

Christopher Teelucksingh 89496: AIML Assignment 2: [80 marks]

Regression: Linear, Polynomial, Multiple, Scale, Train/Test

```
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [90,80,87,88,90,86,89,87,94,78,77,85,86]
```

Figure 1: the x-axis represents a car's age, and the y-axis represents its speed.

Using the two vectors in Figure 1 above, write code in Python to:

1. Create a Scatter Plot with the values of x and y [4 marks]

```
x=[5,7,8,7,2,17,2,9,4,11,12,9,6]
y=[90,80,87,88,90,86,89,87,94,78,77,85,86]
plt.scatter(x,y)
plt.xlabel('Age of car')
plt.ylabel('Speed of car')
plt.show()
```

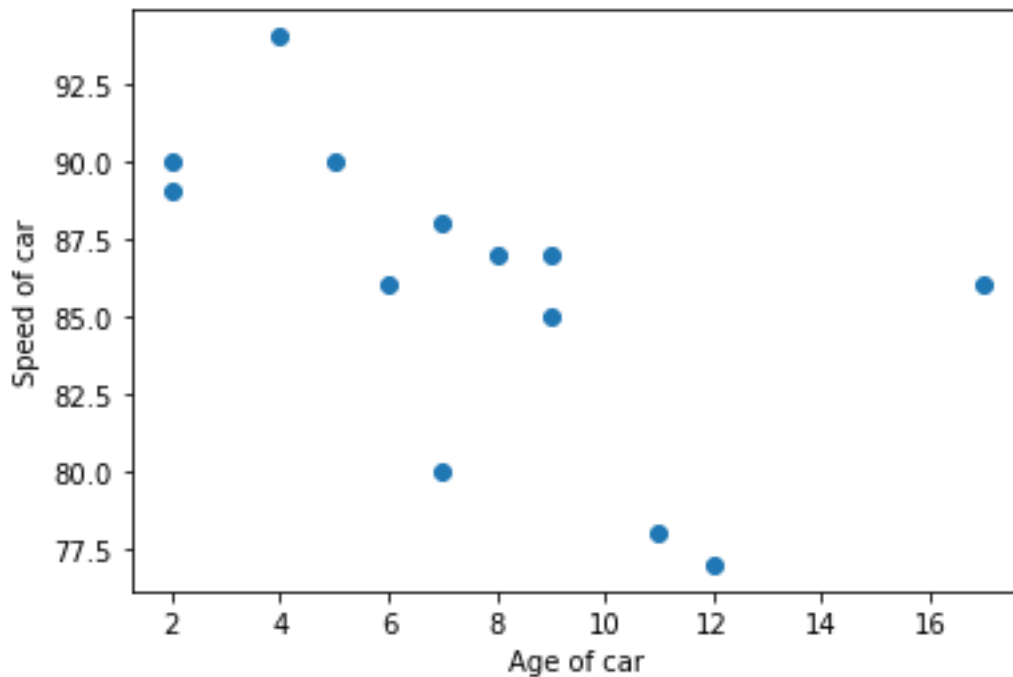


Figure 1 scatter plot for question 1

2. Fit the data to a Linear Regression line, and plot it on the scatter plot [4 marks]

```
slope,intercept,r,p,std_err = stats.linregress(x,y)
Reg_LineY = np.array(x)*slope + intercept
plt.scatter(x,y)
plt.plot(x,Reg_LineY)
plt.xlabel('Age of car')
plt.ylabel('Speed of car')
plt.grid()
plt.show()
```

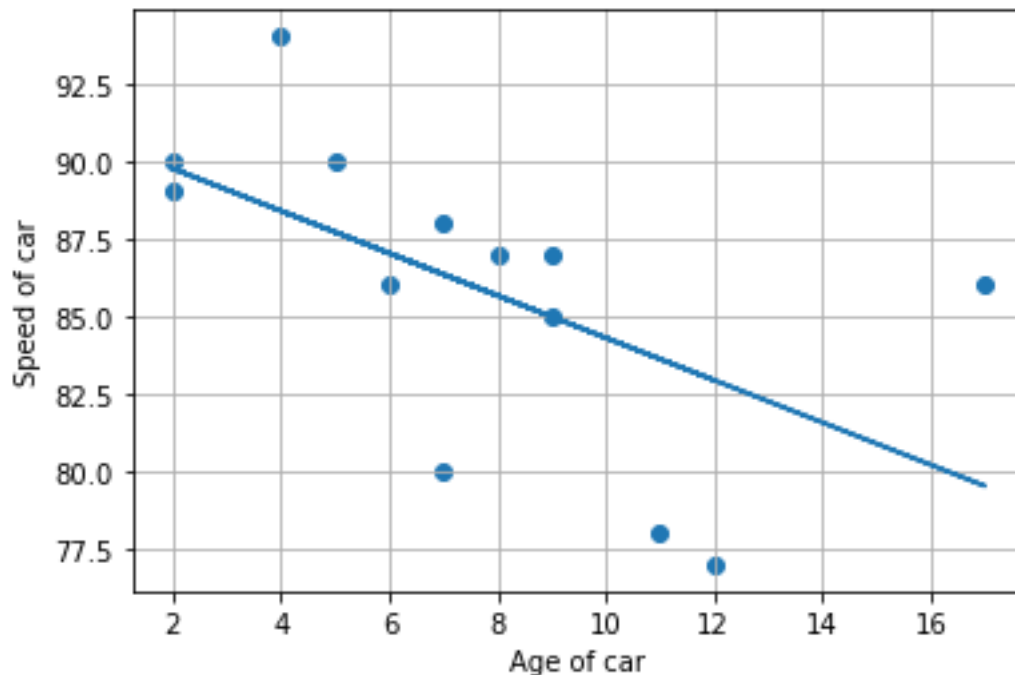


Figure 2 Plot for Question 2

3. Estimate the coefficient of correlation, r , and explain what it means [4 marks]

```
print(r)
... #3 estimate coefficient
... print(r)
-0.5753514642532261
```

The coefficient of correlation r ranges from 1 to -1. It is used as a measurement of the relationship between 2 variables usually x and y . Any value of r that is greater than 1 or -1 represents an error. Values of r that are close to 1 and -1 represent better relationship between variables. If they are exactly 1 and -1, they represent perfectly positive and negative relationships between the variables. Values that are closer to 0 represent a poor relationship between 2 variables. In this case, an r value of -0.5753 represents an adequate or average negative relationship between the car year (x) and the speed of the car (y).

4. Predict the speed of a 13 year old car using the linear reg. model [4 marks]

```
Car_Age=13
Speed_Predict= Car_Age *slope + intercept
print(Speed_Predict)
```

```
...: print(Speed_Predict)  
82.2560706401766
```

5. Fit the data to a third degree polynomial and plot it on the scatter plot[4 marks]

```
n=3  
mymodel=np.poly1d(np.polyfit(x,y,n))  
myline=np.linspace(1,17,100)  
plt.scatter(x,y)  
plt.plot(myline,mymodel(myline))  
plt.xlabel('Age of car')  
plt.ylabel('Speed of car')  
plt.grid()  
plt.show()
```

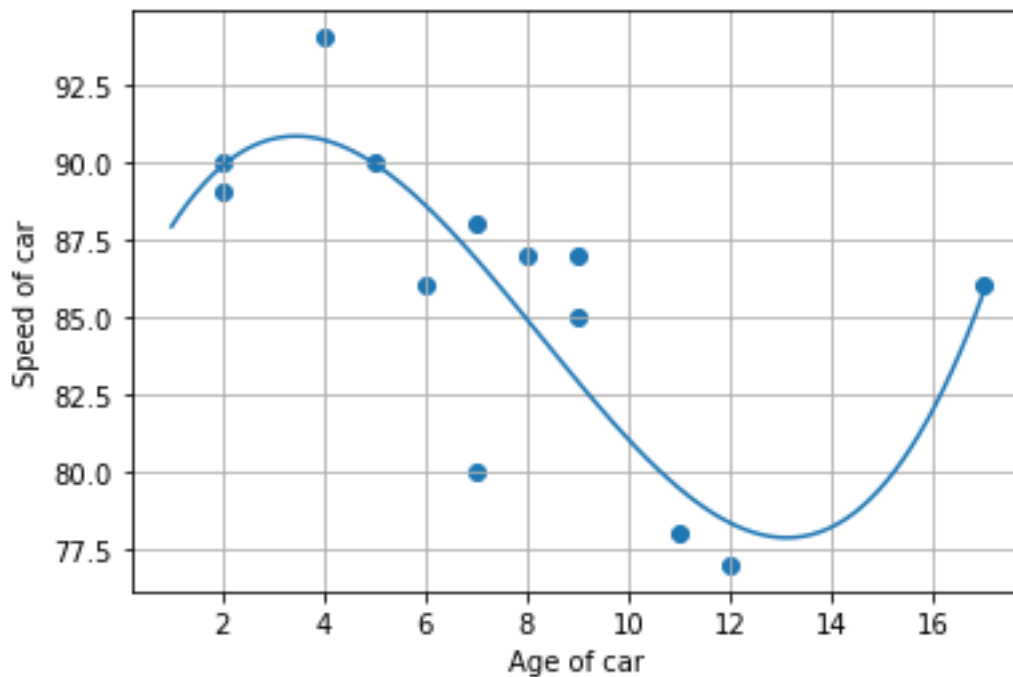
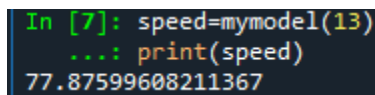


Figure 3 3rd degree polynomial plot for Q5

6. Predict the speed of a 13 year old car using the 3rd deg. polynomial [4 marks]

```
speed=mymodel(13)
```

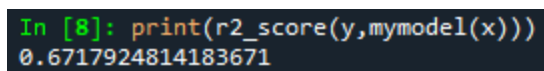
```
print(speed)
```

A screenshot of a Jupyter Notebook cell showing the execution of two lines of Python code. The first line is 'In [7]: speed=mymodel(13)' and the second line is '...: print(speed)'. The output of the code is '77.87599608211367'.

As seen from the image above the predicted speed of a 13 year old car using the 3rd degree polynomial is 77.876

7. Estimate the r-squared value and explain what it means [4 marks]

```
print(r2_score(y,mymodel(x)))
```

A screenshot of a Jupyter Notebook cell showing the execution of a single line of Python code: 'In [8]: print(r2_score(y,mymodel(x)))'. The output of the code is '0.6717924814183671'.

As seen from the image above, the estimated r-squared value is 0.6718. The r squared value ,also known as the coefficient of determination, is used to determine the relationship between x axis values and the y axis values. The r squared value ranges from 0 to 1 where a value of '0' indicates that there is no relationship between the x and y axis values and a '1' indicates that the values of the x and y axis are 100% related. If there is no relationship the polynomial regression cannot be used for prediction. In this case , the r-squared value was estimated to be 0.6718. This indicates that the age of the car (x axis values) are 67.18% related to the speed of the car (y axis values).

8. Explain any differences in values obtained in (4) and (6) [4 marks]

The predicted speed of a 13 year old car using the linear regression model is : 82.256 with a r value of -0.57535

The predicted speed of a 13 year old car using the polynomial regression model is : 77.876 with a r-squared value of 0.6718.

From the values stated above , it can be seen that the predicted speed from the linear model was higher than that of the polynomial model. There are several reasons for this difference.

- The linear model best fit line has several datapoints that are very far from it, [(7,80) , (11,78), (12,77) ,(17,86)]. Since these points are far from the best fit line, it can be said that the line is not a perfect fit which means that the prediction can be quite higher or lower compared to the actual data points. Compared to the polynomial model line, there are only 2 points that are far from the best fit line [(4,94) , (5,80)] , this means that the best fit line can provide a better prediction that is closer to the actual data points since it has less outliers.
- This first point can also be illustrated by looking at the r and r-squared values. The r-squared value of the polynomial model is closer to the perfect relationship indicator (r-squared value of '1') than the linear model r value (r value of '-1'). This fact can then indicate that the polynomial model has a better/stronger relationship between axes than the linear model which means that the predicted value will be closer to actual data points. This results in the prediction of the polynomial model being smaller than the linear model since it is more accurate.

Use the attached excel file data1.csv to perform the following:

9. Read in the data1 file into your workspace and store it as a data frame [4 marks]

```
dir_path = r"C:\Users\Admin\Desktop\compEng\Year  
3\Semester2\AIML\AIML\Assignment1"  
file_name = "data1.csv"  
file_path = os.path.join(dir_path, file_name)  
  
# reading in the file as an object  
df = pd.read_csv(file_path)
```

10. Extract the weight and volume fields and store it in a variable, X [2 marks]

```
X = df[["Weight", "Volume"]]
```

11. Extract the CO2 field and store it as variable, y [2 marks]

```
y = df[["CO2"]]
```

12. Fit a Linear Regression model to variables X and y [4 marks]

```
linReg = linear_model.LinearRegression()  
linReg.fit(X, y)
```

13. Predict value of CO₂ of the Volvo XC70 with weight of 1746 kg & volume of 2000cm³ , and explain why this value is different from 117, as given in the data file. [6 marks]

```
predictedCO2 = linReg.predict([[1746, 2000]])
```

predictedCO2	Array of float64	(1, 1)	[[108.71820661]]
--------------	------------------	--------	------------------

The predicted CO₂ of the Volvo XC70 with weight of 1746 kg & volume of 2000cm³ is : 108.7182 grams.

This is smaller than the value in the data1.csv file which states that the CO₂ value is 117g. This discrepancy can be due several factors.

- The varying values of weight and volumes may cause the model to not perfectly fit the data which can result in a different value from the actual datapoint
- The multiple regression model uses several assumptions such as there being a linear relationship between the x and y values and that the x values are not too correlated to each other. Both these assumptions can results in differences in the predicted value.
- Every model has some error value called the residual value, E, which is the difference between the predicted value and the actual value, to compensate for it not being able to perfectly predict values.

14. Find the reg. coefficient between X and y and explain its meaning [4 marks]

```
print(linReg.coef_)
```

```
In [15]: print(linReg.coef_)  
[[0.00766976 0.00818285]]
```

The regression coefficients for this model is :

Weight = 0.00766976

Volume = 0.00818285

This means that for every 1 kg of weight increase, the CO₂ output increases by 0.00766976g and for every 1cm³ of volume increase the CO₂ output increases by & 0.00818285g.

15. Find the R² (R-squared) score and explain its meaning [4 marks]

```
print(linReg.score(X,y))
```

```
In [16]: print(linReg.score(X,y))  
0.40901685950193534
```

The R² score is : 0.4090. The R² score is used to measure the relationship between the x-axis and the y-axis values. It ranges from 0 to 1, where '0' means no relationship and a '1' indicates a perfect relationship. Since the R² value obtained in this case is 0.4090, it means that there is a relationship between the x and y axis values since it is greater than '0' however the relationship is mediocre since it is not particularly close to '1'.

16. Scale the variables in X using the standardization method [4 marks]

```
scale = StandardScaler()
```

```
scaledX = scale.fit_transform(X)
```

scaledX - NumPy object array			
	0	1	
0	-2.03852	-1.84635	
1	-0.956365	-1.32625	
2	-1.5016	-1.53429	
3	-1.76798	-1.84635	
4	-0.623394	-0.286055	
5	-1.5016	-1.09221	
6	-0.75242	-0.67613	
7	0.313085	-0.286055	
8	-0.739934	-0.286055	
9	-0.581773	-0.026005	
10	-1.28933	-1.32625	
11	-1.24771	-0.806155	
12	-0.739934	-1.5863	
13	-0.157236	-0.026005	
14	0.150762	-0.026005	
15	0.16741	-0.026005	
16	0.313085	-0.026005	
17	-0.0406964	1.53429	
18	-0.710799	-0.026005	
19	0.159086	1.01419	
20	1.22459	-0.026005	
21	0.575299	1.01419	
22	0.313085	1.27424	
23	0.521191	-0.026005	
24	0.521191	1.01419	
25	0.729297	-0.286055	
26	0.833351	1.01419	
			27
			28
			29
			30
			31
			32
			33
			34
			35

27	1.81145	1.01419
28	0.970701	-0.026005
29	1.72821	1.01419
30	1.312	1.27424
31	1.89886	1.01419
32	-0.227992	-0.026005
33	0.417138	-0.026005
34	0.47957	-0.026005
35	0.437948	2.31444

Figure 4 Image showing Scaled values for Question16

17. Repeat question (13) using this scaled X variable [2 marks]

```
linRegScaled = linear_model.LinearRegression()
linRegScaled.fit(scaledX, y)
scaledValue = scale.transform([[1746, 2000]])
scaledPredictedCO2 = linRegScaled.predict([scaledValue[0]])
```

```
scaledPredictedCO2  Array of float64  (1, 1)  [[108.71820661]]
```

18. Explain any differences in values in ques: (13) & (17) [2 marks]

The predicted CO₂ of the Volvo XC70 with weight of 1746 kg & volume of 2000cm³ in question 13 is : 108.7182 grams.

The predicted CO₂ of the Volvo XC70 with weight of 1746 kg & volume of 2000cm³ in question 17 is : 108.7182 grams.

Both Predicted CO₂ values are the same thus there is no difference. There is no difference between the values since the scaled values preserve the relationship between the x axis values and the y axis values.

19. Explain what is the train/test method and why is it important [4 marks]

The train/ test method is used to determine the accuracy of a model. This method as its name suggests involves training and testing. The data set is split into a training set and a testing set, where 80% of the data is used for training and the remaining 20% of data is used for the testing. Training the model means creating the model and testing the model means testing how accurate the model is. The train/test model is important because it can help determine if the model is accurate and good enough to be used to predict values. This is achieved by examining the R² score of the training set and the testing set. Once both shows an okay relationship, then the model is suitable for prediction.

20. Apply the train/test method as follows: i) use the first 30 values of variables X and y to train a Linear Regression model, ii) repeat question (13), (14), (15) and iii) explain any differences between the values obtained in here and in question (13) for the value of the CO₂ (calculate % differences between predicted and actual value for your discussion) for the Volvo XC70, the reg. coefficient and R-squared values. [10 marks]

i)

```
#-----training section-----
```

```
#extract first 30 rows from dataframe
```

```
trainx= df[["Weight","Volume"]].head(30)
```

```
trainy=df["CO2"].head(30)
```

```
#fit training data to linear model
```

```
TrainlinReg = linear_model.LinearRegression()
```

```
TrainlinReg.fit(trainx, trainy)
```

ii)

Q13 for Training Set

```
#predict value of co2 for volvo Xc70 with weight of 1746kg and vol of 2000cm
```

```
Train_predictedCO2 = TrainlinReg.predict([[1746, 2000]])
```

Predicted value for Train Lin Reg Model

```
Train_predictedCO2  Array of float64  (1,)  [105.28221078]
```

Q14 for Training Set

```
#find the reg coefficient between X and y and explain its meaning
```

```
print(TrainlinReg.coef_)
```

Reg Coefficient Value for Train Lin Reg Model

```
In [22]: print(TrainlinReg.coef_)  
[0.00681982 0.00431929]
```

This means that for every 1 kg of weight increase, the CO₂ output increases by 0.00681982g and for every 1cm³ of volume increase the CO₂ output increases by & 0.00431929g.

Q15 for Training Set

#find the R2 score and explain its meaning

```
print(TrainlinReg.score(trainx,trainy))
```

R2 Value for Train Lin Reg model

```
In [23]: print(TrainlinReg.score(trainx,trainy))  
0.2695794301945249
```

This R2 value means that there is a relationship between the x axis values and y axis values since it is greater than 0, but the relationship is not very good since it is fairly far away from 1.

iii)

Table 1 PredictedCO2, R and R2 values

	All Values	First 30 Values(Trained Model)
Predicted CO2 for Volvo Xc70	108.1782	105.28221
R coefficient	0.00766976 , 0.00818285	0.00681982 ,0.00431929
R2 Value	0.40901685950193534	0.2695794301945249

Actual CO2 Output for Volvo Xc70 : 117g

From the table Above, it can be seen that the predicted CO2 output , R coefficient and R2 values for the 30 values Trained Model is smaller than the Model that was trained with all the values of the dataset.

The difference between the Predicted CO2, R coefficient and R2 values can be due to the fact that there were less datapoints used in the 30 datapoint Trained model

compared to the model with all 36 datapoints. If less datapoints are used , it can mean that the model will be less accurate compared to one with more datapoints. Evidence to support this is the fact that the R2 value of the 30 datapoint model is less than the all datapoint model. Smaller R2 values tends to be less accurate since the relationship between axes are poor the close the R2 value is to 0.

Another cause of the difference is that, the value we want to predict, CO2 for Volvo Xc70, was not used in the training of the model.

The Remaining Datapoints , 31 to 36, were then used to test the model. The Weight and Volume for each of these datapoints were used as inputs to the model in order to predict the output CO2. The table below shows the Predicted and actual values. The percentage difference was also calculated between the predicted and actual values and shown in the image below.

The percentage difference was calculated according the following formula:

$$percentdiff = \frac{predictedCO2 - ActualCO2}{ActualCO2} \times 100$$

Table 2 Predicted CO2 and Actual CO2 values for testing set

Car	Model	Predicted CO2	ActualCO2
Mercedes	E-Class	104.75254549	115
Volvo	XC70	105.28221078	117
Ford	B-Max	100.0695657	104
BMW	216	101.12663776	108
Opel	Zafira	101.22893506	109
Mercedes	SLK	105.04810053	120

Table 3 Percent Diff between Predicted CO2 and Actual CO2 values for testing set

Car	Model	PercentDiff
Mercedes	E-Class	8.91083001
Volvo	XC70	10.01520446
Ford	B-Max	3.77926375
BMW	216	6.36422429
Opel	Zafira	7.12941738
Mercedes	SLK	12.45991622

```

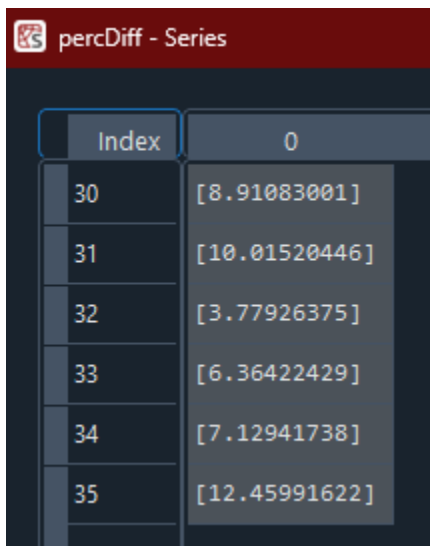
#-----Testing Section-----
#extract last 6 rows from dataframe since the first 30
#rows were used for training. There are 36 rows in total
testx= df[["Weight","Volume"]].tail(6)
testy=df["CO2"].tail(6)

testResults=pd.Series(dtype=float,name="PredCO2")

#predict CO2 for each value of the testing set
for index, row in testx.iterrows():
    #predictCo2
    testResults.loc[index]=TrainlinReg.predict([[row["Weight"],
row["Volume"]]])

#calc percentage diff between actual values and predicted values
percDiff=abs(((testResults-testy)/testy)*100)

```



Index	0
30	[8.91083001]
31	[10.01520446]
32	[3.77926375]
33	[6.36422429]
34	[7.12941738]
35	[12.45991622]

Please type out your solutions in this word document just after the respective question (so that I can see the marks allocated when I am marking), then convert it to a pdf (file ➡ save as ➡ type pdf, make sure you have Adobe Acrobat Reader installed) submit your solutions as a pdf file on canvas

CODE:

```
#Christopher Teelucksingh
```

```
#89496
```

```
#AIML3001 Assignment 1
```

```
import matplotlib.pyplot as plt
```

```
from scipy import stats
```

```
import numpy as np
```

```
from sklearn.metrics import r2_score
```

```
import pandas as pd
```

```
import os
```

```
from sklearn import linear_model
```

```
from sklearn.preprocessing import StandardScaler
```

#1 scatter plot

```
x=[5,7,8,7,2,17,2,9,4,11,12,9,6]
```

```
y=[90,80,87,88,90,86,89,87,94,78,77,85,86]
```

```
plt.scatter(x,y)
```

```
plt.xlabel('Age of car')
```

```
plt.ylabel('Speed of car')
```

```
plt.show()
```

#2 linear regression

```
slope,intercept,r,p,std_err = stats.linregress(x,y)
```

```
Reg_LineY = np.array(x)*slope + intercept
```

```
plt.scatter(x,y)
```

```
plt.plot(x,Reg_LineY)
```

```
plt.xlabel('Age of car')
```

```
plt.ylabel('Speed of car')
```

```
plt.grid()
```

```
plt.show()
```

#3 estimate coefficient of correlation r

```
print(r)
```

#4 predict spd of 13 yr old car using lin reg model

```
Car_Age=13
```

```
Speed_Predict= Car_Age *slope + intercept  
print(Speed_Predict)
```

```
#5 3rd degree polynomial plot of data  
n=3  
mymodel=np.poly1d(np.polyfit(x,y,n))  
myline=np.linspace(1,17,100)  
plt.scatter(x,y)  
plt.plot(myline,mymodel(myline))  
plt.xlabel('Age of car')  
plt.ylabel('Speed of car')  
plt.grid()  
plt.show()
```

```
#6 predict speed of 13 year old car using 3rd degree polynomial  
speed=mymodel(13)  
print(speed)
```

```
#7 estimate the r-squared value  
print(r2_score(y,mymodel(x)))
```

```
#8 difference between lin and poly prediction
```

```
#-----excel sheet section-----
```

```
#9 read in excel file and store the data in a dataframe
```

```
dir_path = r"C:\Users\Admin\Desktop\compEng\Year  
3\Semester2\AIML\AIML\Assignment1"
```

```
file_name = "data1.csv"
```

```
file_path = os.path.join(dir_path, file_name)
```

```
# reading in the file as an object
```

```
df = pd.read_csv(file_path)
```

```
#10 extract weight and volume fields and store in X
```

```
X = df[["Weight", "Volume"]]
```

```
#11 extract co2 field and store in variable y
```

```
y = df[["CO2"]]
```

```
#12 fit a lin reg model to variables X and y
```

```
linReg = linear_model.LinearRegression()
```

```
linReg.fit(X, y)
```

```
#13 predict value of co2 for volvo Xc70 with weight of 1746kg and vol of 2000cm
```

```
predictedCO2 = linReg.predict([[1746, 2000]])
```

```
#14 find the ref coefficient between X and y and explain its meaning  
print(linReg.coef_)
```

```
#15 find the R2 score and explain its meaning  
print(linReg.score(X,y))
```

```
#16 scale the variables in X using the standardization method  
scale = StandardScaler()  
scaledX = scale.fit_transform(X)
```

```
#17 repeat step 13 using scaled X var  
linRegScaled = linear_model.LinearRegression()  
linRegScaled.fit(scaledX, y)  
scaledValue = scale.transform([[1746, 2000]])  
scaledPredictedCO2 = linRegScaled.predict([scaledValue[0]])
```

```
#20 applying train/test method
```

```
#-----training section-----
```

```
#extract first 30 rows from dataframe
```

```
trainx= df[["Weight","Volume"]].head(30)
```

```
trainy=df["CO2"].head(30)
```

```
#fit training data to linear model
```

```
TrainlinReg = linear_model.LinearRegression()
```

```
TrainlinReg.fit(trainx, trainy)
```

```
#predict value of co2 for volvo Xc70 with weight of 1746kg and vol of 2000cm
```

```
Train_predictedCO2 = TrainlinReg.predict([[1746, 2000]])
```

```
#find the ref coefficient between X and y and explain its meaning
```

```
print(TrainlinReg.coef_)
```

```
#find the R2 score and explain its meaning
```

```
print(TrainlinReg.score(trainx,trainy))
```

```
#-----Testting Section-----
```

```
#extract last 6 rows from datafram since the first 30
```

```
#rows were used for training.There are 36 rows in total
```

```
testx= df[["Weight","Volume"]].tail(6)
```

```
testy=df["CO2"].tail(6)
```

```
testResults=pd.Series(dtype=float,name="PredCO2")
```

```
#predict CO2 for each value of the testing set
```

```
for index, row in testx.iterrows():
```

```
    #predictCo2
```

```
    testResults.loc[index]=TrainlinReg.predict([[row["Weight"], row["Volume"]]])
```

```
#calc percentage diff between actual values and predicted values
```

```
percDiff=abs(((testResults-testy)/testy)*100)
```