ABDELGHAFOR'S VIRTUAL INTERNSHIP

MACHINE LEARNING PROGRAM

SESSION (1)

PREPARED BY : MARK KOSTANTINE

LECTURE OVERVIEW

- Machine Learning Introduction
- Supervised Learning : Get Started
- Linear Regression
- Polynomial Regression
- Multiple Regression
- Tasks
- Questions

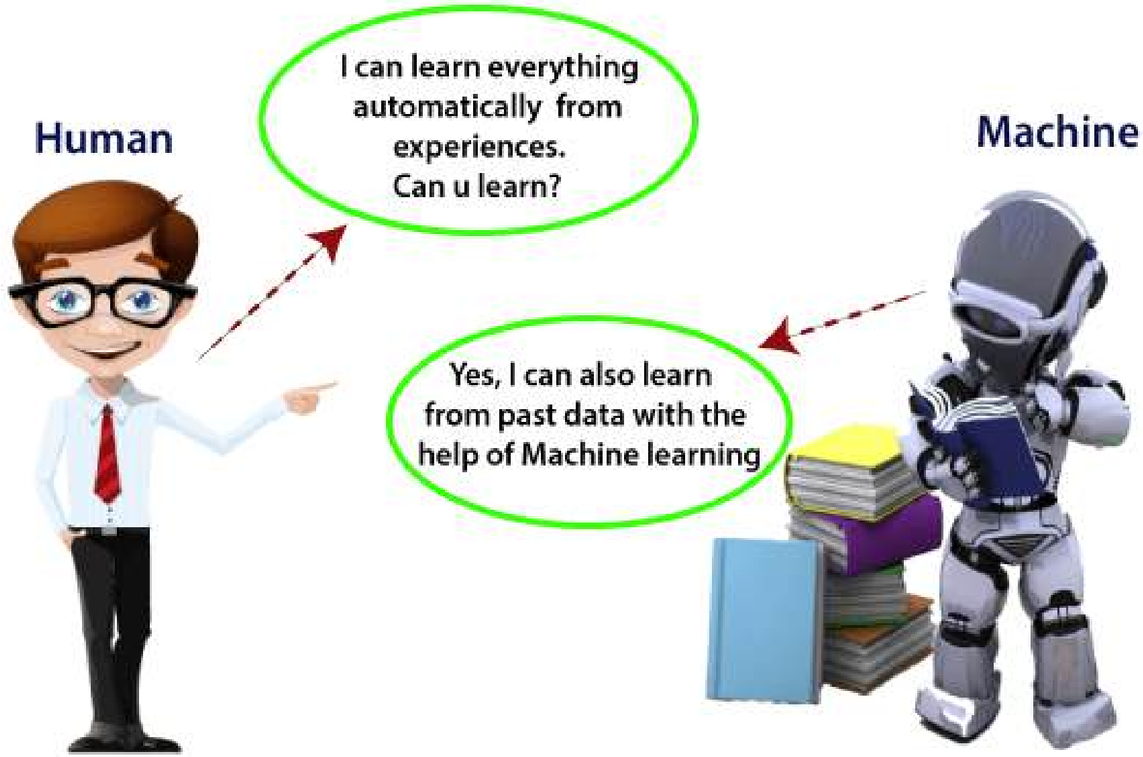
WHAT IS MACHINE LEARNING ?

Machine Learning is said as a subset of **artificial intelligence** that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959

We can define it in a summarized way as :

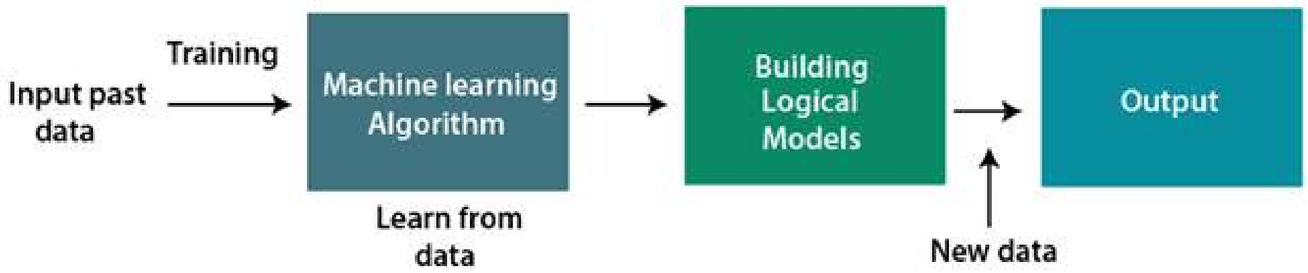
- Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

A machine has the ability to learn if it can improve its performance by gaining more data.



HOW DOES MACHINE LEARNING WORK ?

A Machine Learning system builds prediction models, learns from previous data, and predicts the output of new data whenever it receives it. The amount of data helps to build a better model that accurately predicts the output, which in turn affects the accuracy of the predicted output.



FEATURES OF MACHINE LEARNING

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

CLASSIFICATION OF MACHINE LEARNING

At a broad level, machine learning can be classified into three types:

- **Supervised learning**
- **Unsupervised learning**
- **Reinforcement learning**

DEEP LEARNING VS. MACHINE LEARNING

Machine learning algorithms leverage structured, labeled data to make predictions—meaning that specific features are defined from the input data for the model and organized into tables. This doesn't necessarily mean that it doesn't use unstructured data; it just means that if it does, it generally goes through some pre-processing to organize it into a structured format.

Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts.

SUPERVISED MACHINE LEARNING

- Supervised learning is the types of **machine learning** in which machines are trained using well **"labelled"** training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

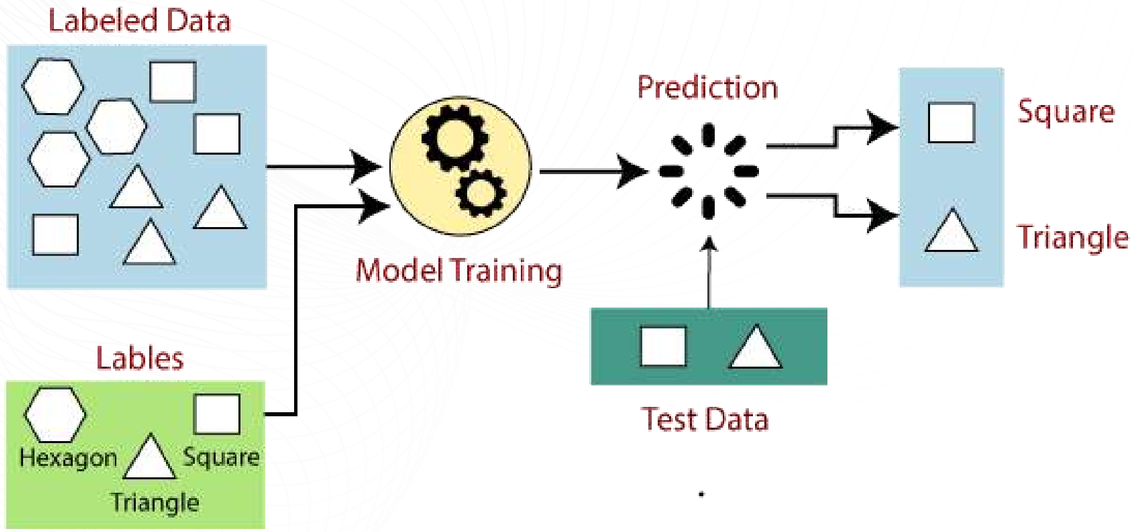
- In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

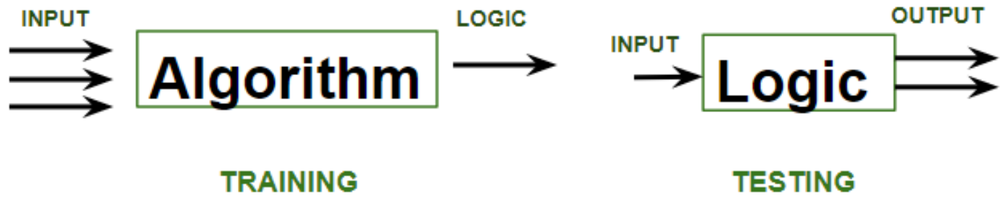
- Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to **find a mapping function to map the input variable(x) with the output variable(y).**

- In the real-world, supervised learning can be used for **Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.**

HOW SUPERVISED LEARNING WORKS?

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.



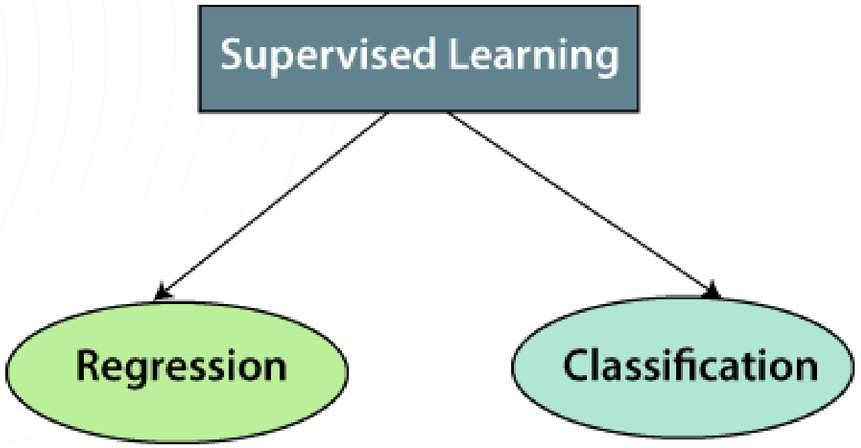


STEPS INVOLVED IN SUPERVISED LEARNING

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training **dataset, test dataset, and validation dataset.**
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

TYPES OF SUPERVISED LEARNING ALGORITHMS

Supervised learning can be further divided into two types of problems



- **Regression:** Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.
- **Classification:** Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

TYPES OF SUPERVISED LEARNING ALGORITHMS

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

ADVANTAGES OF SUPERVISED LEARNING

- With the help of supervised learning, the model can predict the output on the basis of prior experiences.
- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering, etc.**

DISADVANTAGES OF SUPERVISED LEARNING

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

LINEAR REGRESSION

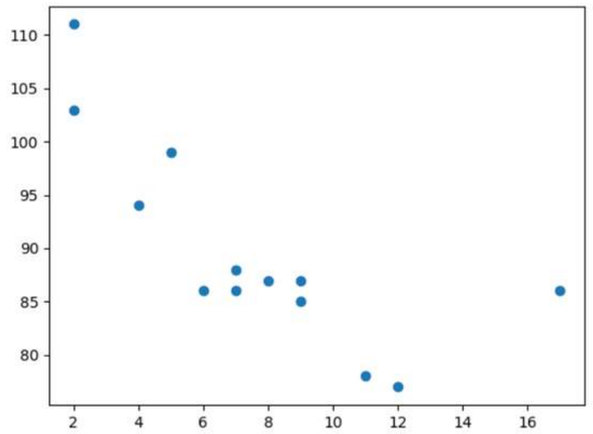
Linear Regression

- Linear regression uses the relationship between the data-points to draw a straight line through all them.
- This line can be used to predict future values.
- In Machine Learning, predicting the future is very important.

How Does it Work?

- Python has methods for finding a relationship between data-points and to draw a line of linear regression.
- how to use these methods instead of going through the mathematic formula.
- Start by drawing a scatter plot

```
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()
```

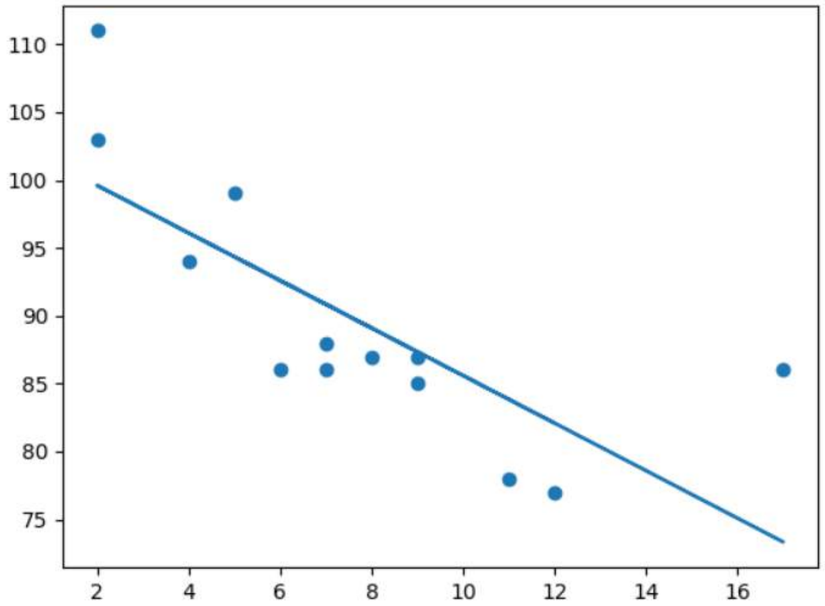


LINEAR REGRESSION

How Does it Work?

- Import **scipy** and draw the line of Linear Regression

```
import matplotlib.pyplot as plt
from scipy import stats

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]

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

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

mymodel = list(map(myfunc, x))

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



LINEAR REGRESSION

R for Relationship

- It is important to know how the relationship between the values of the x-axis and the values of the y-axis is, if there are no relationship the linear regression can not be used to predict anything.
- This relationship - the coefficient of correlation - is called **r**.
- The **r** value ranges from **-1 to 1, where 0 means no relationship, and 1 (and -1) means 100% related.**
- Python and the Scipy module will compute this value for you, all you have to do is feed it with the x and y values.

```
from scipy import stats

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]

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

print(r)
```

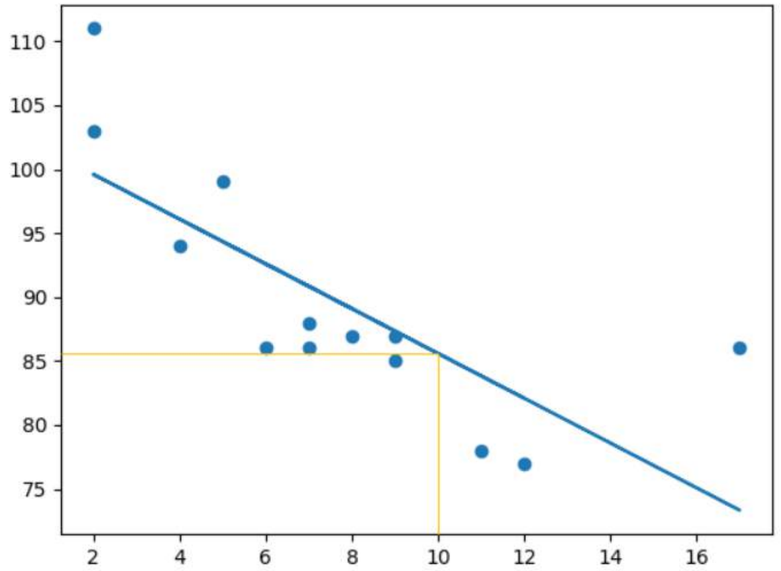
Note: The result **-0.76** shows that there is a relationship, not perfect, but it indicates that we could use linear regression in future predictions.

LINEAR REGRESSION

Predict Future Values

- Now we can use the information we have gathered to predict future values.
- To do so, we need the same myfunc( )

```
from scipy import stats

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]

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

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

speed = myfunc(10)

print(speed)
```

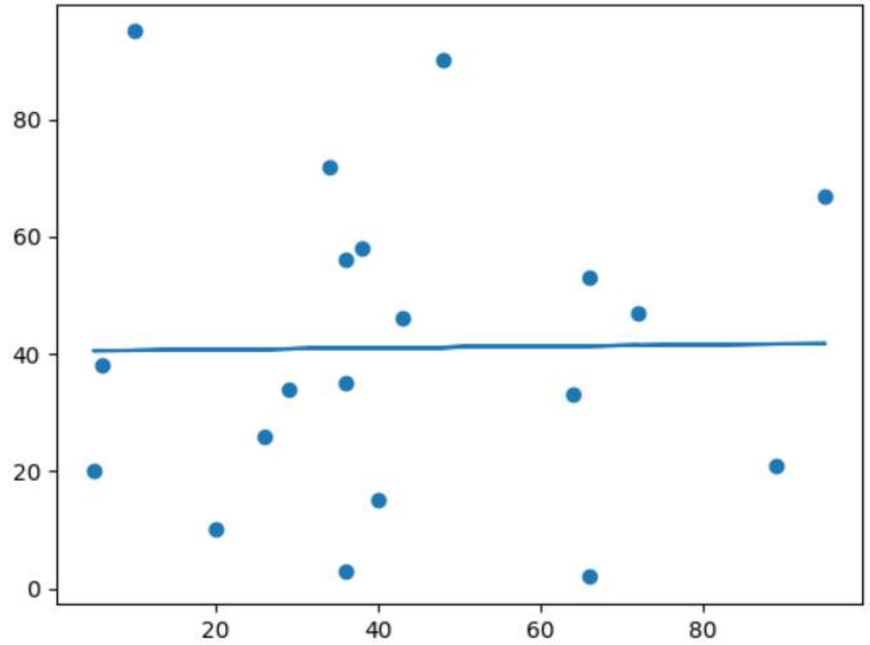


LINEAR REGRESSION

Bad Fit ?

- There’s an example where linear regression would not be the best method to predict future values.

```
import matplotlib.pyplot as plt
from scipy import stats

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

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

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

mymodel = list(map(myfunc, x))

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



LINEAR REGRESSION

Bad Fit ?

- And the r for relationship ?

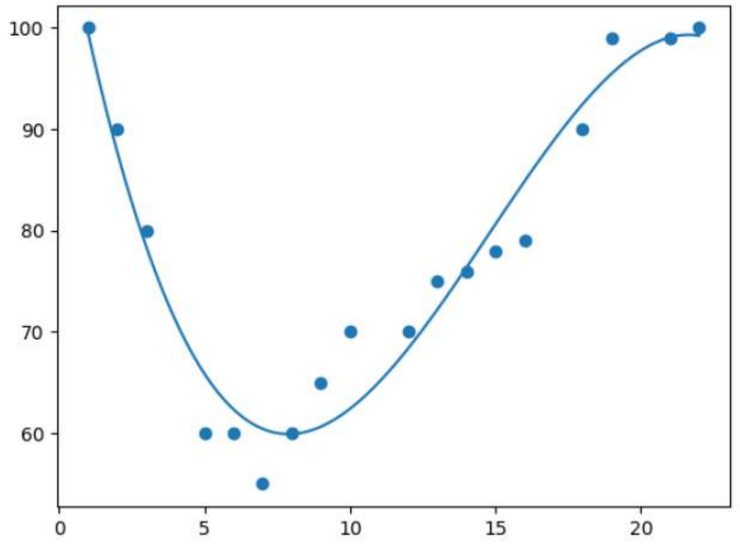
```
import numpy
from scipy import stats

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

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

print(r)
```

The result **0.013** indicates a very bad relationship, and tells us that this data set is not suitable for linear regression.

POLYNOMIAL REGRESSION

What is it?

- If your data points clearly will not fit a linear regression (a straight line through all data points), it might be ideal for polynomial regression.
- Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.

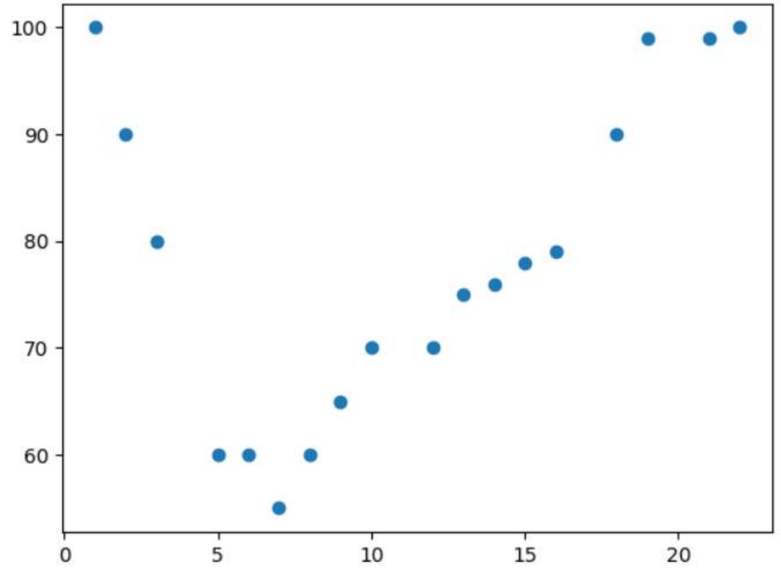


POLYNOMIAL REGRESSION

How Does it Work?

- Python has methods for finding a relationship between data-points and to draw a line of polynomial regression.
- we have registered 18 cars as they were passing a certain tollbooth.
- We have registered the car's speed, and the time of day (hour) the passing occurred.
- The x-axis represents the hours of the day and the y-axis represents the speed

```
import matplotlib.pyplot as plt

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()
```

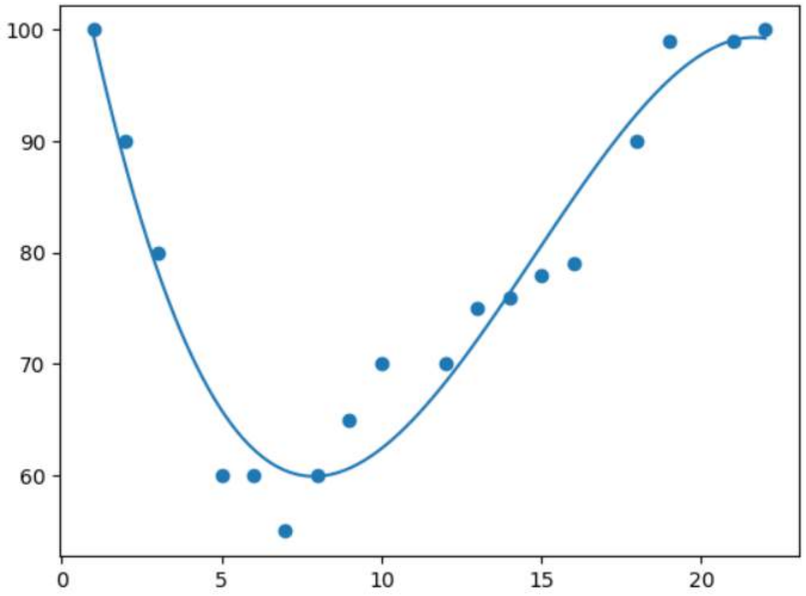


POLYNOMIAL REGRESSION

How Does it Work?

Import **numpy** and **matplotlib** then draw the line of Polynomial Regression

```
import numpy
import matplotlib.pyplot as plt

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]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

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

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



$$y = ax^3 + bx^2 + cx + d$$

POLYNOMIAL REGRESSION

R-Squared

- It is important to know how well the relationship between the values of the x- and y-axis is, if there are no relationship the polynomial regression can not be used to predict anything.
- The relationship is measured with a value called the **r-squared.**
- The **r-squared** value ranges from **0 to 1, where 0 means no relationship, and 1 means 100% related.**
- Python and the Sklearn module will compute this value for you, all you have to do is feed it with the x and y arrays

```
import numpy
from sklearn.metrics import r2_score

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]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

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

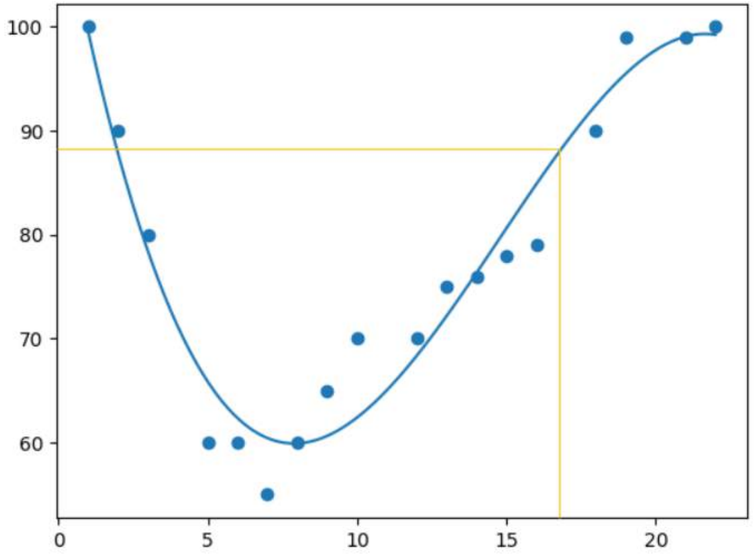
Note: The result **0.94** shows that there is a very good relationship, and we can use polynomial regression in future predictions.

POLYNOMIAL REGRESSION

Predict Future Values

- Now we can use the information we have gathered to predict future values.
- To do so, we need the same **mymodel** array from the previous exam

```
import numpy
from sklearn.metrics import r2_score

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]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

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

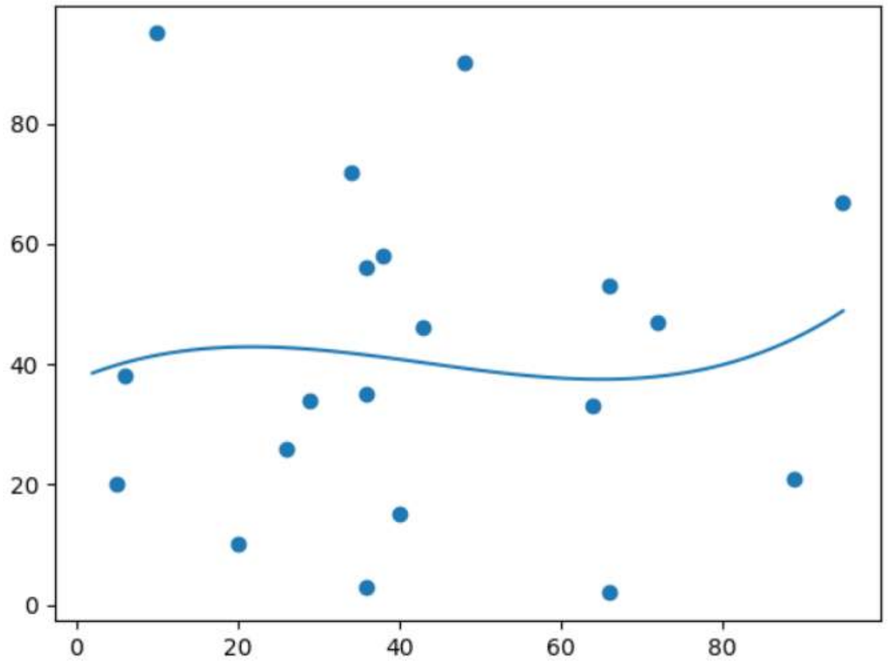


POLYNOMIAL REGRESSION

Bad Fit?

There's an example where polynomial regression would not be the best method to predict future values.

```
import numpy
import matplotlib.pyplot as plt

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(2, 95, 100)

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



POLYNOMIAL REGRESSION

Bad Fit?

And the r-squared value ?

```
import numpy
from sklearn.metrics import r2_score

x = [89,43,36,36,95,10,66,34,38,20,26,29,48,64,6,5,36,66,72,40]
y = [21,46,3,35,67,95,53,72,58,10,26,34,90,33,38,20,56,2,47,15]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

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

The result **0.00995** indicates a very bad relationship, and tells us that this data set is not suitable for polynomial regression.

MULTIPLE REGRESSION

How Does it Work?

Multiple regression 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.

Simple Linear Regression equation: $y = b_0+b_1x$ .....(a)

Multiple Linear Regression equation: $y= b_0+b_1x+ b_2x_2+ b_3x_3+....+ b_nx_n$ .....(b)

Polynomial Regression equation: $y= b_0+b_1x + b_2x^2+ b_3x^3+....+ b_nx^n$ .....(c)

When we compare the above three equations, we can clearly see that all three equations are Polynomial equations but differ by the degree of variables. The Simple and Multiple Linear equations are also Polynomial equations with a single degree, and the Polynomial regression equation is Linear equation with the nth degree. So if we add a degree to our linear equations, then it will be converted into Polynomial Linear equations.

MULTIPLE REGRESSION

Assumptions for Multiple Linear Regression

- A linear relationship should exist between the Target and predictor variables.
- The regression residuals must be normally distributed.
- MLR assumes little or no multicollinearity (correlation between the independent variable) in data.

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("data.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

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

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:
predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
```

MULTIPLE REGRESSION

Coefficient

- The coefficient is a factor that describes the relationship with an unknown variable.
- Example: if **x** is a variable, then **2x** is x two times. **x is the unknown variable, and the number 2 is the coefficient.**

```
import pandas
from sklearn import linear_model

df = pandas.read_csv("data.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

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

print(regr.coef_)
```

- The result array represents the coefficient values of weight and volume.
  - Weight: **0.00755095**
  - Volume: **0.00780526**
- These values tell us that if the weight increase by 1kg, the CO2 emission increases by 0.00755095g.
- And if the engine size (Volume) increases by 1 cm3, the CO2 emission increases by 0.00780526 g.

TASKS

01.

Linear Regression

https://www.kaggle.com/datasets/abhishek14398/salary-dataset-simple-linear-regression

02.

Polynomial Regression

https://www.kaggle.com/datasets/parteekbhatia/polynomial-linear-regression-dataset

03.

Multiple Regression

https://www.kaggle.com/datasets/hussainnasirkhan/multiple-linear-regression-dataset

ANY QUESTIONS?

Feel free to ask

MACHINE LEARNING PROGRAM

THANK YOU

UPCOMING NEXT WEEK : SESSION (2)