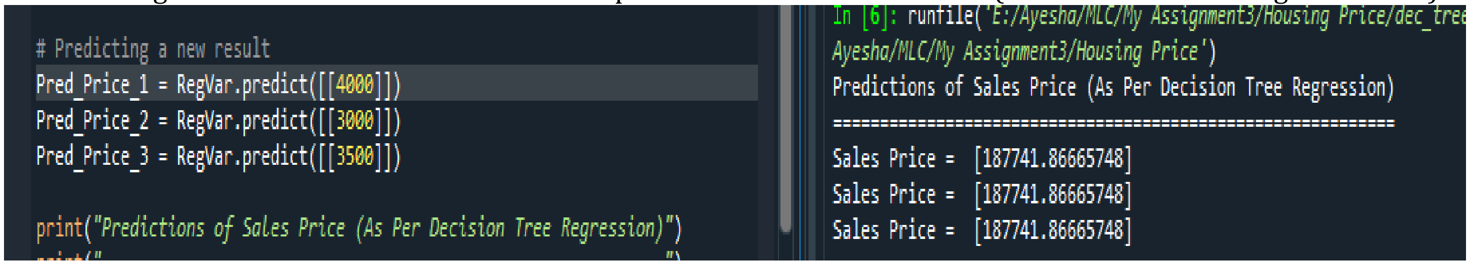
The problem was regarding the prediction of temperatures of two countries in 2016 and 2017 (based on the old data provided). The Best Fit Line given by Polynomial Regression appeared to be more accurate than the line given by Linear Regression.

### QUESTION # 3(POLYNOMIAL REGRESSION)

The problem was regarding the prediction of global production of CO2 of a place in 2011, 2012 and 2013 (based on the old data provided). The *Best Fit Line* given by Polynomial Regression appeared to be more accurate than that of Linear Regression.

### QUESTION # 4(POLYNOMIAL REGRESISON)

The problem was regarding the prediction of housing price based on the IDs provided, Here, the accuracy of the line given by Polynomial Regression was better than that of Linear Regression. Upon applying Decision Tree Regression, a line with an even better accuracy was obtained. However, it was noticed that the predicted values for the Housing Price remained the same irrespective of the *ID* entered (as shown in the figure below).



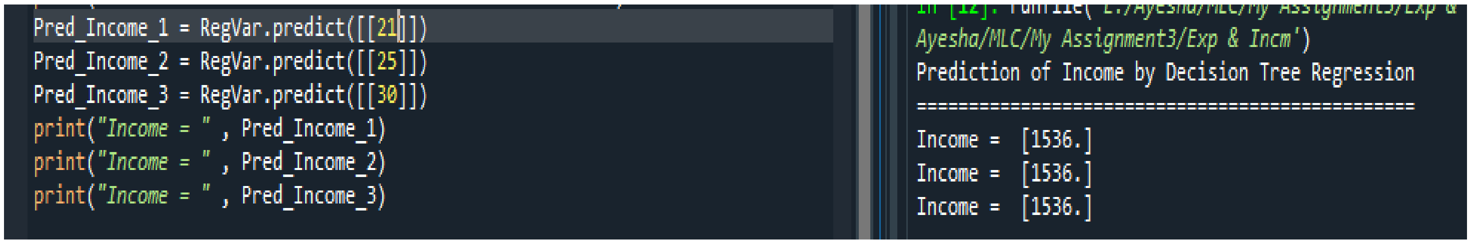
```
# Predicting a new result
Pred_Price_1 = RegVar.predict([[4000]])
Pred_Price_2 = RegVar.predict([[3000]])
Pred_Price_3 = RegVar.predict([[3500]])

print("Predictions of Sales Price (As Per Decision Tree Regression)")
```

```
In [6]: runfile('E:/Ayesha/MLC/My Assignment3/Housing Price/dec_tree
Ayesha/MLC/My Assignment3/Housing Price')
Predictions of Sales Price (As Per Decision Tree Regression)
=========================================================
Sales Price =  [187741.86665748]
Sales Price =  [187741.86665748]
Sales Price =  [187741.86665748]
```

Therefore, Polynomial Regression was chosen as the best solution for this problem.

### QUESTION # 5(POLYNOMIAL REGRESSION)

The problem was regarding the prediction of income based on monthly experience (based on the old data provided). Here, the accuracy of the line given by Polynomial Regression was found to be better than that of Linear Regression. And, upon the application Decision Tree Regression, the line that appeared to be best fit (to the scattered data points) was obtained. But, similar predicted values were obtained for income for different monthly experiences which is why Polynomial Regression was again chosen as the best solution for this problem. (Snapshot has been given below.)



```
Pred_Income_1 = RegVar.predict([[21]])
Pred_Income_2 = RegVar.predict([[25]])
Pred_Income_3 = RegVar.predict([[30]])
print("Income = " , Pred_Income_1)
print("Income = " , Pred_Income_2)
print("Income = " , Pred_Income_3)
```

```
In [12]: runfile('E:/Ayesha/MLC/My Assignment3/Exp &
Ayesha/MLC/My Assignment3/Exp & Incm')
Prediction of Income by Decision Tree Regression
===============================================
Income =  [1536.]
Income =  [1536.]
Income =  [1536.]
```