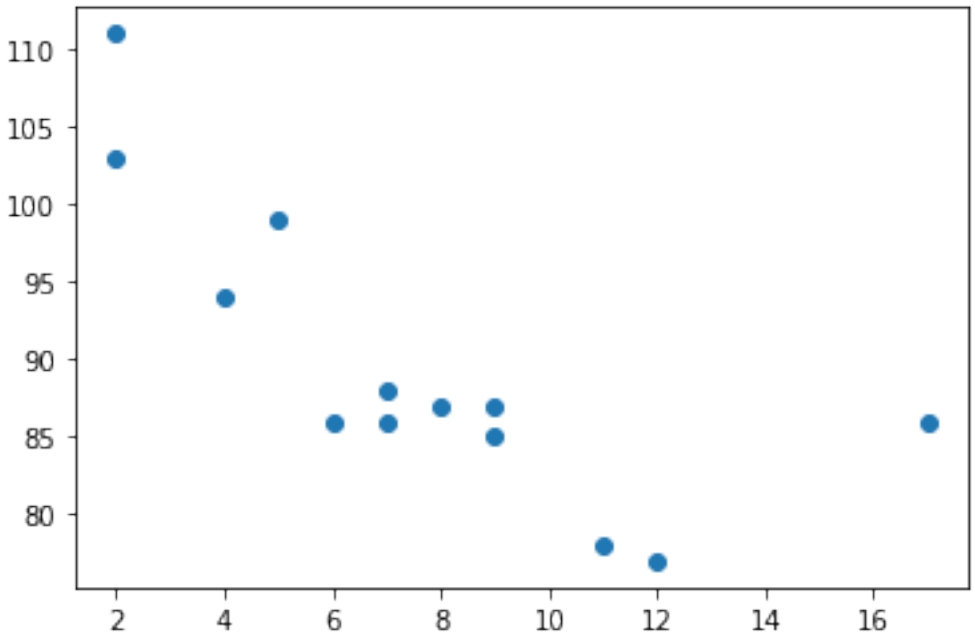
20220119_Supplier_Management_MBW7

January 19, 2022

[1]:
```
# machine learning basics
```

[2]:
```
# linear regression

# Uses the relationship btw the data points to draw a straight line through all␣
 ↪them.

# this line can be used to predict values.

import matplotlib.pyplot as plt

x =[5,7,8,7,2,17,2,9,4,11,12,9,6]
y =[99,86,87,88,111,86,103,87,94,78,77,85,86]

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

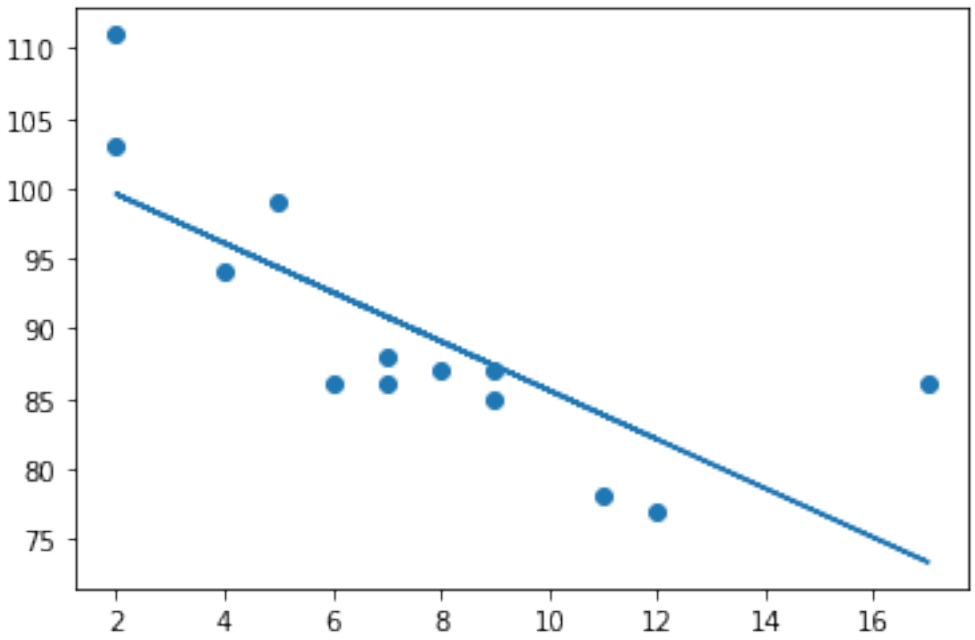


[3]:
```
# execute a method that returns some important key values of the linear␣
↪regression

from scipy import stats

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

[4]:
```
# create a function that uses the slope and intercept values to return a new␣
↪value.
# this new value represents where on the y-axis the corresponding x value will␣
↪be placed

def myfunc(x):
    return slope * x + intercept
```

[5]:
```
# run each value of the x array through the function.
# this results in a new array with a new values for the y-axis

mymodel = list(map(myfunc,x))
```

[6]:
```
# draw the plot

plt.scatter(x,y)
plt.plot(x,mymodel)
plt.show()
```

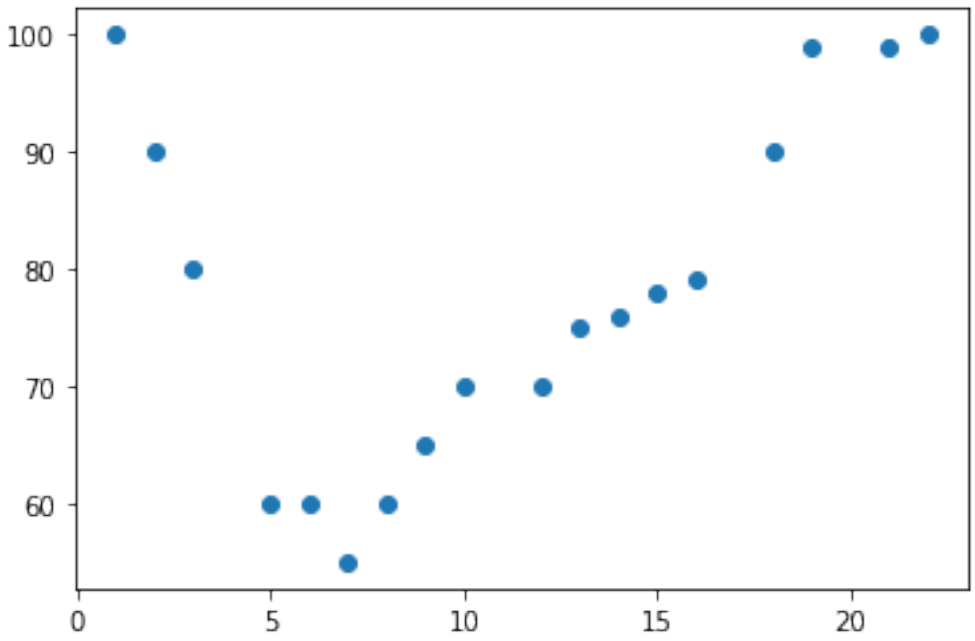


[8]:
```
# predict new values

speed = myfunc(10)

print(speed)
```

```
85.59308314937454
```

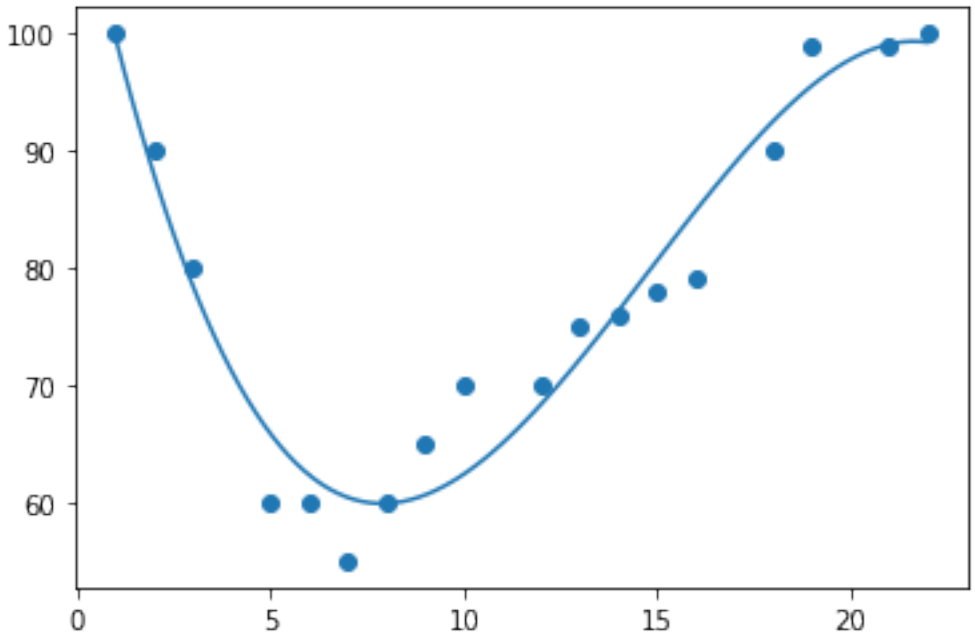
[9]:
```
# polynomial regression

# Python has methods for finding relationships btw data points and to draw a␣
 ↪polinomial regression.
```

[10]:
```
x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

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



[15]:
```
import numpy as np
import matplotlib.pyplot as plt

# method to display a polynomial regression model

mymodel = np.poly1d(np.polyfit(x,y,3))
```

```
#specify how the line will display, we start at position 1, and end at position␣
 ↪22

myline = np.linspace(1,22,100)

plt.scatter(x,y)
plt.plot(myline,mymodel(myline))
plt.show()
```



[16]:
```
# predict new values

speed = mymodel(17)
print(speed)
```

```
88.87331269697987
```

[18]:
```
# mutltiple regression

# MR is like "linear regression" but with more than one independent value,
# meaning that we try to predict a value based on two or more variables.

import pandas

# the pandas packages allows us to load .csv files

df = pandas.read_csv("/Users/h4/desktop/cars.csv")
```

[19]:
```
df.head()
```

[19]:

	Car	Model	Volume	Weight	CO2
0	Toyoty	Aygo	1000	790	99
1	Mitsubishi	Space Star	1200	1160	95
2	Skoda	Citigo	1000	929	95
3	Fiat	500	900	865	90
4	Mini	Cooper	1500	1140	105

[21]:
```
# variables

X = df[['Weight','Volume']]

# independent value

y = df[['CO2']]
```

[22]:
```
# The method to do the mult. linear regression is from package sklearn

from sklearn import linear_model

# From the sklearn module we will use LinearRegression() method to create a␣
 ↪linear regression object.
# This object has a method called "fit()" that takes the indepndent and␣
 ↪dependent variables as parameters and
# fills the regression object with data that describes the relationship.

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

[22]:
```
LinearRegression()
```

[24]:
```
# predict the CO2 of a car where the weight is 2300Kg, and volume is 1300 cm3

predictedCO2 = regr.predict([[2300,1300]])

predictedCO2
```

```
/Users/h4/opt/anaconda3/lib/python3.8/site-packages/sklearn/base.py:445:
UserWarning: X does not have valid feature names, but LinearRegression was
fitted with feature names
  warnings.warn(
```

[24]:
```
array([[107.2087328]])
```

[26]:
```
# Coefficient

```

```
# Coefficient is a factor that describes the relationship with an unknown␣
 ↪variable.

# We can ask for the coefficient value of weight against CO2, and for volume␣
 ↪against CO2.

print(regr.coef_)
```

```
[[0.00755095 0.00780526]]
```

[27]:
```
# This result array represents the coefficients values of weight and volume.
#These values tells us that if the weight increases by 1kg, the CO" emission␣
 ↪increases by 0.00755095gr
```

[30]:
```
# Scale Features

# When yzour data has different values, and even different measurement units,␣
 ↪it can be difficult to compare them.

df2 = pandas.read_csv('/Users/h4/desktop/cars2.csv', sep=';')
```

[31]:
```
df2.head()
```

[31]:

	Car	Model	Volume	Weight	CO2
0	Toyota	Aygo	1.0	790	99
1	Mitsubishi	Space Star	1.2	1160	95
2	Skoda	Citigo	1.0	929	95
3	Fiat	500	0.9	865	90
4	Mini	Cooper	1.5	1140	105

[32]:
```
# z = x-mean(x)/stddev(x)
```

[34]:
```
from sklearn.preprocessing import StandardScaler

scale=StandardScaler()

scaleX = scale.fit_transform(X)

print(scaleX)
```

```
[[-2.10389253 -1.59336644]
 [-0.55407235 -1.07190106]
 [-1.52166278 -1.59336644]
 [-1.78973979 -1.85409913]
 [-0.63784641 -0.28970299]
 [-1.52166278 -1.59336644]
 [-0.76769621 -0.55043568]
```

```
 [ 0.3046118  -0.28970299]
 [-0.7551301  -0.28970299]
 [-0.59595938 -0.0289703 ]
 [-1.30803892 -1.33263375]
 [-1.26615189 -0.81116837]
 [-0.7551301  -1.59336644]
 [-0.16871166 -0.0289703 ]
 [ 0.14125238 -0.0289703 ]
 [ 0.15800719 -0.0289703 ]
 [ 0.3046118  -0.0289703 ]
 [-0.05142797  1.53542584]
 [-0.72580918 -0.0289703 ]
 [ 0.14962979  1.01396046]
 [ 1.2219378  -0.0289703 ]
 [ 0.5685001   1.01396046]
 [ 0.3046118   1.27469315]
 [ 0.51404696 -0.0289703 ]
 [ 0.51404696  1.01396046]
 [ 0.72348212 -0.28970299]
 [ 0.8281997   1.01396046]
 [ 1.81254495  1.01396046]
 [ 0.96642691 -0.0289703 ]
 [ 1.72877089  1.01396046]
 [ 1.30990057  1.27469315]
 [ 1.90050772  1.01396046]
 [-0.23991961 -0.0289703 ]
 [ 0.40932938 -0.0289703 ]
 [ 0.47215993 -0.0289703 ]
 [ 0.4302729   2.31762392]]
```

[ ]: