AI-ML Training

Prerequisite:

Hardware

Computer/Laptop intel Core i5 and above, with Hard Disk min 500gb and above and min Ram 8gb and above

Internet connect without any proxy settings, as it would be required to install certain Data Science packages during the training (e.g., nltk, keras, TensorFlow, Pytorch)

Software

1) Web Browser

2) Anaconda installed from the below link and the below versions.

Anaconda Packages: https://repo.anaconda.com/archive/

Anaconda for WinOS: Anaconda3-2020.11-Windows-x86_64.exe

Anaconda for MacOS: Anaconda3-2020.11-MacOSX-x86_64.pkg

Required skills: Python

Introduction to Data Science

It consists of 3 fields: AI, ML and DL.

Artificial intelligence:

- Mimic human behavior
- Incorporate the human behavior to machines
- The effort to automate intellectual tasks normally performed by humans.
- Example: Playing Chess, image classification, language translations and speech recognition.

Machine learning:

- Systematic study of algorithms and systems that improve their knowledge or performance with experience.
- Programming paradigm
- System is trained rather than explicitly programmed.
- Presented with many examples relevant to a task and it finds statistical structure in these examples which eventually allows the system to come up with rules for automating the task.
- Implicitly Rule learning/ Pattern learning
- Error is identified by calculating the difference between actual and predicted value.
- What is Pattern? Calculation done on input to get the output.

Example:

Input number Output number

2= 4

3=6

4=8

5=?

Deep learning:

- Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost.
-

What is Learning?

- We learn something in certain ways.
- How to humans learn: Remember, generalize and keep adapting to changing things
- We will incorporate these things into machines

Use cases:

- Adaptive behavior is incorporated by Artificial intelligence
- Remember and Generalize – ML
- Remember and Generalize – DL

Difference between ML and DL?

Machine learning is about computers being able to think and act with less human intervention; deep learning is about computers learning to think using structures modeled on the human brain. Machine learning requires less computing power; deep learning typically needs less ongoing human intervention.

ML Use cases:

- Spam filters
- Computer games
- Voice recognition
- Algorithm to decide whether bank will give loan or not

The need for ML?

- Making Data-Driven Decisions.
- Efficiency and Scale.
- Learning specific patterns from the data.

Application development:

- Traditional approaches.
- We need to explicitly mention the logic/ Rules to get the output.

Why do we need the ML for it?

Situation where the logic or rules are complex. If we increase the complexity of the problem, then we require ML to solve it.

Input number Output number

21234=435565

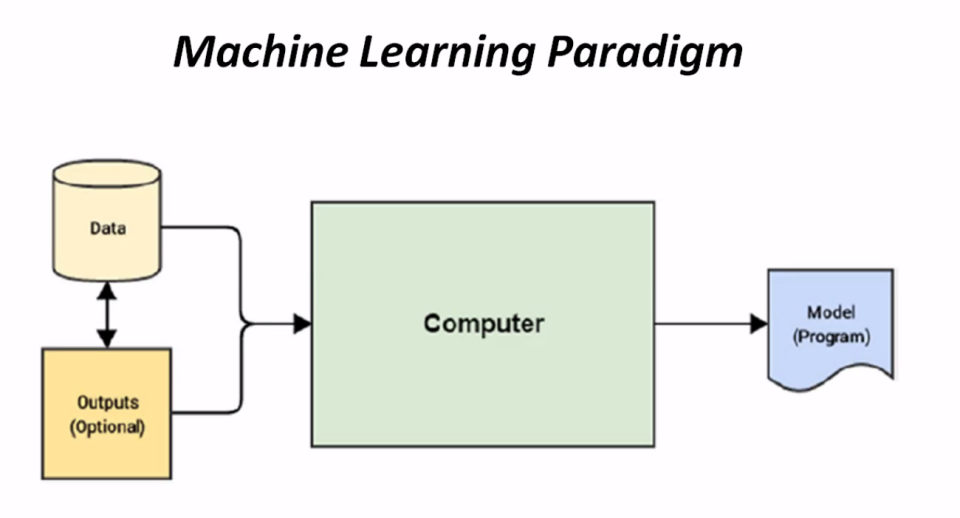
23454=435455

234=?

The simpler the better.

ML Paradigm:

Data + Output -> Computer -> Model (Program)



Traditional programming Paradigm:

Data -> Computer - Model (Program) -> Output

Why make machine learn?

Real world problem and business cases – Complex

- Lack of human expertise
- Scenarios and behavior can keep changing over time. Example: Infrastructe, network connectivity.
- Extremely difficult to explain or translate expertise into computational tasks.
- Address domain specific cases huge volumes of data with too many complex conditions and constraints.

Typical Machine learning tasks:

- Classification or categorization. Example: Identify dog, cat or any other classes

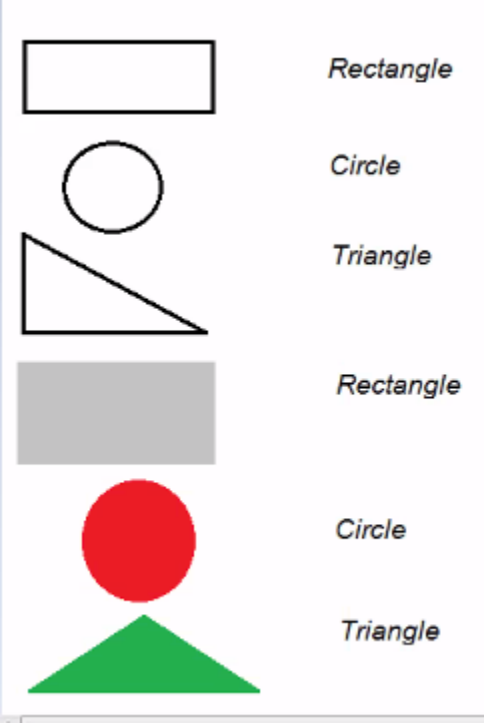
- Regression – Prediction of any number. Example: Salary prediction, housing prices prediction.
- Anomaly detection – identify unusual patterns. Examples: Fraud detection of credit card
- Translation – Feed data belonging to a specific language and translating it to other language.
- Clustering or grouping – To build categories. Example: solution to rubric cube
- Transcriptions – unstructured data like media, image, audio. Examples: Image to text translation.

Types of Machine learning:

- Supervised
- Unsupervised

Supervised learning:

Feed input and output data into the machine. Giving samples of data and then helping it to do the task.

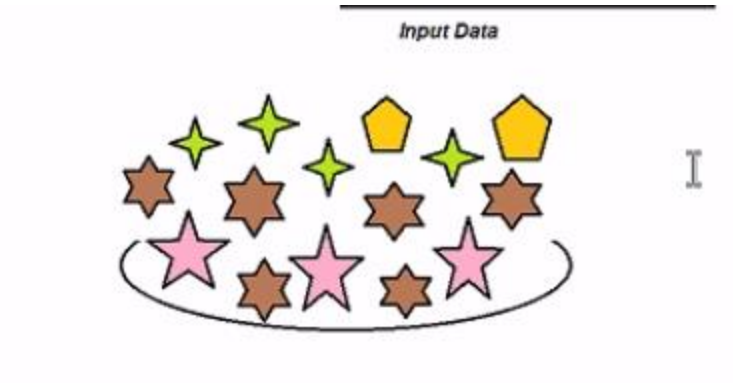


Unsupervised Learning:

Only based on the input data. Output data won't be provided.

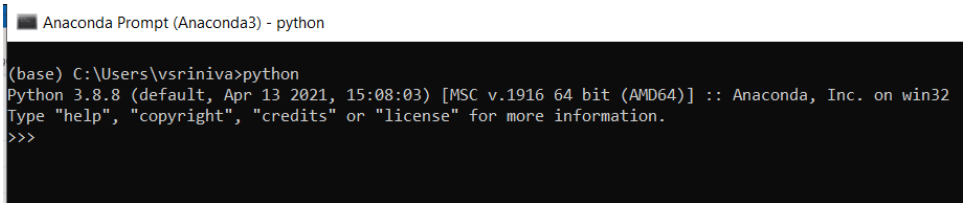
Characteristic details of the data: Dimensions, size and color

Learning patterns among the data based on the similarities



Check the anaconda python version:

It must be equal to or above 3.6 version.



Machine Learning with Python

Why Python?

- Its open source
- Easy to learn
- Interactive Environment
- Python has rich set of libraries that supports data science implementation.

Anaconda:

Python data science repository

IDE:

Supports the python packages

1) Jupyter Notebooks - Data exploration - Interactive Environment

2) Spyder – Algorithm Implementation – Scripting editor

3) Pycharm

4) Visual Code

5) Intellij

6) Eclipse

First two is the popular one. It has got interactive environment and scripting editor.

Jupyter Notebook:

Step 1: Create a Folder named “MLPractice” at any location of your choice

Step 2: WinOS

Start Anaconda Prompt

> **jupyter notebook --notebook-dir=”C:\username\desktop\MLPractice”**

```
(base) C:\Users\vsriniva>jupyter notebook --notebook-dir="C:\Users\vsriniva\Desktop\Data_science_training\ML_practice"
[I 11:37:39.902 NotebookApp] The port 8888 is already in use, trying another port.
[I 11:37:39.906 NotebookApp] The port 8889 is already in use, trying another port.
[W 2022-03-07 11:37:43.587 LabApp] 'notebook_dir' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release
[W 2022-03-07 11:37:43.587 LabApp] 'password' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[W 2022-03-07 11:37:43.589 LabApp] 'password' has moved from NotebookApp to ServerApp. This config will be passed to ServerApp. Be sure to update your config before our next release.
[I 2022-03-07 11:37:43.603 LabApp] JupyterLab extension loaded from C:\Users\vsriniva\Anaconda3\lib\site-packages\jupyterlab
[I 2022-03-07 11:37:43.603 LabApp] JupyterLab application directory is C:\Users\vsriniva\Anaconda3\share\jupyter\lab
[I 11:37:43.614 NotebookApp] Serving notebooks from local directory: C:\Users\vsriniva\Desktop\Data_science_training\ML_practice
[I 11:37:43.614 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 11:37:43.614 NotebookApp] http://localhost:8890/
[I 11:37:43.615 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
)_Lib: debug printing for files [.*] and level [100] is turned on
)_Lib: debug printing for files [.*] and level [200] is turned on
)_Lib: debug printing for files [.*] and level [300] is turned on
:1752:.../engine/vf_shex/vf_shex.cpp(91): INFO: DllCanUnloadNow returned S_OK.
[I 11:37:46.816 NotebookApp] 302 GET /tree (::1) 1.000000ms
[I 11:37:50.112 NotebookApp] 302 POST /login?next=%2Ftree (::1) 122.000000ms
[I 11:37:59.078 NotebookApp] Creating new notebook in
[I 11:38:04.242 NotebookApp] Kernel started: 056d406e-741c-481b-b02b-65d6688d5b78, name: python3
```

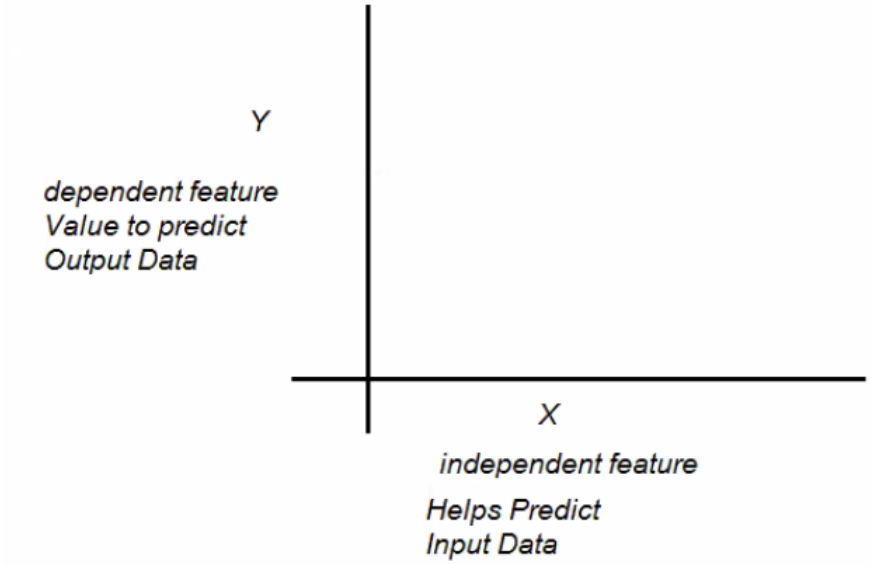
Practical session (Hands on):

Python basics

Python Packages for Machine learning

Understanding the X and Y variables:

Columns Feature / Variable
Rows Observation



independent feature Helps Predict Input Data	dependent feature Value to predict Output Data
X	Y
YrsExp	Salary
YrsExp,Skills,Edu,City	Salary
Investment, Revenue,tax,state	Profit
Age,Income,City	Customer Response

Packages and Modules:

Modules:

- Modules in Python are simply Python files with a .py extension
- The name of the module will be the name of the file.

- A Python module can have a set of functions, classes or variables defined and implemented.
- These are prebuilt.

Example:

Module color (color.py)

Function red()

Function blue()

Function green()

We will see how to import the module and use it.

import color

Color.red()

Color.green()

OR

from color import red, blue

From color import *

Packages:

- Packages are namespaces which contain multiple packages and module themselves. They are simply directories.
- We create a directory “drawing” include module in it: color, line, rectangle, square and circle
- To use line module from drawing package

Import drawing.line

From drawing import circle

Import matplotlib.pyplot as plt

From matplotlib import pyplot as plt2

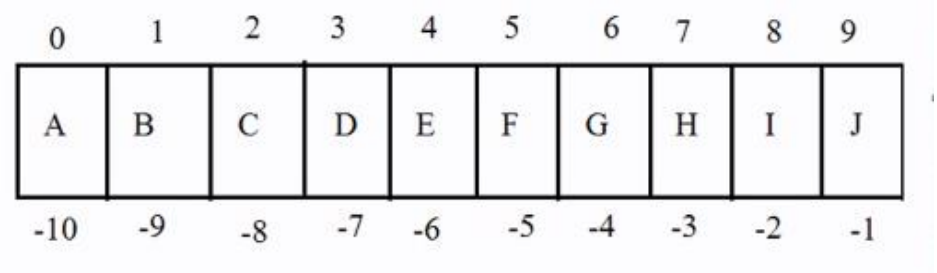
To Install a new packages:

Conda install <package_name>

Or

pip install <package_name>

Slicing or traversing the array:



Syntax: letters[start: stop: step]

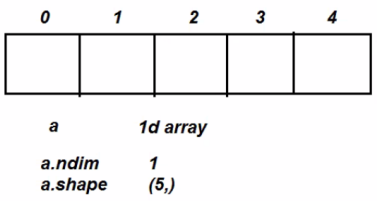
Positive direction: letters[start: stop: 1]

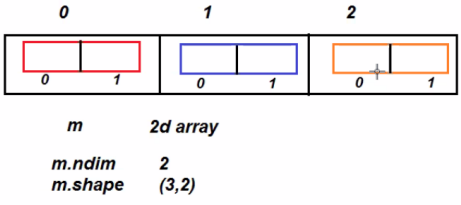
Negative direction: letters[start: stop: -1]

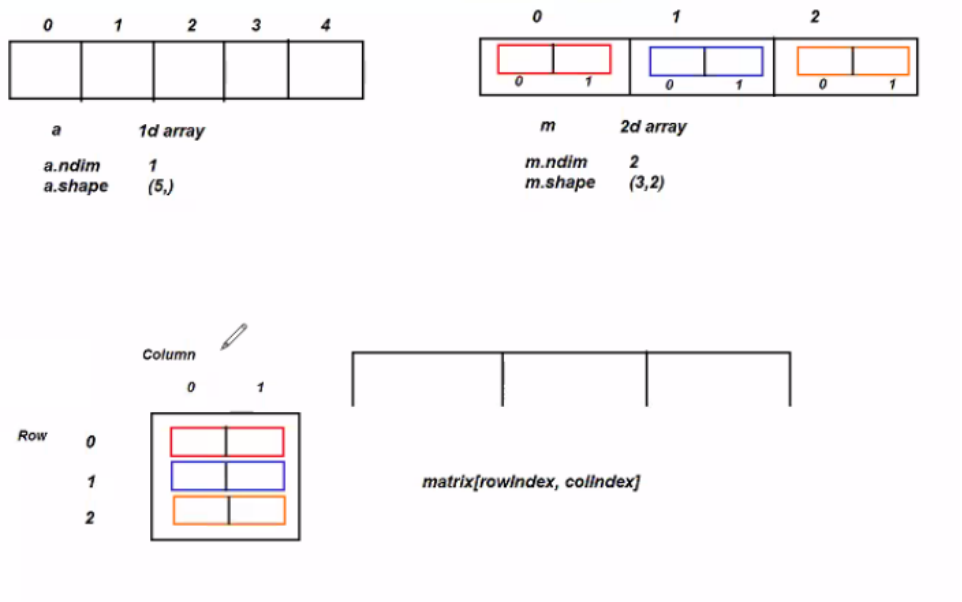
Difference between array and list:

Arrays	List
Homogeneous type storage	mixe datatype storage
Contigious mem. Single segment allocation in mem	non-Contigious mem. multiple segment allocation
Parallel execution	non-parallel execution
numy integrates C++ code in python and hence the c-compiler embedded interpretes it fast	no C++ code integrated
Uses vectorization algorithm no loops needed for oper.	Loop overheads for operations

2 Dimensional arrays:






0 1 2 3 4
a 1d array
a.ndim 1
a.shape (5,)
0 1 2
0 1 0 1 0 1
m 2d array
m.ndim 2
m.shape (3,2)
Column
0 1
Row
0
1
2
matrix[rowIndex, colIndex]

Data science Training – Day 2

Pandas:

Pandas is a Dataframe. It resembles Excel sheets.

Panel Sheets == Panel Data == Pandas

Popularly used for data cleaning, data exploration and map to most of the data sources like delimited files, JSON files and DB etc.

We have None and Nan – unknown value. There is no “Null” value considered in python. For this we need to perform data cleaning operations.

Statistics:

Understanding the information is essential to get the things better and make right choices. Choose right option and deal with things.

We will be able to describe the information using statistics.

Descriptive Statistics:

The basic descriptive statistics to give us an idea on the variables and their distributions

Permit the analyst to describe many pieces of data with few indices.

Central Tendencies:

Mean/ Average:

- Generalization capabilities – without seeing the datapoints, we will know where the data lies.
- Mean () function is used to calculate average
- Mean is not a good measure in presence of outliers

Median:

- Sort the data values in ascending or descending order. Then take up the mid value.
- True mid-point value

Median

- Mean is not a good measure in presence of outliers
- For example Consider below data vector
  - 1.5,1.7,1.9,0.8,0.8,1.2,1.9,1.4, 9 , 0.7 , 1.1
- 90% of the above values are less than 2, but the mean of above vector is 2
- There is an unusual value in the above data vector i.e 9
- It is also known as outlier.
- Mean is not the true middle value in presence of outliers. Mean is very much effected by the outliers.
- We use median, the true middle value in such cases
- Sort the data either in ascending or descending order

Median

1.5		0.7
1.7		0.8
1.9		0.8
0.8		1.1
0.8		1.2
1.2		1.4
1.9		1.5
1.4		1.7
9		1.9
0.7		1.9
1.1		9

- Mean of the data is 2
- Median of the data is 1.4
- Even if we have the outlier as 90, we will have the same median
- Median is a positional measure, it doesn't really depend on outliers
- When there are no outliers then mean and median will be nearly equal
- When mean is not equal to median it gives us an idea on presence of outliers in the data

If mean and median is not close to each other, then it indicates presence of outliers.

Mean and Median

Import “Census Income Data/Income_data.csv”

```
#Mean and Median on python
gain_mean=Income["capital-gain"].mean()
gain_mean

gain_median=Income["capital-gain"].median()
gain_median
```

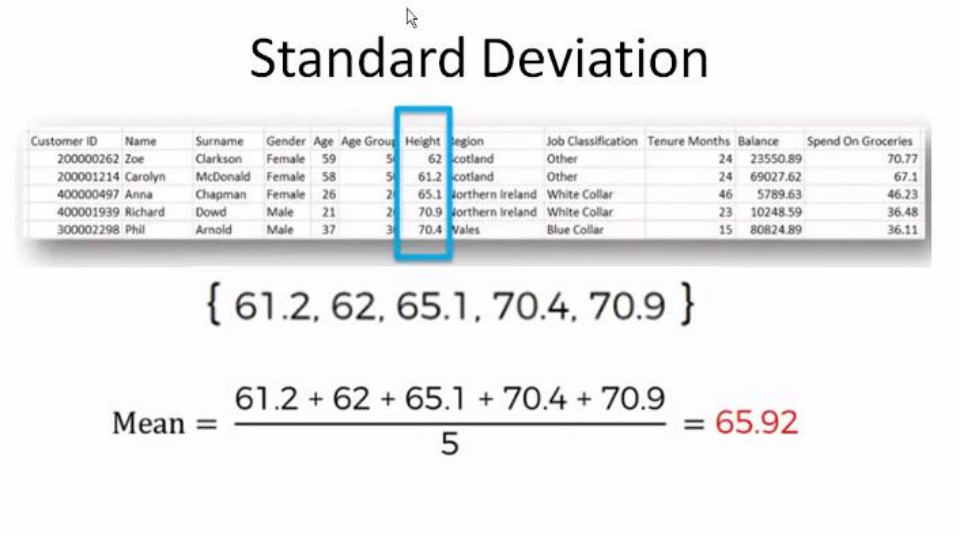
Mean is far away from median. Looks like there are outliers, we need to look at percentiles and box plot.

Mode:

Most popular value.

Most popularly we use mean and median.

Standard Deviation:



Difference between mean and each value, will give the idea of how similarity they are. Some differences can be positive and negative. You can square and sum it. Divide by the total.

Variance:

- Tells the variation in the dataset.
- No units
- Large the variance = more are the values are different from each other
- Less variance = similar values

Standard deviation:

- Square root of variance. Has units.
- Both variance and standard deviation conveys the same thing.

$$\{ 61.2, 62, 65.1, 70.4, 70.9 \}$$

$$\mu \quad \text{Mean} = \frac{61.2 + 62 + 65.1 + 70.4 + 70.9}{5} = 65.92 \text{ meters}$$

$$\text{Variance} = \frac{(61.2 - 65.92)^2 + (62 - 65.92)^2 + (65.1 - 65.92)^2 + (70.4 - 65.92)^2 + (70.9 - 65.92)^2}{5}$$

$$\text{Variance} = \sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N} = 16.64 \text{ no units}$$

$$\text{Std. Dev.} = \sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} = 4.08$$

Dispersion:

Dispersion

- Just knowing the central tendency is not enough.
- Two variables might have same mean, but they might be very different.
- Look at these two variables. Profit details of two companies A & B for last 14 Quarters in MMs

															Mean
Company A	43	44	0	25	20	35	-8	13	-10	-8	32	11	-8	21	15
Company B	17	15	12	17	15	18	12	15	12	13	18	18	14	14	15

- Though the average profit is 15 in both the cases
- Company B has performed consistently than company A.
- There was even loses for company A
- Measures of dispersion become very vital in such cases

Only looking at the mean and judge the dataset is similar, that is wrong!

You can't narrow down with help of mean and median.

Take variance and standard deviation.

Variance and Standard deviation

- Dispersion is the quantification of deviation of each point from the mean value.
- Variance is average of squared distances of each point from the mean
- Variance is a fairly good measure of dispersion.
- Variance in profit for company A is 352 and Company B is 4.9

Value	Value-Mean	(Value-Mean)^2
43	28	784
44	29	841
0	-15	225
25	10	100
20	5	25
35	20	400
-8	-23	529
13	-2	4
-10	-25	625
-8	-23	529
32	17	289
11	-4	16
-8	-23	529
21	6	36
15.0		352

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

Value	Value-Mean	(Value-Mean)^2
17	2	4
15	0	0
12	-3	9
17	2	4
15	0	0
18	3	9
12	-3	9
15	0	0
12	-3	9
13	-2	4
18	3	9
18	3	9
14	-1	1
14	-1	1
15.0		4.9

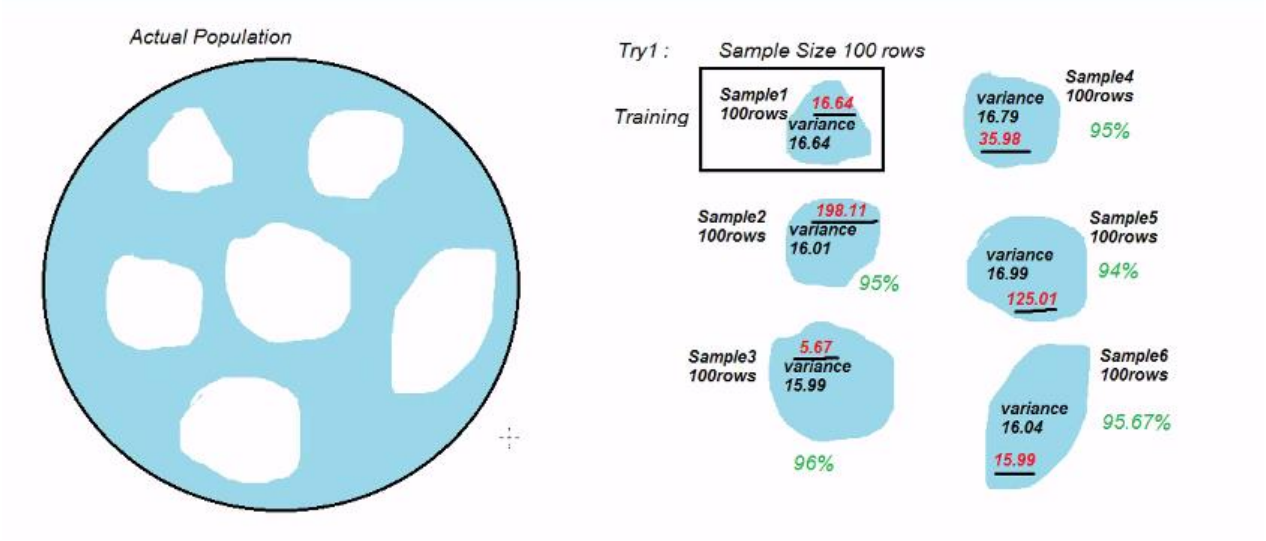
When we are going to give the data into the machine, we need to train it and then test it.

Training set 10 20 30 40 50 60

Testing set 11.23 21.34 35.67 42.11 56.77 59.99

You can see Testing set will be close to the training set.

To build generalization, we will pick up the samples and perform the operations.



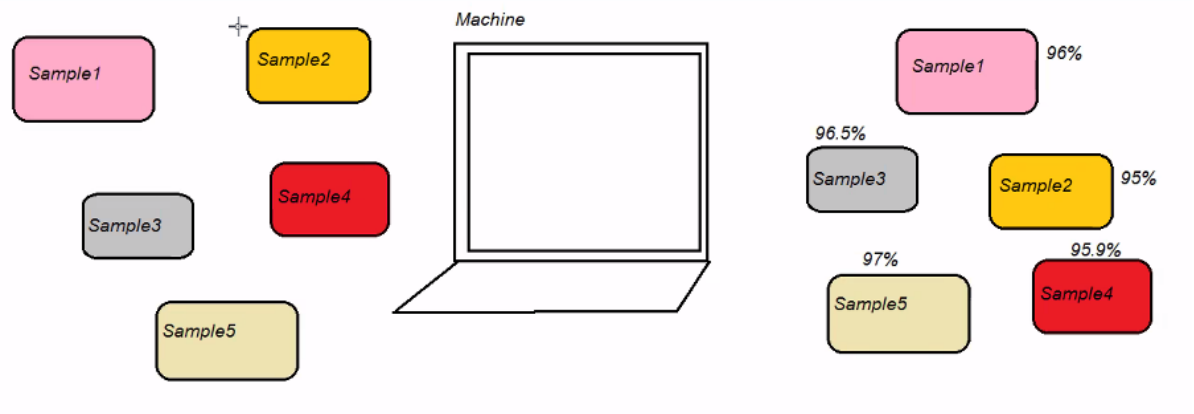
If there is a stable variations, then we consider it has good sample.

Generalization capability is the main thing we focus from variation and standard deviation

Similar variations:

Good fitting!

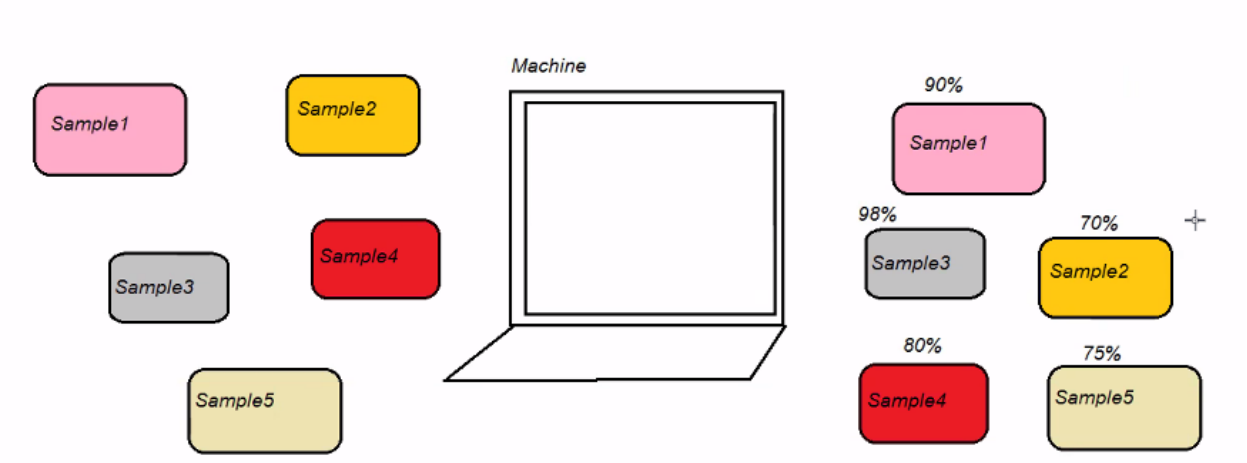
If the samples, we pick up is good and gives a better generalization capability.



It has got enough visibility to all the samples.

Huge variations:

Overfitting!



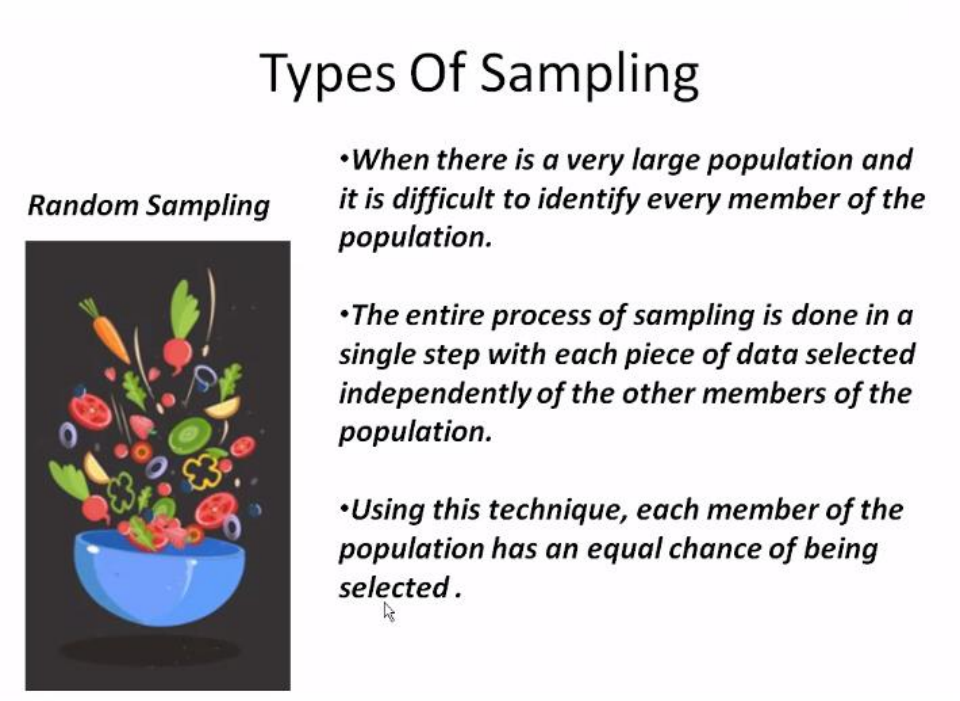
Training Set	10k rows from actual population if 100k rows
Testing Set1	1000 rows from actual population if 100k rows
Testing Set2	1000 rows from actual population if 100k rows
Testing Set3	1000 rows from actual population if 100k rows
Testing Set4	1000 rows from actual population if 100k rows

Sampling can be with or without replacement.

Deciding how much we must give as samples – it's an art. It's all about trial and error!

Types of Sampling:

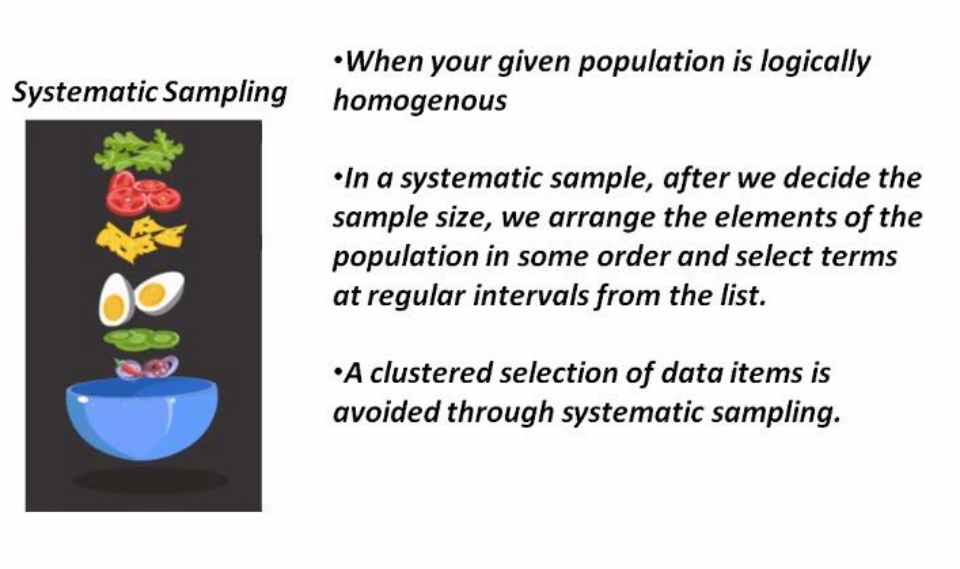
Random sampling:



Values are similar inside the dataset and variation is too similar, then random sampling is good.

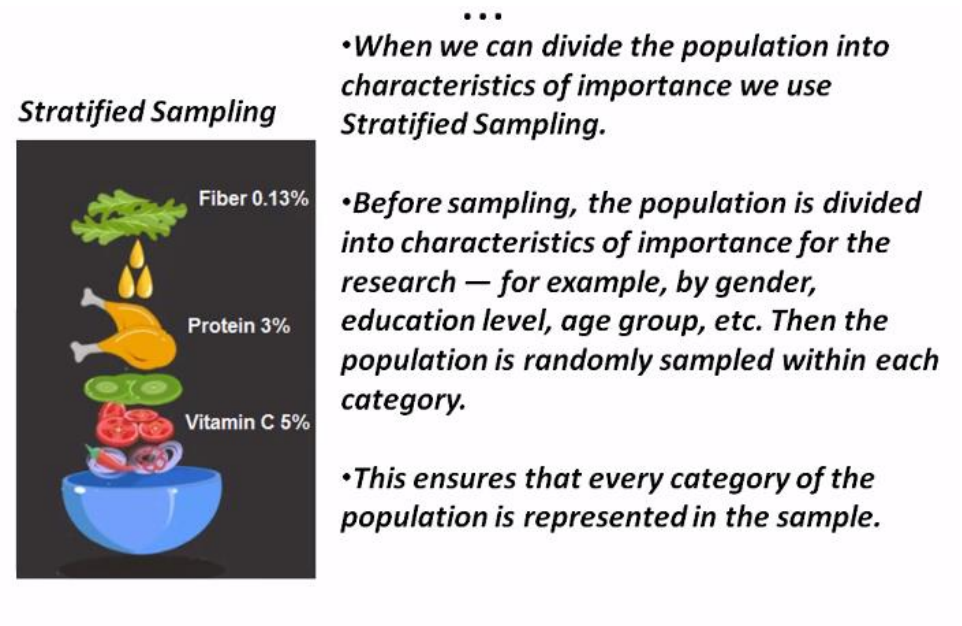
In random sampling, we face clustered selection.

Systematic Sampling:



When the values and variation is not similar, then systematic sampling is good.

Stratified Sampling:



Give importance to each category. Ensure all categories are picked up in the samples. More population more probability of the dataset. Less prominent cities, then less probability of the data.

Testing samples are randomized sampling. Training samples can be systematic and stratified sampling.

Sampling of Dataframe implementation:

https://towardsdatascience.com/how-to-sample-a-dataframe-in-python-pandas-d18a3187139b

Types of values/ features:

1) Discrete

2) Continuous

Discrete:

Concrete boundaries – well defined boundaries

Example:

Year of birth – 2000,2001, 2002, 2003 and so on..

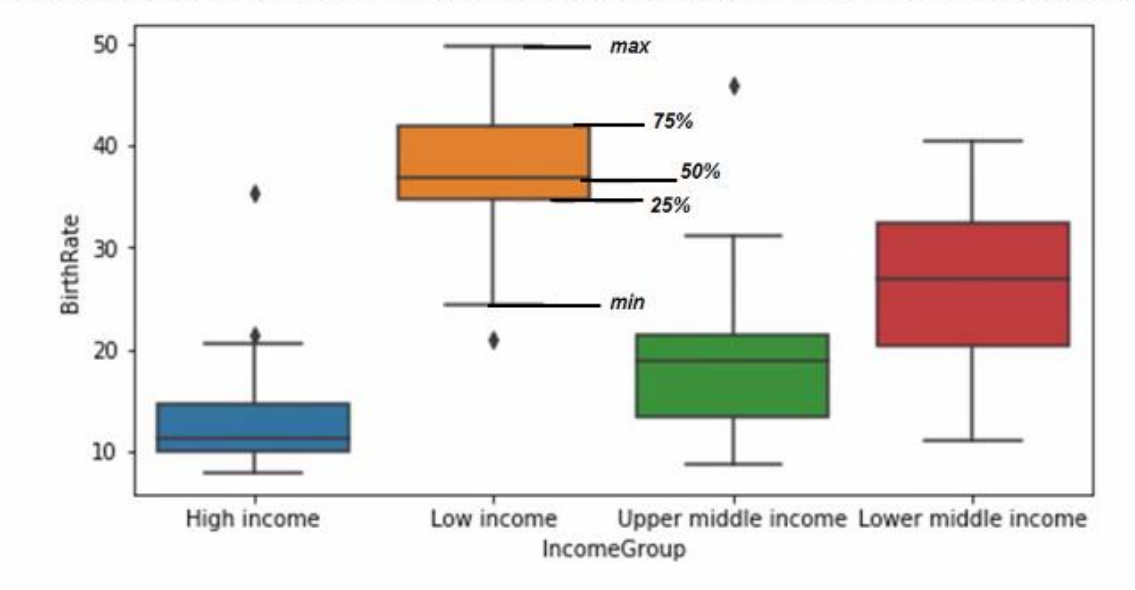
Response – 0 or 1

DeptID - 10 20 30 40 50

Continuous:

Example:

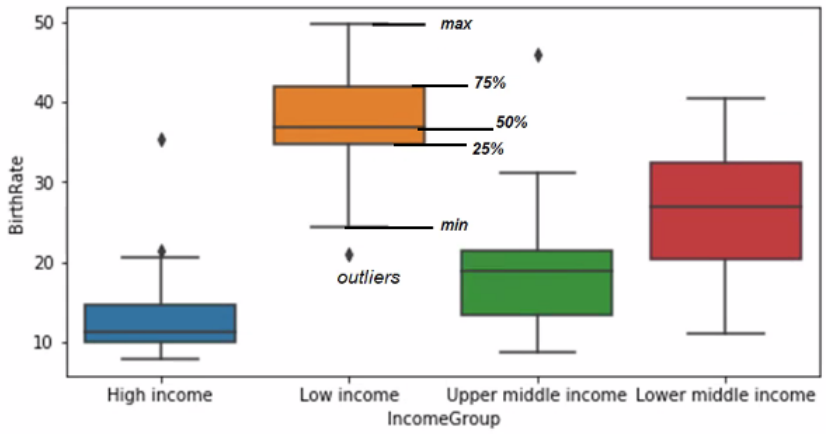
Exact weight of animal in the jungle?? 1Kg ~ 100 Kg



75% *Third Quartile (Q3)*
Upper Quartile

50% *Second Quartile (Q2)*
Middle Quartile
Median

25% *First Quartile (Q1)*
Lower Quartile

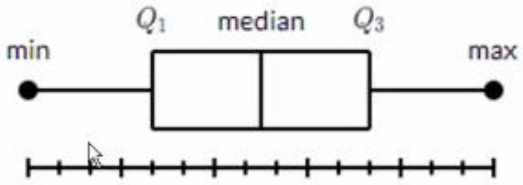


75% *Third Quartile (Q3)*
Upper Quartile

50% *Second Quartile (Q2)*
Middle Quartile
Median

25% *First Quartile (Q1)*
Lower Quartile

Quartiles



Q2 median

Q1 25% first quartile

Q3 third quartile

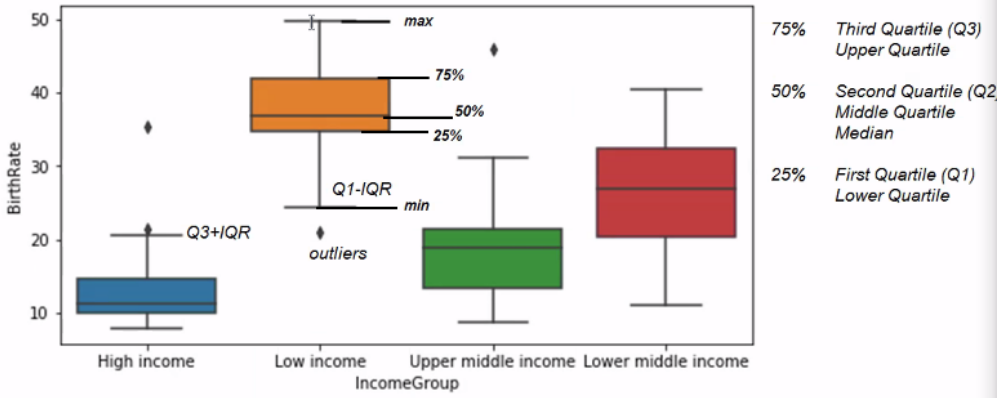
59, 60, 65, 65, 68, 69, 70, 72, 75, 75, 76, 77, 81, 82, 84, 87, 90, 95, 98

min median max

Total 19 values

Calculate the outlier:

	BirthRate
count	67.000000
mean	12.753433
std	4.467886
min	7.900000
25%	10.100000
50%	11.300000
75%	14.545000
max	35.362000



Outlier Calculation

Step 1: Calculate IQR (Inter Quartile Range) variance in Quartile

IQR = (Q3 – Q1) *1.5 = (14.545 - 10.10) * 1.5 = 6.667

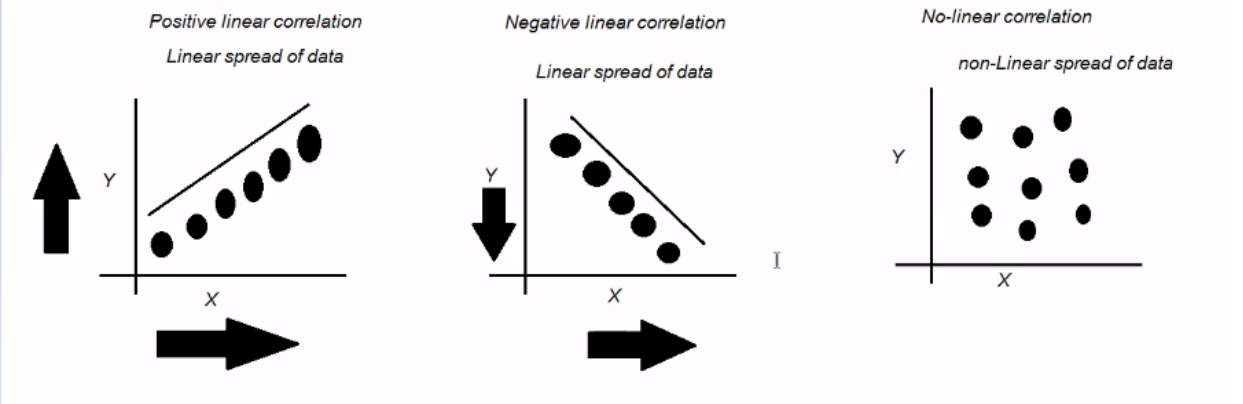
Step 2:

Q3 + IQR = 14.545 + 6.667 = 21.212 # all values above this are outliers

Q1 – IQR = 10.10 - 6.667 = 3.43 # all values above this are outliers

At middle income, you see mean and median are very similar.

Correlations: Measure of linearity between X and Y.



Pearson Coefficients:

Pearson Coefficient

$$Pearsonr = \frac{N * sum(xy) - sum(x) * sum(y)}{sqrt([N * sum(x^2) - sum(x)^2] * [N * sum(y^2) - sum(y)^2])}$$

Using scipy we can find it.

Correlation is calculated using r-value / Pearsonr-value & p-value

Pearsonr-value conveys the percentage of correlation between X & y

p-value conveys the percentage of uncorrelation between X & y

==========

Pearsonr-value (-1 to 1)

Pearsonr-value = 0.95 X & y are 95% correlated
Pearsonr-value = 0.08 X & y are 8% correlated

Pearsonr-value	
0 to < 0.25	No correlation , No relevance between x & y
0.25 to < 0.50	Negligible correlation / relevance between x & y
0.50 to < 0.75	Moderate correlation / relevance between x & y
> 0.75	Very Strong correlation / relevance between x & y

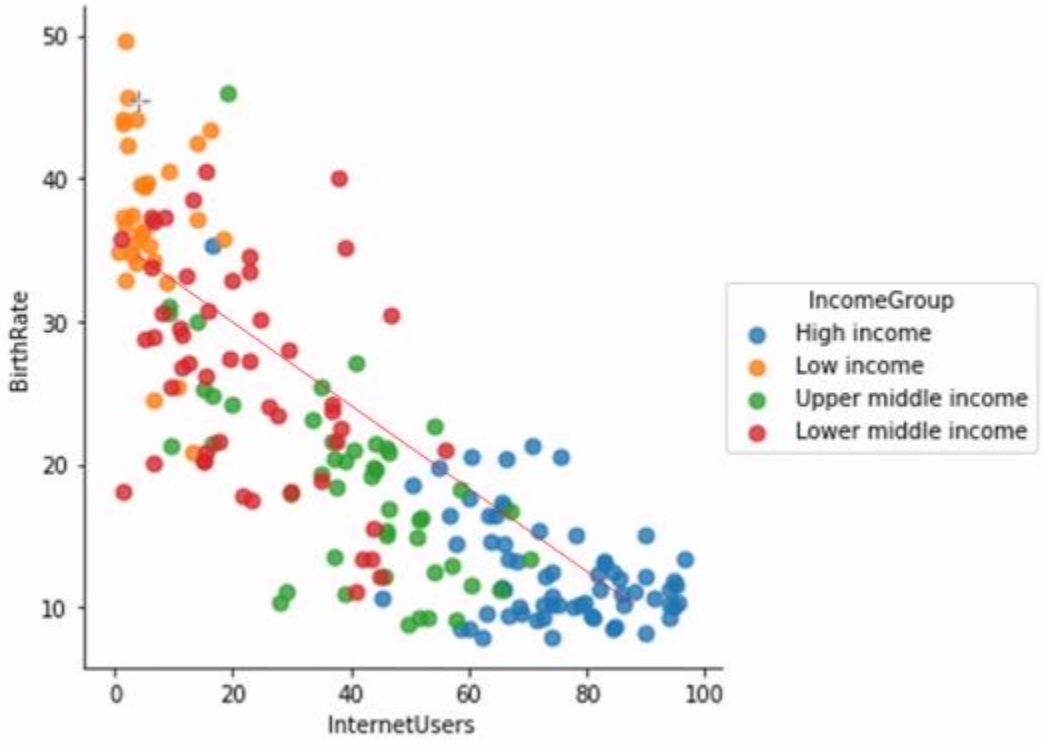
==========

p-value

p-value=0.98 X & y are 98% uncorrelated
p-value=0.02 X & y are 2% uncorrelated

X features with p-value above 0.05 highly uncorrelated , we ignore or avoid choosing those X features

X features with p-value below 0.05 highly correlated , we consider choosing those X features



X features 10 in number to predict Y

8 features have correlation in moderatae to very strong
2 features have neglible to no-correlation
1 x feature pearsonr- 45%
2nd X feature pearsonr- 12%

Flow of data in Data science

Data Collection

Data Cleaning

Data exploration: Calculate mean median, mode, std deviation and variance

Data Preprocessing: Whatever data you are cleaning, you will apply preprocessing techniques on it. Examples: Encoding, dummy variable creation, train-test split, scaling of values.

It is a value-added step. This will improve the quality of the data.

Model implementation:

Supervised learning – Accept both input and output data – Remember & Generalize

Regression – Value to predict is Continuous

Linear Regression

Polynomial Regression

Decision Tree Regression

Random Forest Regression

Classification – Value to predict is Discrete

Logistic Regression

Support Vector Machine

Decision Tree classifier

Random Forest Classifier

Naïve Bayes

Unsupervised learning – Accept only input data – Remember & Generalize

Clustering

K-means clustering

Reinforcement learning – Accept data on the go – online learning – Adaptive

Upper Confidence Bound

Thompsan Sampling

Deep learning – Accept input data and output data – Remember & Generalize

Artificial Neural Network

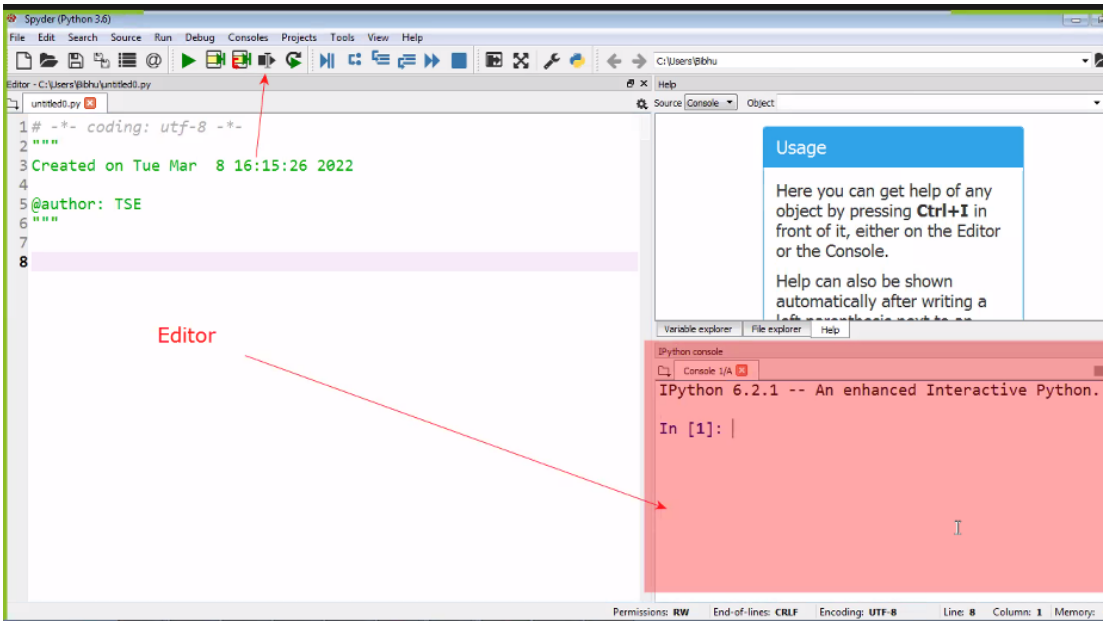
Convolutional Neural Network

Recurrent-Neural Network – LSTM (Long short-term memory)

Natural language processing – Preprocessing text to number

Regression Problem:

Open Spyder:



Continuous value Prediction:

Continuous Value Prediction, measure of model performance is Error

YrsExp	ActualSal	PredictedSal	Error = diff(acutal,predicted) = avg((diff(acutal,predicted) ^2))
1	10000	10500	-500
2.5	12000	12000	0
3	15000	20000	-5000
3.5	17000	11000	6000
4	20000	19000	1000
4.5	25000	27000	-2000
			----------Error

Less the error => Better is the algorithm => Target

Error is the parameter of judgement.

Performance measure is error – sum of error divided by the total number of values.

Discrete Value prediction:

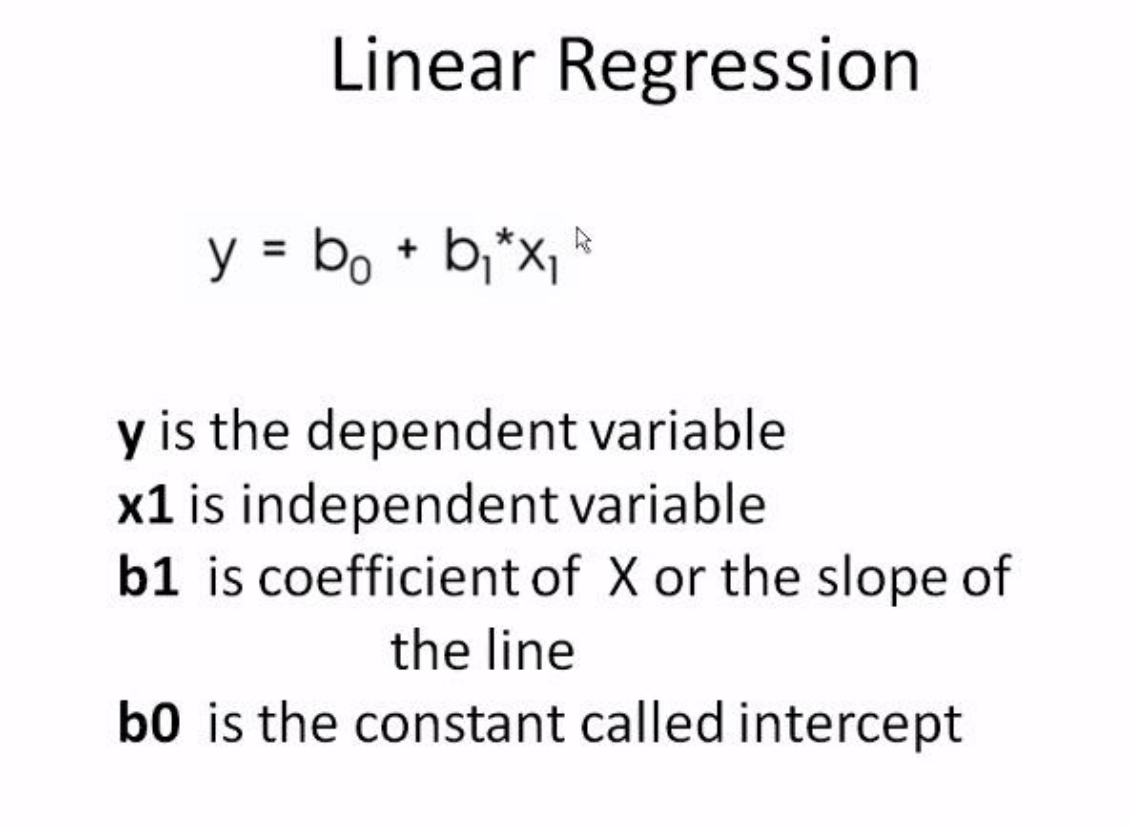
Discrete Value Prediction, Measure of model performance is Accuracy percentage

Age	ActualResponse	PredictedResponse	
			6/9*100=66% accuracy score
20	0	0	
30	1	1	
21	0	0	
35	1	0	
40	1	1	
45	1	1	
50	0	1	
18	0	0	
19	1	0	

Here, you must count the total number of correct prediction and divided by the total predictions.

Accuracy is the key measure of judgement.

Linear regression is to be chosen when the data is linear data distribution / moderate or strong correlation based on PSNR value. No need to plot every time and check. Based on Peasonr value also we can decide.



Linear Regression:

Y = b0 + b1 * x1

Salary Prediction

*Salary = base Package + Amount * Total Experience*

Base Package are the people with 0 years of experience.

Example:

B0 = 25000

Total experience = 5000

Amount = 2

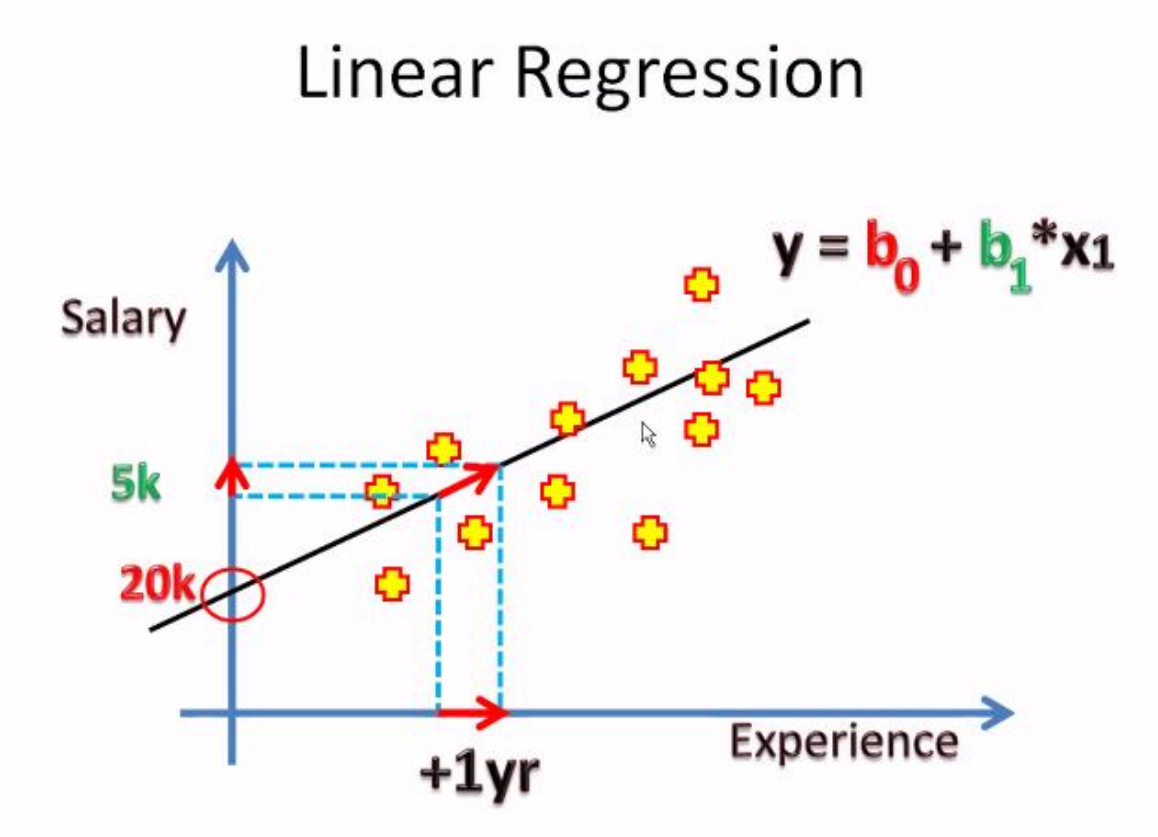
Salary = 35000

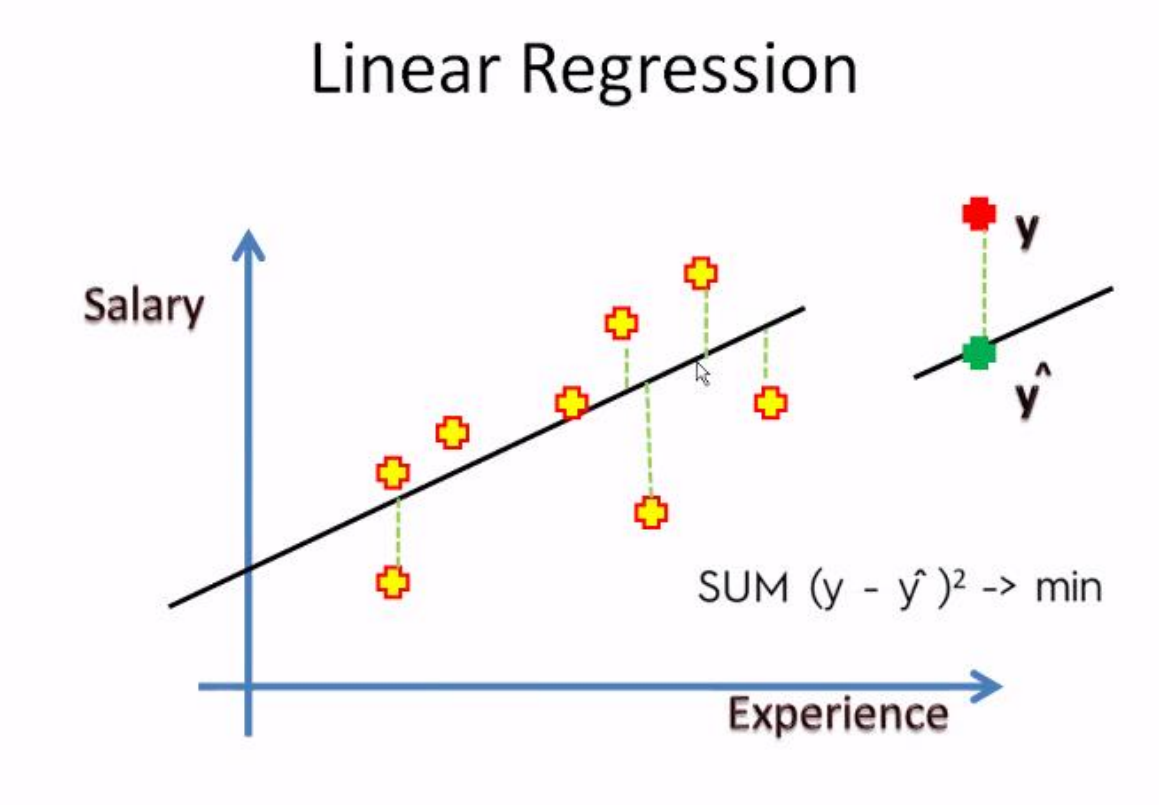
How b0 and b1 is calculated?

$$\frac{sum(y)*sum(x^2) - sum(x)sum(xy)}{n*sum(x^2) - sum(x)^2} \quad b0 = intercept$$

$$\frac{n*sum(xy) - sum(x)*sum(y)}{n*sum(x^2) - sum(x)^2} \quad b1 = slope$$

Linear Regression



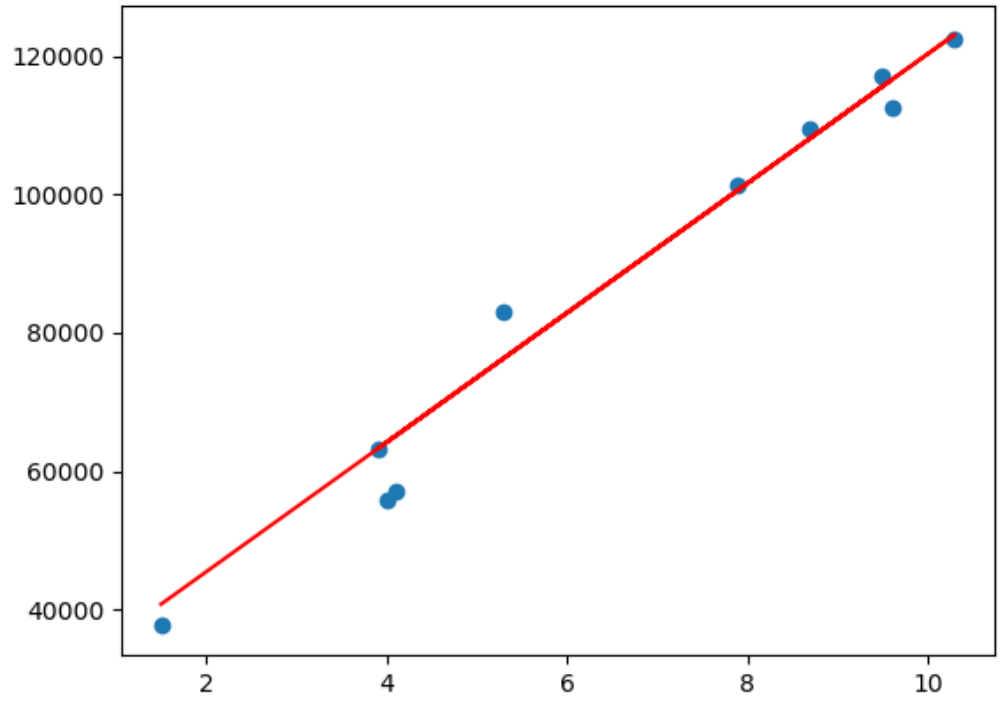


Why is intercept needed?

If there is no intercept, then you won't be able to draw the best fir line. That gets difficult!

Intercept is the grace mark which pushes the slope line to fit the datapoints.

Output of simple regression:



Sample	100 rows			
Random Split	80-20 or 70-30 or 75-25			
Train Data	80%	80 rows	X_train,y_train	YrsExp,Salary
Test Data	20%	20 rows	X_test,y_test	YrsExp,Salary

Machine Getting trained X_train,y_train YrsExp,Salary

Machine Testing

Predict salary for X_test (YrsExp)
PredSalary = y_pred
Actual Salary = y_test

```
In [10]: np.mean((y_test-y_pred)**2)
Out[10]: 21026037.329511296

In [11]: np.sqrt(np.mean((y_test-y_pred)**2))
Out[11]: 4585.415720467588 5
```

Here, we have only one train data and we can have multiple test data.

Test Data1	20%	20 rows	X_test,y_test	YrsExp,Salary	4585
Test Data1	20%	20 rows	X_test,y_test	YrsExp,Salary	4401
Test Data1	20%	20 rows	X_test,y_test	YrsExp,Salary	4500
Test Data1	20%	20 rows	X_test,y_test	YrsExp,Salary	4587
Test Data1	20%	20 rows	X_test,y_test	YrsExp,Salary	4505

Use Case
Client: I want a ML solution for my business

Business Objective / Acceptance criteria
I will accept ML solution if I se improvement in customer satisfaction by 25%
and revenue improvement of 2million

Data : Data cleaning , Data Exploration, Data Preprocessing

Train the Machine

Test the Model => 98% accurate results

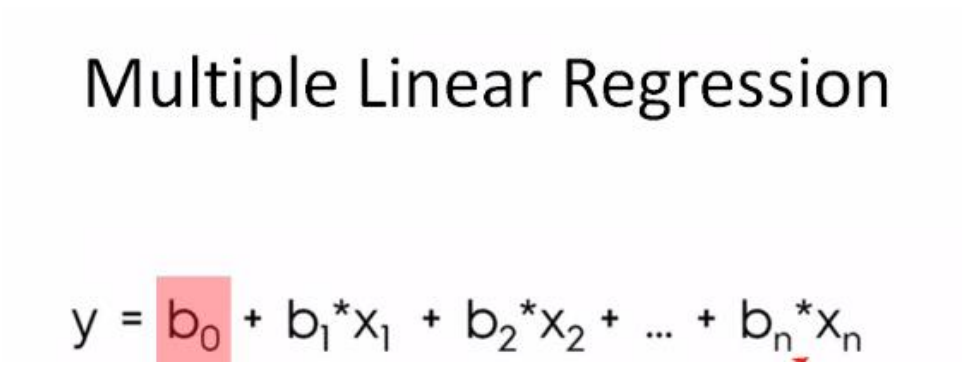
Situation1: Model1 98% accuracy
improvement in customer satisfaction 5% excepted was 25%
revenue improvement of 50k expected was 2million

Situation2: Model2 80% accuracy
improvement in customer satisfaction 35% excepted was 25%
revenue improvement of 5million expected was 2million

Data Science Training - Day 3

Multi-Linear Regression:

A	B	C	D	E
RNDSpend	Administration	MarketingSpend	State	Profit
165349.2	136897.8	471784.1	New York	192262
162597.7	151377.59	443898.53	California	191792
153441.51	101145.55	407934.54	Florida	191050
144372.41	118671.85	383199.62	New York	182902
142107.34	91391.77	366168.42	Florida	166188
131876.9	99814.71	362861.36	New York	156991
134615.46	147198.87	127716.82	California	156123
130298.13	145530.06	323876.68	Florida	155753
120542.52	148718.95	311613.29	New York	152212
123334.88	108679.17	304981.62	California	149760
101913.08	110594.11	229160.95	Florida	146122
100671.96	91790.61	249744.55	California	144259
93863.75	127320.38	249839.44	Florida	141586
91992.39	135495.07	252664.93	California	134307



The dataset won't get fit into the below formula:

$$b_0 = \text{intercept} = \frac{\text{sum}(y)*\text{sum}(x^2) - \text{sum}(x)\text{sum}(xy)}{n*\text{sum}(x^2) - \text{sum}(x)^2}$$

$$b_1 = \text{slope} = \frac{n*\text{sum}(xy) - \text{sum}(x)*\text{sum}(y)}{n*\text{sum}(x^2) - \text{sum}(x)^2}$$

To handle the state column (category), we need encoding. We will convert it to numeric format.

Encoding/dummy variable creation /One hot encoder:

Encoding takes unique values and alphabetically sorted. It will represented each unique value as a columns.

California	d0_C
Florida	d1_F
New York	d2_NY

State	d0_C	d1_F	d2_NY
New York	0	0	1
California	1	0	0
Florida	0	1	0

Machine learns through rows – identifying the pattern instead of columns.

You can remove the dummy variables.

State	d0_C	d1_F	d2_NY
New York	0	0	1
California	1	0	0
Florida	0	1	0

In order to have a balance learning, dummy variable is best suited. We won't use weighted labels.

numOfDummyVar = numOfUniqVal - 1

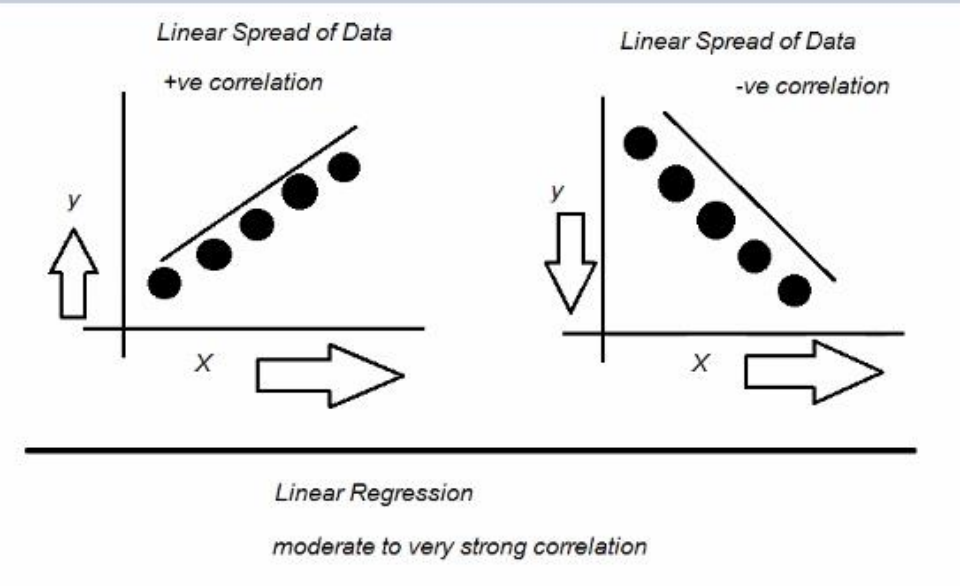
California	d0_C	0
Florida	d1_F	1
New York	d2_NY	2
Texas		3
		4
		5

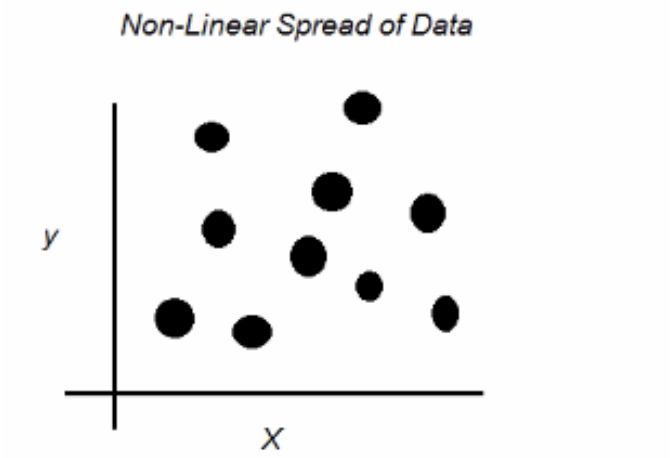
XNew - DataFrame

Index	RNDSpend	Administration	MarketingSpend	State_California	State_Florida	State_New York
0	165349	136898	471784	0	0	1
1	162598	151378	443899	1	0	0
2	153442	101146	407935	0	1	0
3	144372	118672	383200	0	0	1
4	142107	91391.8	366168	0	1	0
5	131877	99814.7	362861	0	0	1
6	134615	147199	127717	1	0	0
7	130298	145530	323877	0	1	0
8	120543	148719	311613	0	0	1
9	123335	108679	304982	1	0	0
10	101913	110594	229161	0	1	0
11	100672	91790.6	249745	1	0	0
12	93863.8	127320	249839	0	1	0
13	91992.4	135495	252665	1	0	0
14	119943	156547	256513	0	1	0
15	114524	122617	261776	0	0	1
16	78013.1	121598	264346	1	0	0
17	94657.2	145078	282574	0	0	1
18	91749.2	114176	294920	0	1	0
19	86419.7	153514	0	0	0	1
20	76253.9	113867	298664	1	0	0
21	78389.5	153773	299737	0	0	1
22	73994.6	122783	303319	0	1	0

XDrop - DataFrame

Index	RNDSpend	Administration	MarketingSpend	State_Florida	State_New York
0	165349	136898	471784	0	1
1	162598	151378	443899	0	0
2	153442	101146	407935	1	0
3	144372	118672	383200	0	1
4	142107	91391.8	366168	1	0
6	134615	147199	127717	0	0
5	131877	99814.7	362861	0	1
7	130298	145530	323877	1	0
9	123335	108679	304982	0	0
8	120543	148719	311613	0	1
14	119943	156547	256513	1	0
15	114524	122617	261776	0	1
10	101913	110594	229161	1	0
11	100672	91790.6	249745	0	0
17	94657.2	145078	282574	0	1





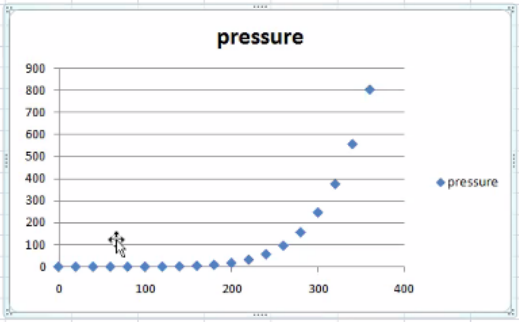
correlation of most of the X features will be in negligible to no-correlation

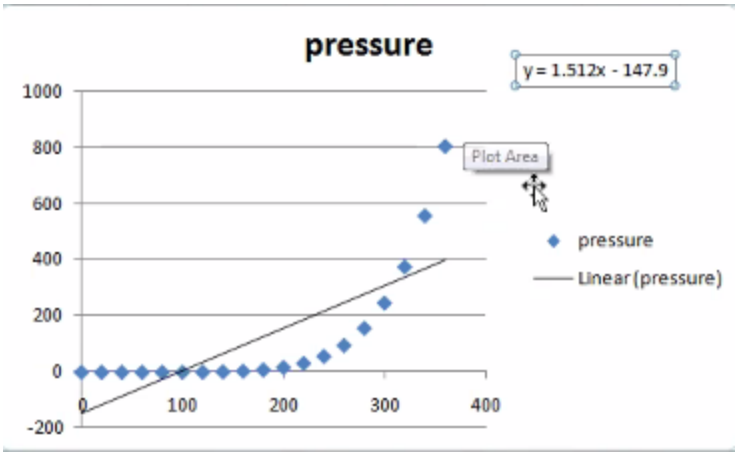
Polynomial Regression
DecisionTree Reg.
RandomForest Reg.

Take the example below:

This is not a linear problem.

tempreture	pressure
0	0.0002
20	0.0012
40	0.006
60	0.03
80	0.09
100	0.27
120	0.75
140	1.85
160	4.2
180	8.8
200	17.3
220	32.1
240	57
260	96
280	157
300	247
320	376
340	558
360	806



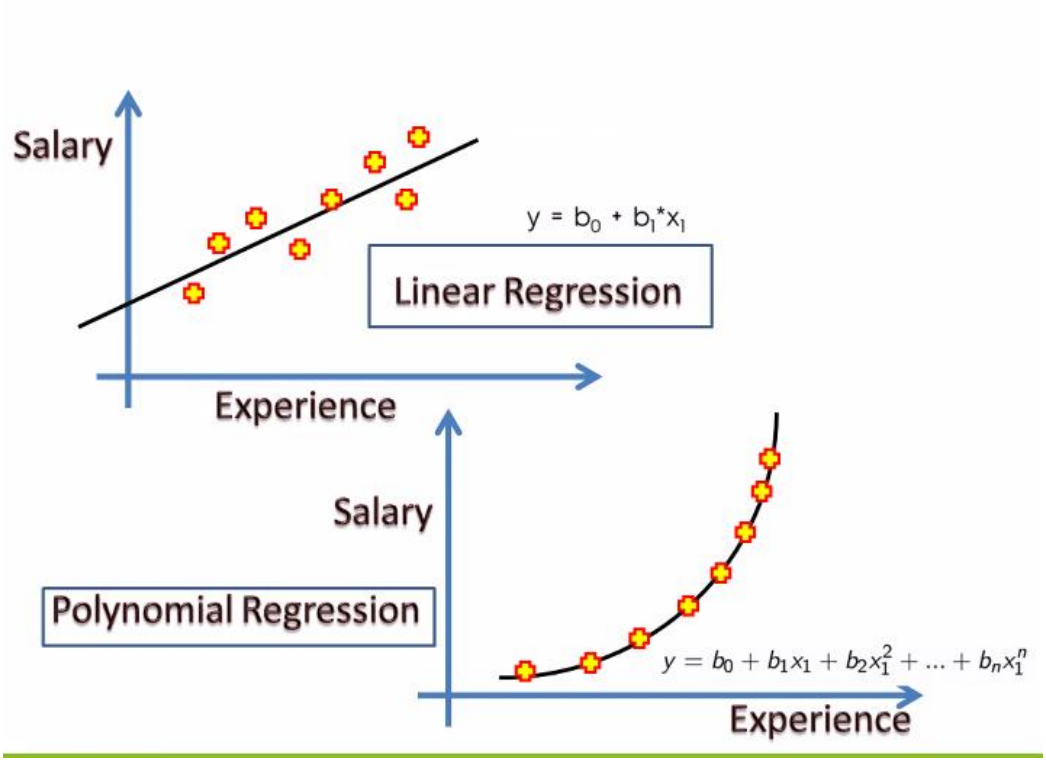


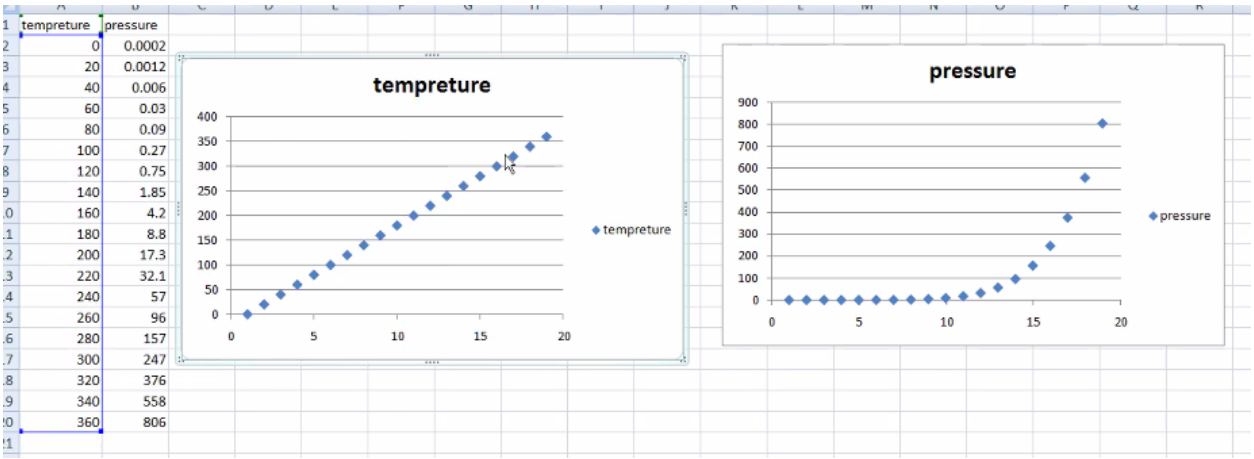
Here linear regression formula falls short. So, in this case you must use polynomial regression

Polynomial Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

By adding degree to each feature, we can solve it. This will curve the best fit line.

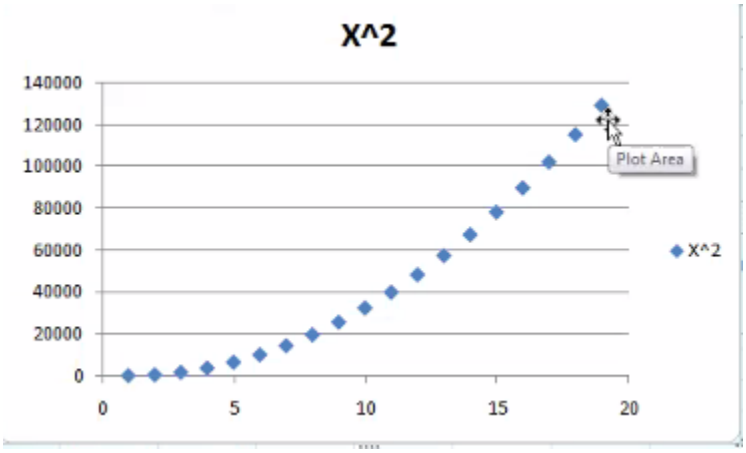


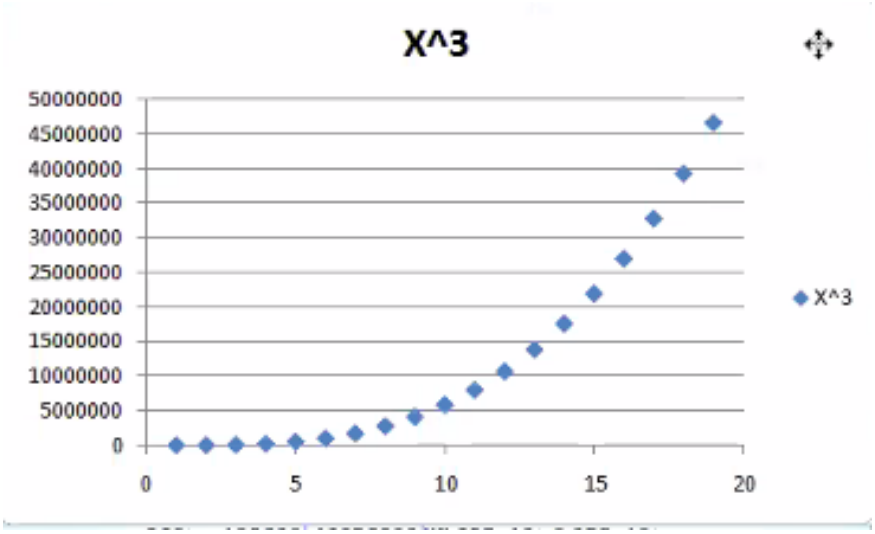


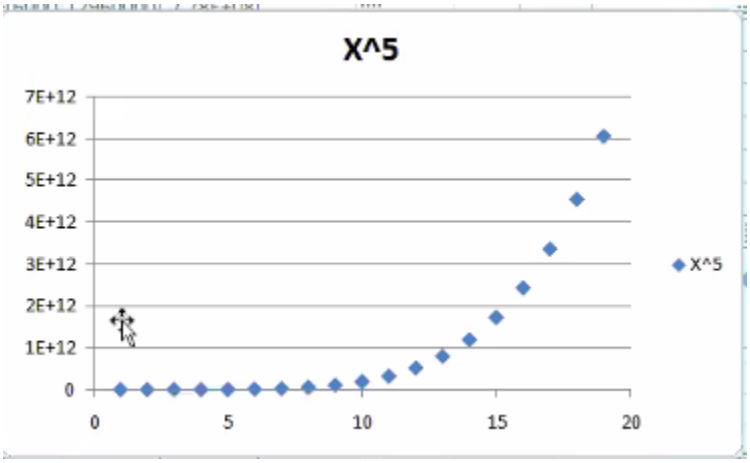
tempreture	pressure
0	0.0002
20	0.0012
40	0.006
60	0.03
80	0.09
100	0.27
120	0.75
140	1.85
160	4.2
180	8.8
200	17.3
220	32.1
240	57
260	96
280	157
300	247
320	376
340	558
360	806

You need to tweet the x value.

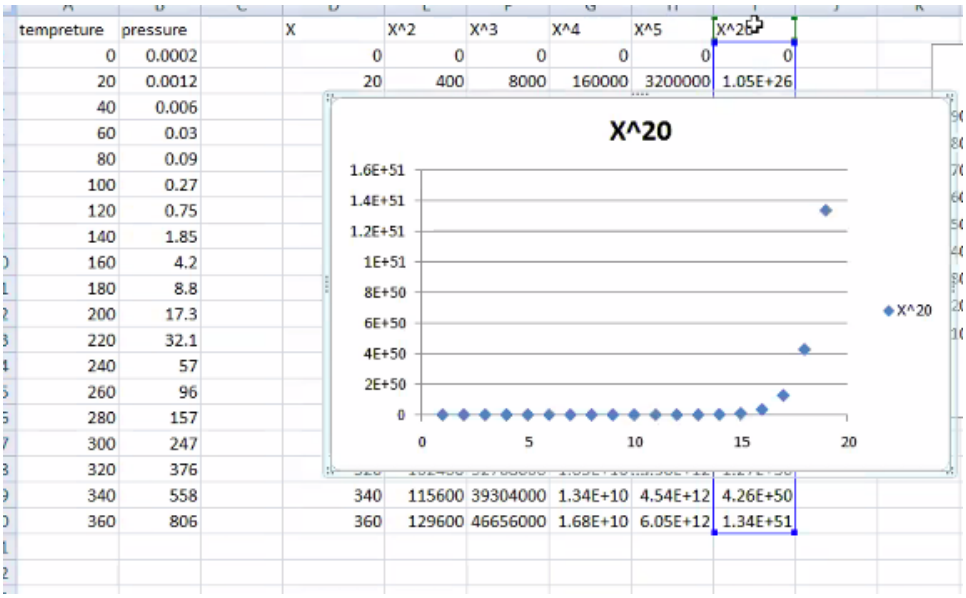
tempreture	pressure		X	X^2	X^3	X^4	X^5
0	0.0002		0	0	0	0	0
20	0.0012		20	400	8000	160000	3200000
40	0.006		40	1600	64000	2560000	1.02E+08
60	0.03		60	3600	216000	12960000	7.78E+08
80	0.09		80	6400	512000	40960000	3.28E+09
100	0.27		100	10000	1000000	1E+08	1E+10
120	0.75		120	14400	1728000	2.07E+08	2.49E+10
140	1.85		140	19600	2744000	3.84E+08	5.38E+10
160	4.2		160	25600	4096000	6.55E+08	1.05E+11
180	8.8		180	32400	5832000	1.05E+09	1.89E+11
200	17.3		200	40000	8000000	1.6E+09	3.2E+11
220	32.1		220	48400	10648000	2.34E+09	5.15E+11
240	57		240	57600	13824000	3.32E+09	7.96E+11
260	96		260	67600	17576000	4.57E+09	1.19E+12
280	157		280	78400	21952000	6.15E+09	1.72E+12
300	247		300	90000	27000000	8.1E+09	2.43E+12
320	376		320	102400	32768000	1.05E+10	3.36E+12
340	558		340	115600	39304000	1.34E+10	4.54E+12
360	806		360	129600	46656000	1.68E+10	6.05E+12





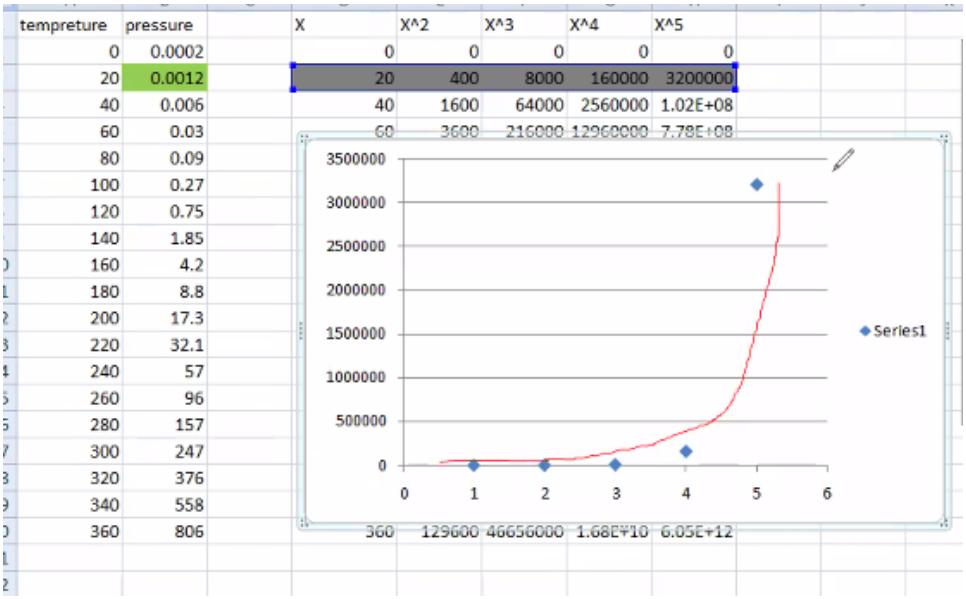


More the degree, better the result. But not too far!



tempreture	pressure	X	X^2	X^3	X^4	X^5	X^26
0	0.0002	0	0	0	0	0	0
20	0.0012	20	400	8000	160000	3200000	1.05E+26
40	0.006						
60	0.03						
80	0.09						
100	0.27						
120	0.75						
140	1.85						
160	4.2						
180	8.8						
200	17.3						
220	32.1						
240	57						
260	96						
280	157						
300	247						
320	376						
340	558	340	115600	39304000	1.34E+10	4.54E+12	4.26E+50
360	806	360	129600	46656000	1.68E+10	6.05E+12	1.34E+51

It’s a trial-and-error method. Until we try with different degree, we won’t know which is best.



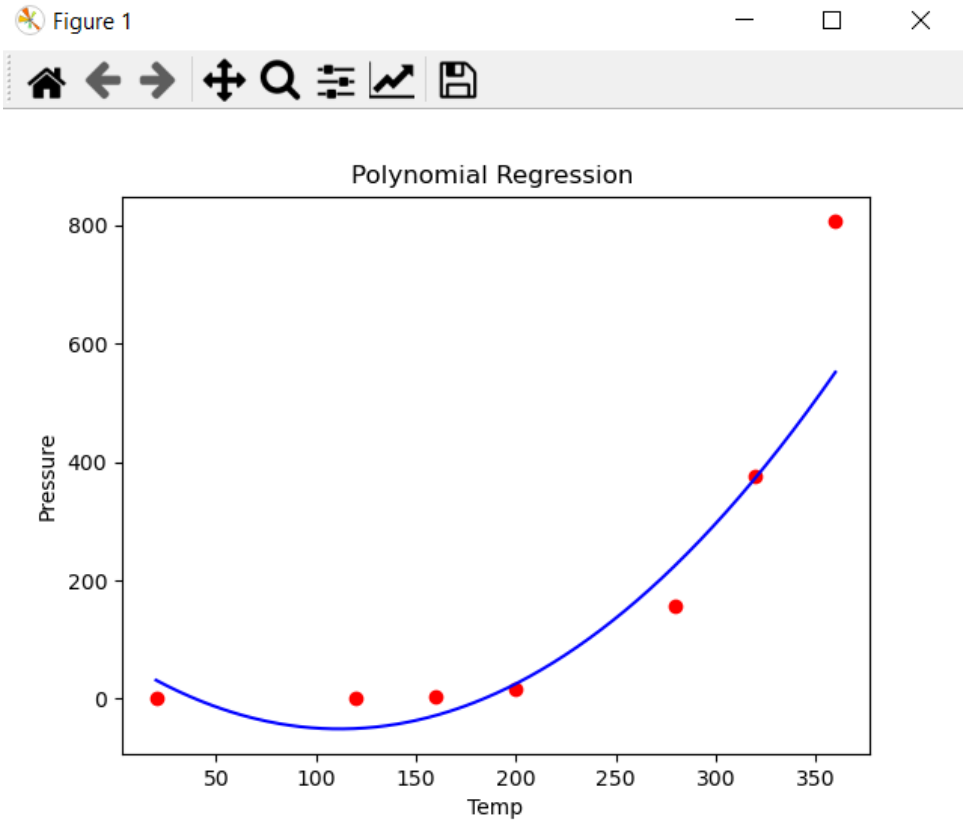
tempreture	pressure	X	X^2	X^3	X^4	X^5
0	0.0002	0	0	0	0	0
20	0.0012	20	400	8000	160000	3200000
40	0.006	40	1600	64000	2560000	1.02E+08
60	0.03	60	3600	216000	12960000	7.78E+08
80	0.09					
100	0.27					
120	0.75					
140	1.85					
160	4.2					
180	8.8					
200	17.3					
220	32.1					
240	57					
260	96					
280	157					
300	247					
320	376					
340	558					
360	806	360	129600	46656000	1.68E+10	6.05E+12

Machine is least interested in column names. Machine is not aware of which column it deals with. It considers it as feature 1,2,3,4, and so on.

tempreture	pressure	x1	x2	x3	x4	x5
0	0.0002	0	0	0	0	0
20	0.0012	20	400	8000	160000	3200000
40	0.006	40	1600	64000	2560000	1.02E+08
60	0.03	60	3600	216000	12960000	7.78E+08
80	0.09	80	6400	512000	40960000	3.28E+09
100	0.27	100	10000	1000000	1E+08	1E+10
120	0.75	120	14400	1728000	2.07E+08	2.49E+10
140	1.85	140	19600	2744000	3.84E+08	5.38E+10
160	4.2	160	25600	4096000	6.55E+08	1.05E+11
180	8.8	180	32400	5832000	1.05E+09	1.89E+11
200	17.3	200	40000	8000000	1.6E+09	3.2E+11
220	32.1	220	48400	10648000	2.34E+09	5.15E+11
240	57	240	57600	13824000	3.32E+09	7.96E+11
260	96	260	67600	17576000	4.57E+09	1.19E+12
280	157	280	78400	21952000	6.15E+09	1.72E+12
300	247	300	90000	27000000	8.1E+09	2.43E+12
320	376	320	102400	32768000	1.05E+10	3.36E+12
340	558	340	115600	39304000	1.34E+10	4.54E+12
360	806	360	129600	46656000	1.68E+10	6.05E+12

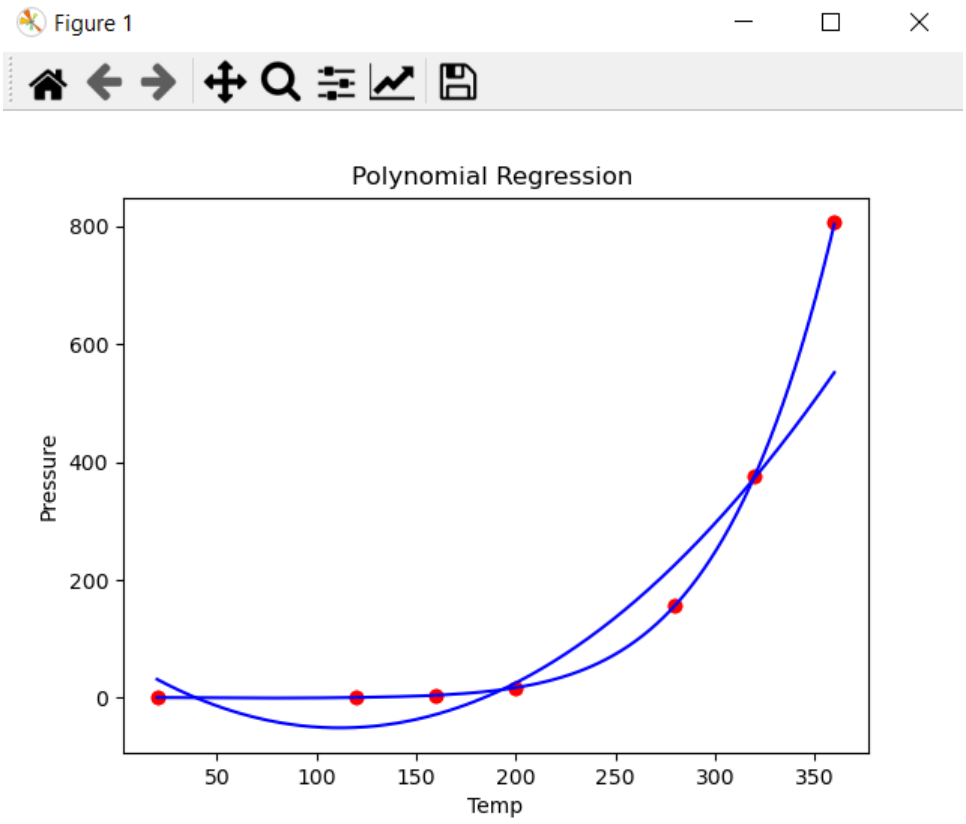
Degree 2 output:

Error is huge here.



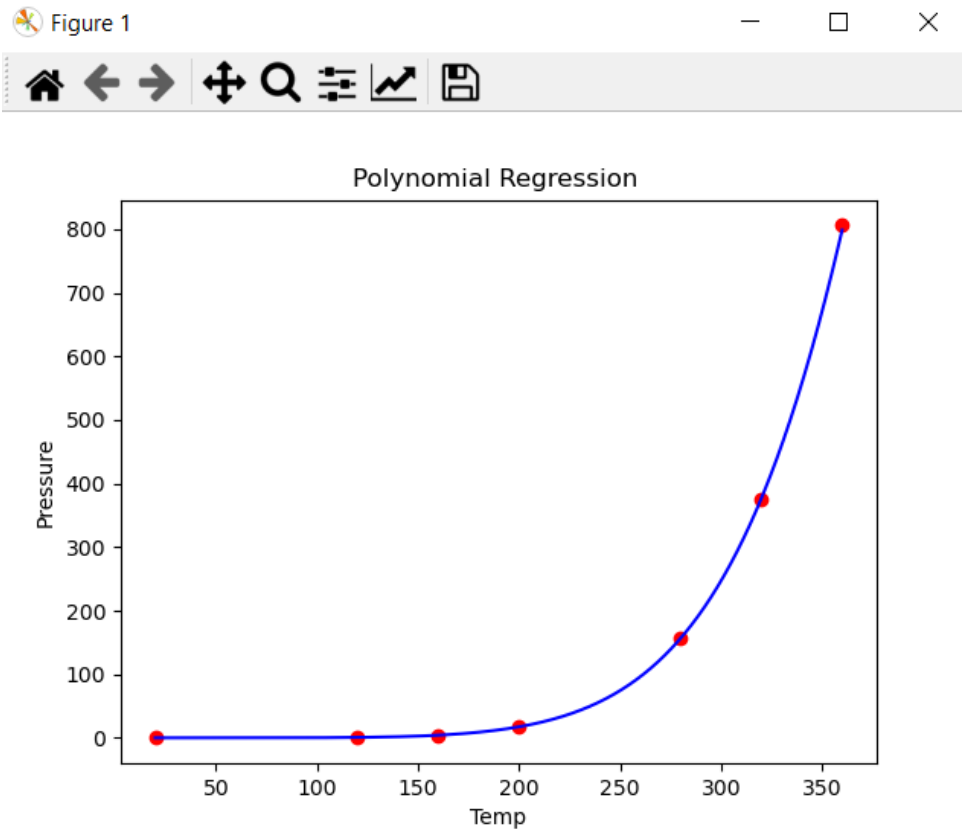
Degree 5:

Error is less. It’s fits 100%.



error_2	float64	1	10553.342229990534
error_5	float64	1	0.40559764248728847

Degree 10:



error_2	float64	1	10553.342229990534
error_5	float64	1	0.40559764248728847
error_10	float64	1	5.366908457395362

Degree 15:

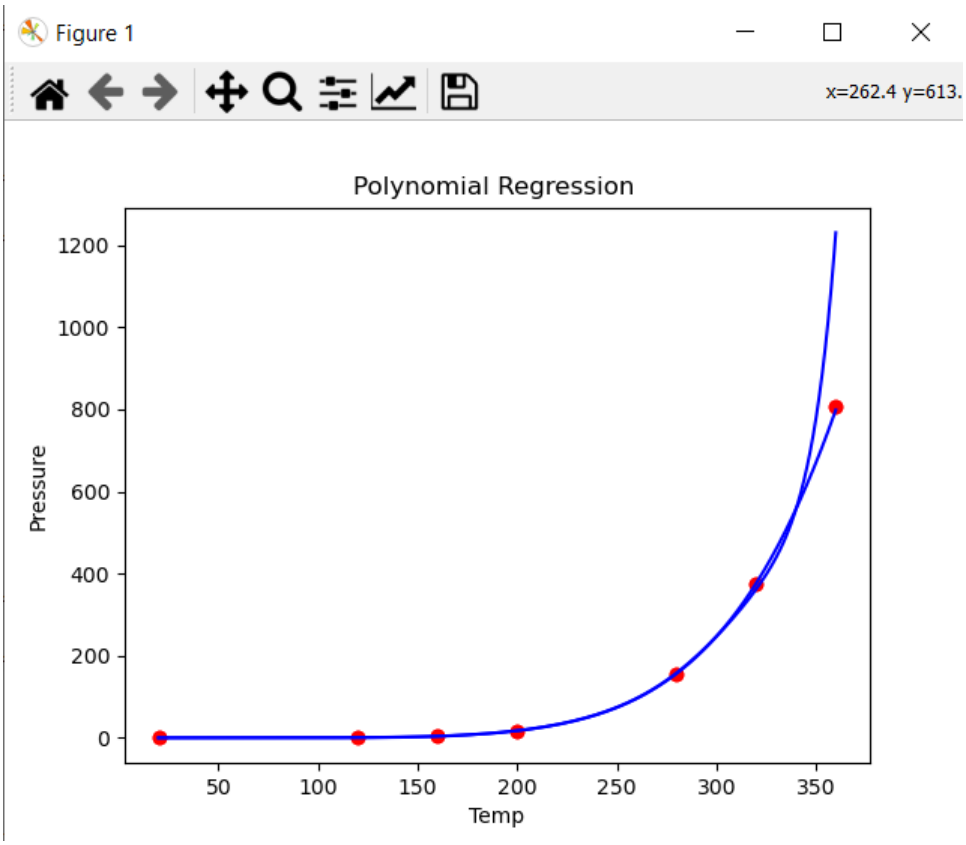
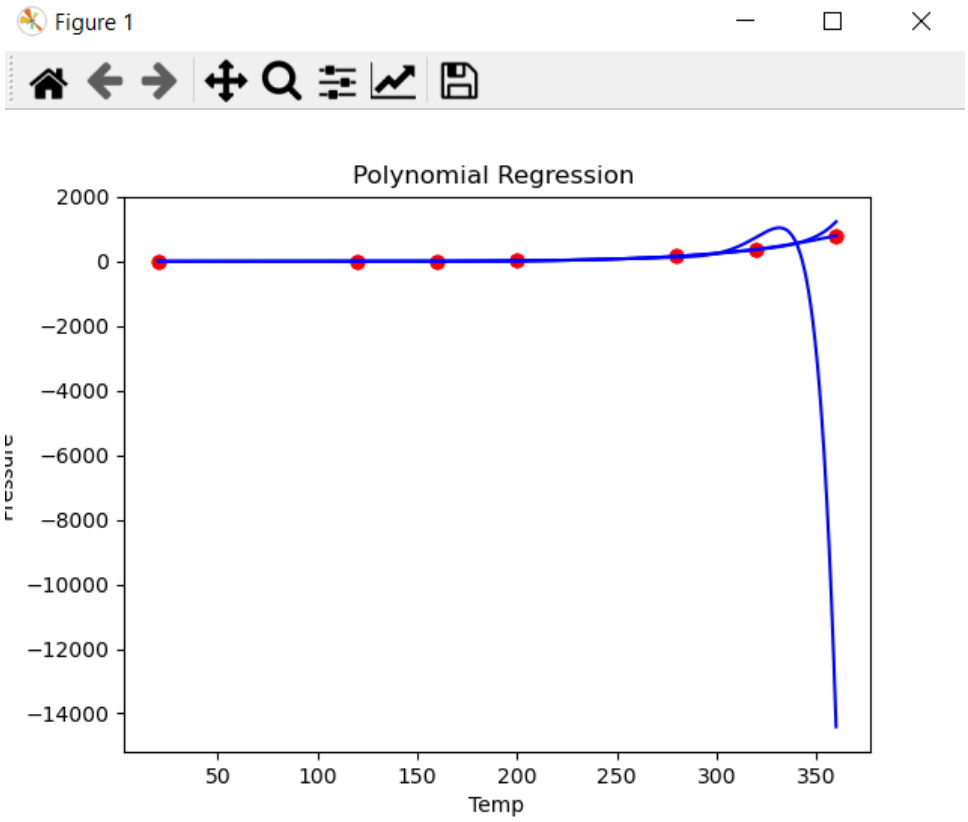


Figure 1
x=262.4 y=613.
Polynomial Regression
Pressure
Temp
0
200
400
600
800
1000
1200
50
100
150
200
250
300
350

Degree 20:



error_2	float64	1	10553.342229990534
error_5	float64	1	0.40559764248728847
error_10	float64	1	5.366908457395362
error_15	float64	1	26561.71204403531
error_20	float64	1	33961969.21293781

What is bias here?

```
# ------------------------------------------------------------------
from sklearn.preprocessing import PolynomialFeatures
poly_Obj = PolynomialFeatures(degree=20  ,  include_bias=False)
Xtrain_poly = poly_Obj.fit_transform(X_train)
Xtest_poly = poly_Obj.fit_transform(X_test)
```

We already discussed about the intercept. It helps to get the curve close to the spread of datapoints.

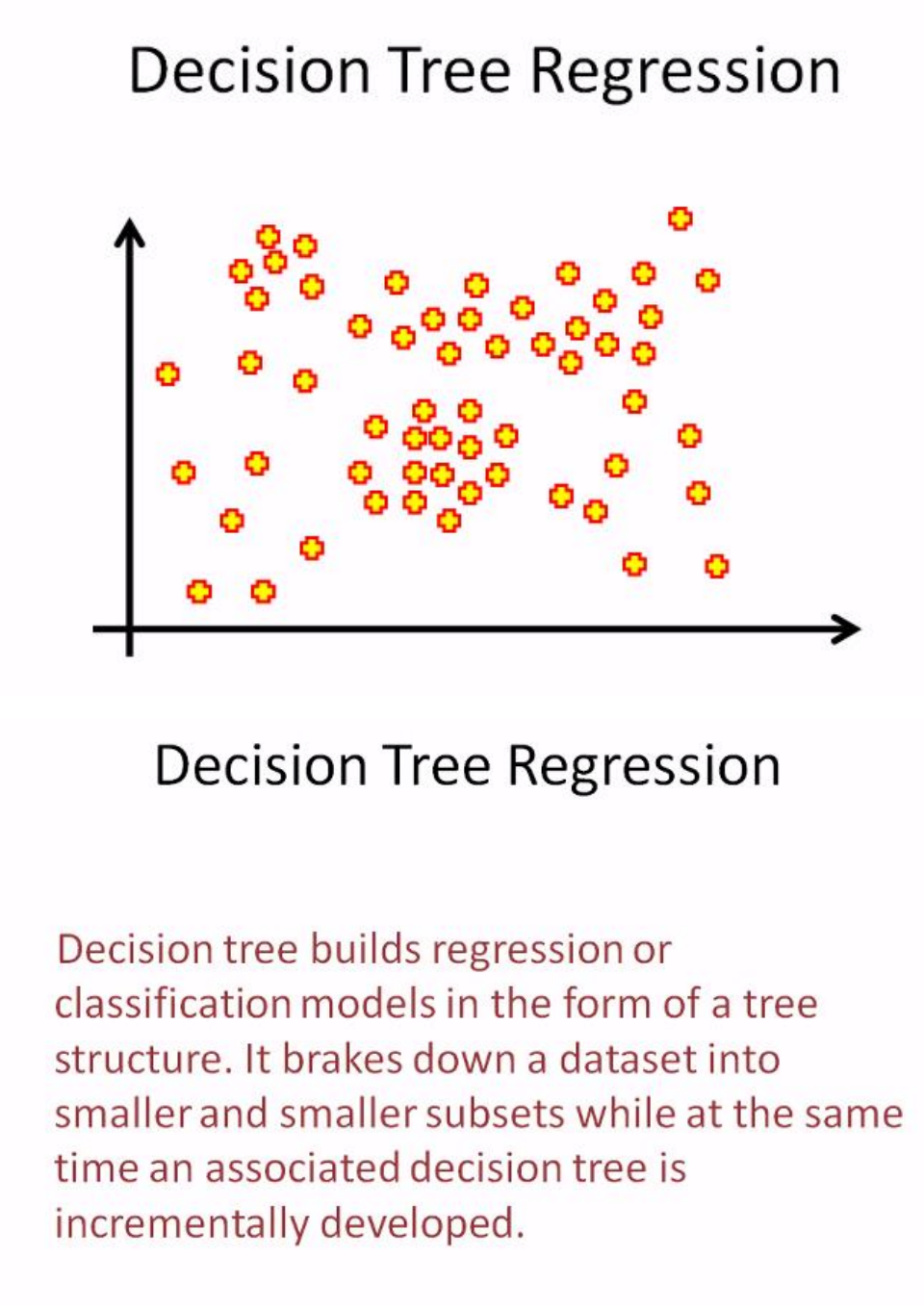
We can manually insert the intercept to keep it close to the spread of the information instead of machine calculating.

Feeding the intercept manually helps or not – range 0 to 1 will be good

Sometimes polynomial won't help. Go for Decision tree and Random Forest.

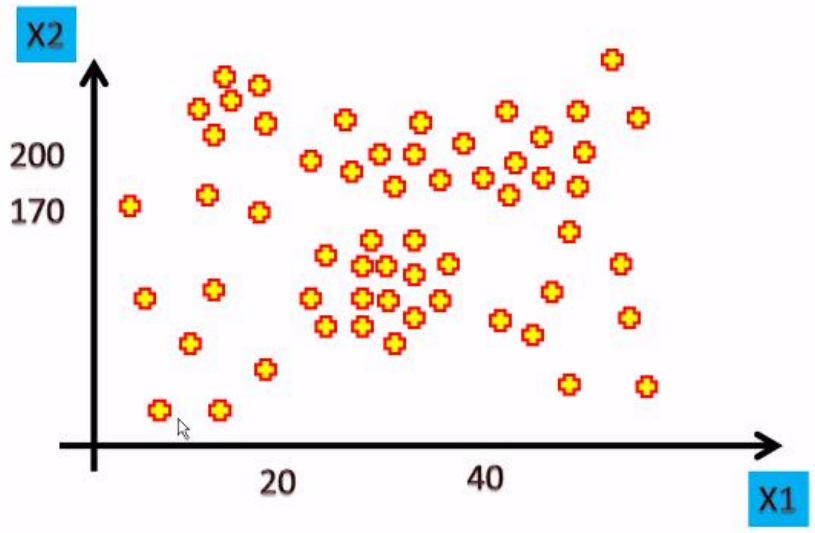
Decision tree regression:

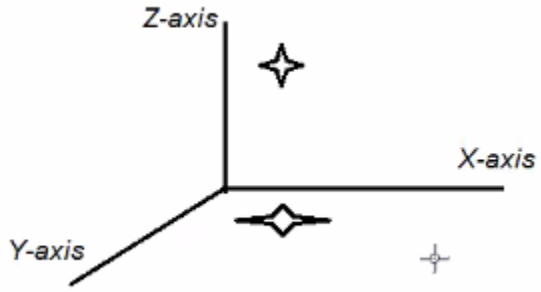
Forms in a tree structure. Breaks down into smaller and smaller subsets.



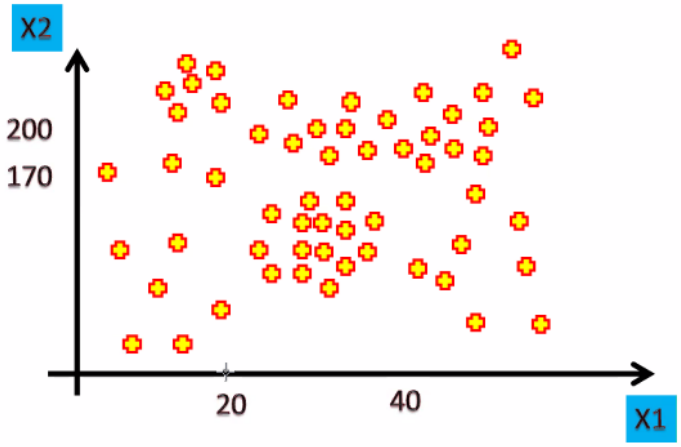
3 D Plotting looks like:

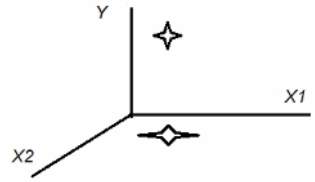
Decision Tree Regression



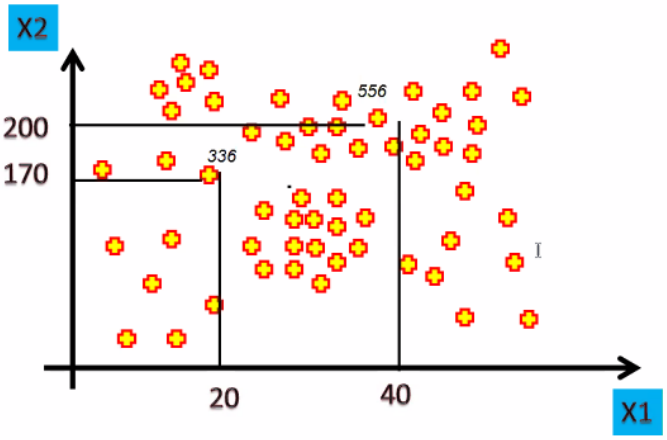


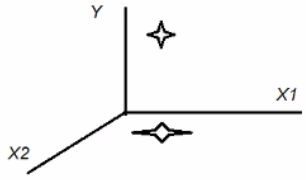
	input	*output*
X1	*X2*	*y*
20	*170*	*336*
40	*200*	*556*





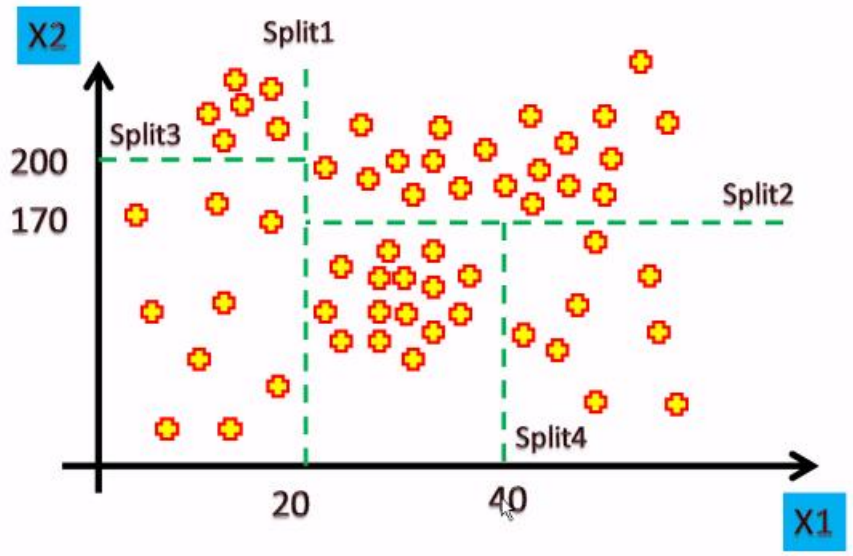
X1	X2	y
20	170	336
40	200	556



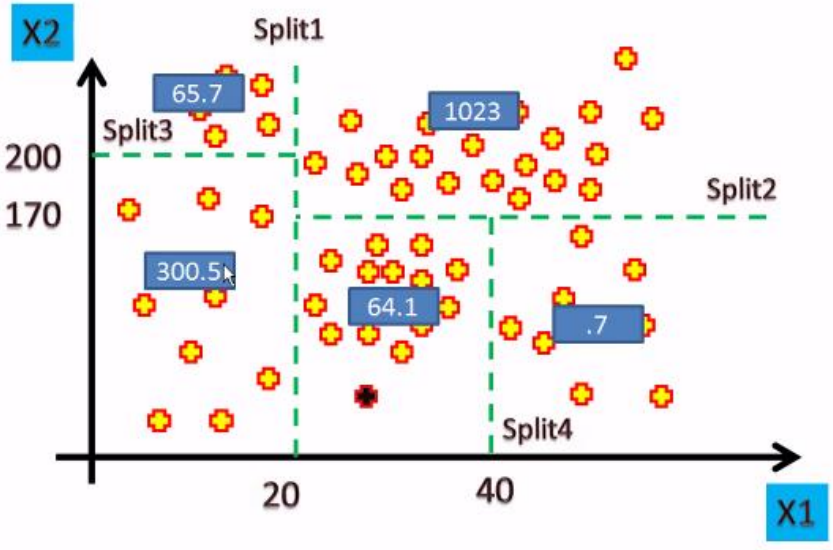


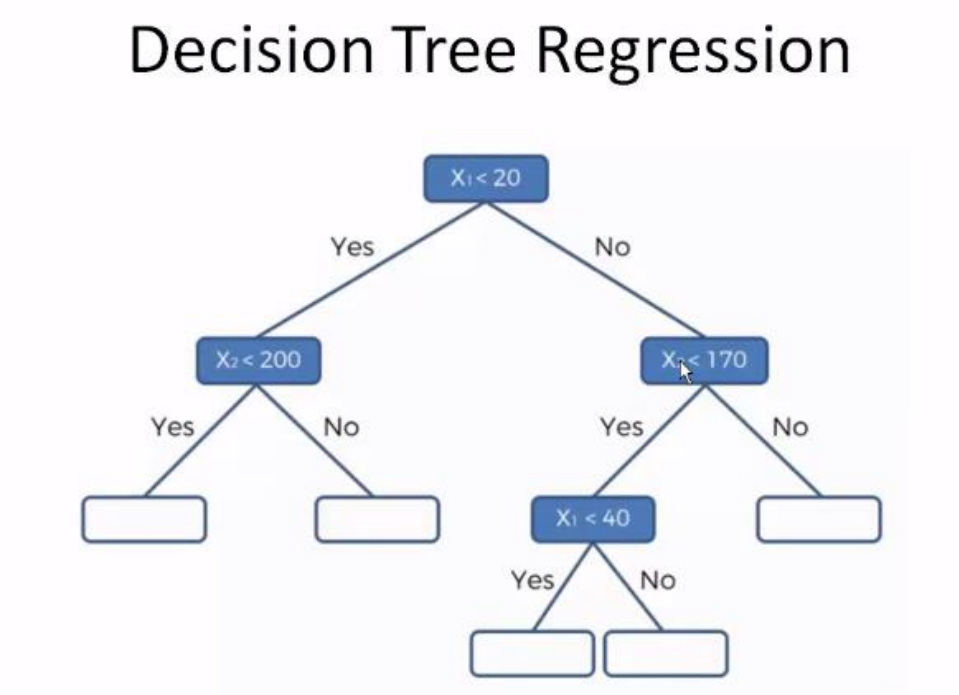
X1	X2	y
20	170	336
40	200	556

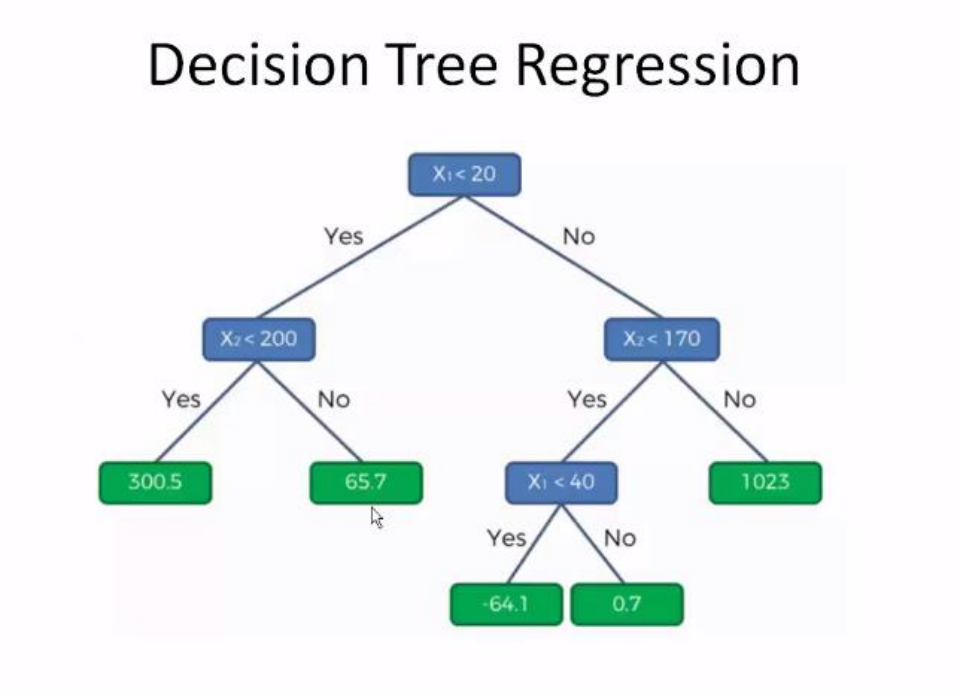
Decision Tree Regression



Decision Tree Regression





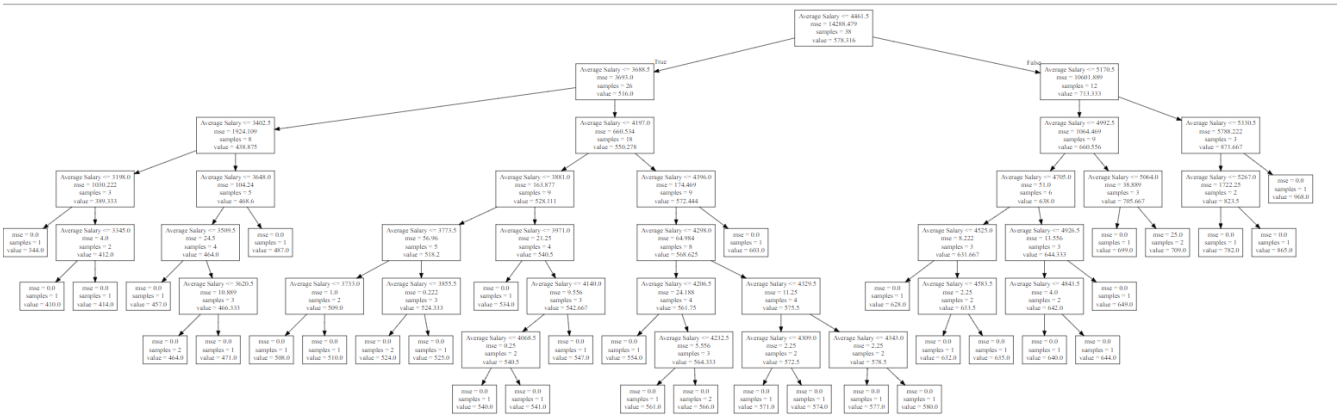


- Iterative in nature and time consuming
- Handle the complexity better than the polynomial regression.
- Good solution
- Performance – takes extra time.
- We can see the actual tree as well
- You will come here only when the polynomial regression is not good.

- Simple is better
- Prediction is just substituting values into the formula and getting it.
- Decision tree is good when we have good amount of data.
- Less data – not good to go with decision tree
- If the points falls under the split region, then the predicted value will be the average of that split region.

Web Graphviz: To visualize the tree structure

http://www.webgraphviz.com/



There will be overfitting problem! More splitting is done here.

Splitting has done in a granular level.

Overfitting:

Machine learns much on the training data and does only good only on training data.

It lacks generalization capabilities

Gives high error/ low accuracy on testing data

Multiple test sets tried against such model will lead to unstable error or accuracy and it won't be able to give appropriate results.

Can we control the number of splits? Whether to split or not depends on the value of N?

Value of N is considered for taking this decision.

Min_samples_split = n Deafult set to 2

If we have 2 or more data points in the split we divide it further.

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()

regressor.fit(X_train,y_train)

# ================================
# Model Testing
# ================================
y_pred = regressor.predict(X_test)

from sklearn.metrics import mean_s
error_DT = mean_squared_error(y_te


# ================================
# Tree plotting
# ================================
```

Arguments

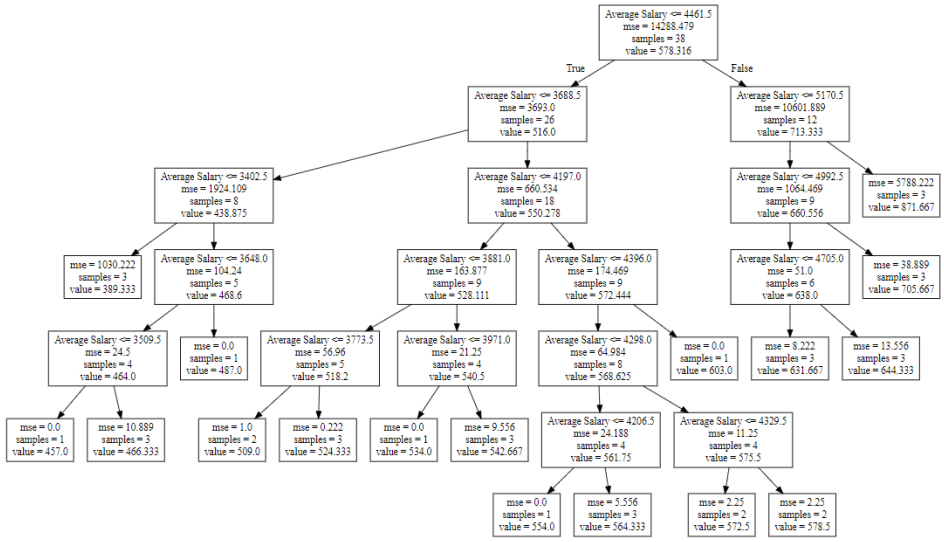
```
DecisionTreeRegressor(criterion="mse",
                      splitter="best",
                      max_depth=None,
                      min_samples_split=2,
                      min_samples_leaf=1,
                      min_weight_fraction_leaf=0.,
                      max_features=None,
                      random_state=None,
                      max_leaf_nodes=None,
                      min_impurity_decrease=0.,
                      min_impurity_split=None,
                      presort='deprecated',
                      ccp_alpha=0.0)
```

Should we split or confirm it as leaf node?

In the above case, it has spitted more. Results in overfitting!

Now, provide the min sample split as 4

error_DT	float64	1	39.9
error_DT4	float64	1	46.1722222222222



If the split value is more, then it results in **underfitting! (Too crisp and too much generalization)**

Step 1: Train model with min sample slit = n

Step 2: Test sample	Error
Sample 1	39.0
Sample 2	45.0
Sample 3	47
Sample 4	46.0

underfitting	Not learning Enough Too much of generalization the model is learning to do testing error will be very high / acc will be very low
Overfitting	Machine learns too much on training data Does only good on training data it lacks generalization capabilities gives high error / low accuracy on testing data multiple test sets tried against such model will lead to unstable error or accuracy

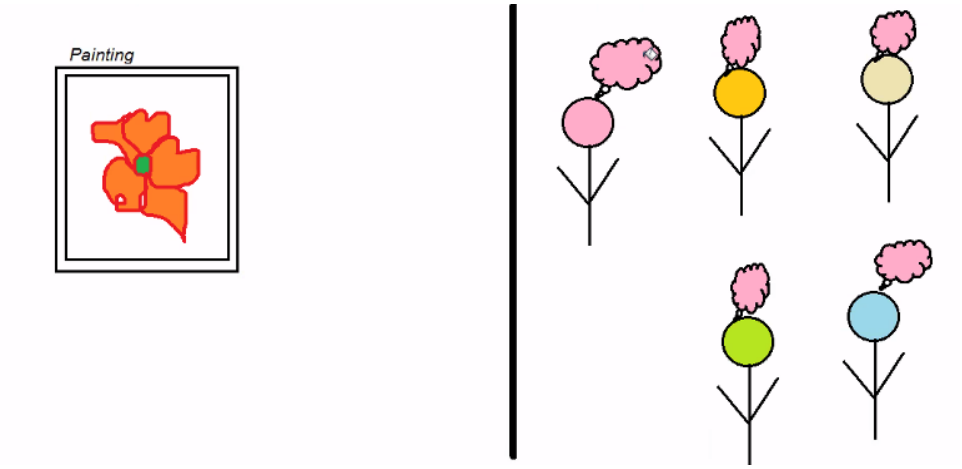
Good fit

- Enough learning
- Has good generalization capability
- Testing error low / accuracy high, and stable

Note: The error should be stable. You should be able to reproduce the same output. It should do good in all machines. So, stability check is a must!

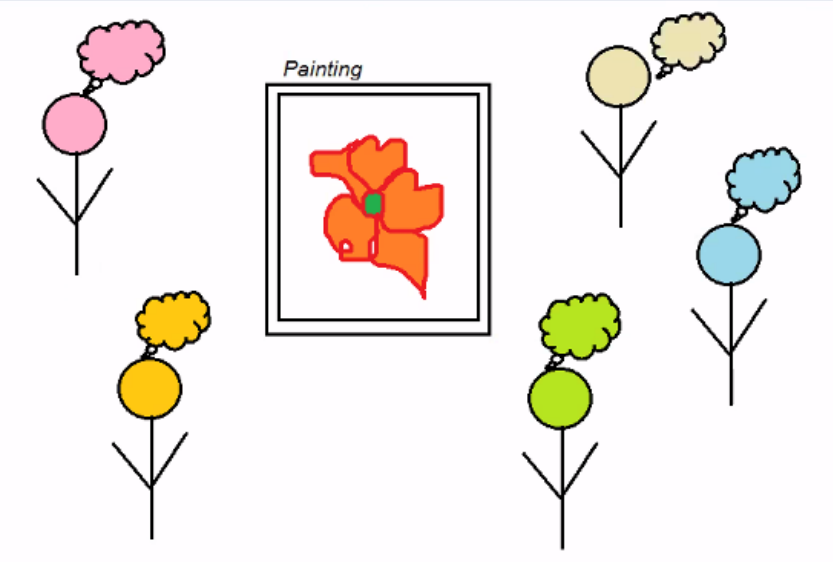
Random Forest:

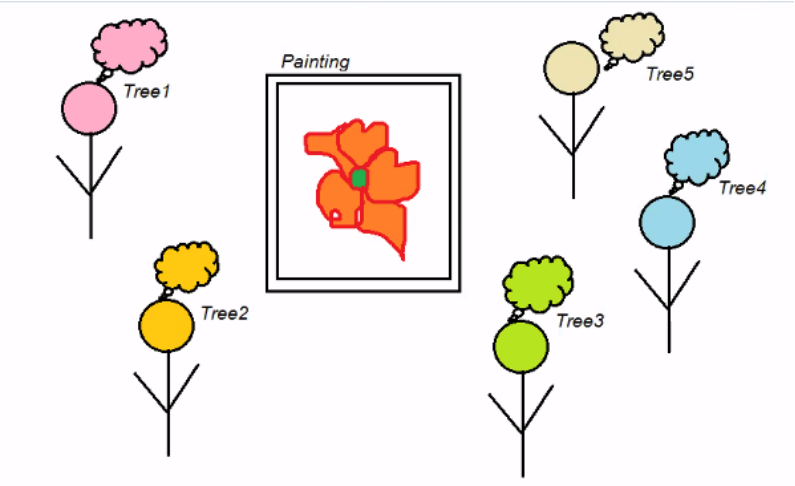
One tree is one perspective of looking at the data. That is the problem -> Inappropriate results



Now everyone is going inside the art gallery and having their own perspective. Better idea about the painting. Multiple perspective out of same painting -> Random Forest.

It decides from multiple tress.

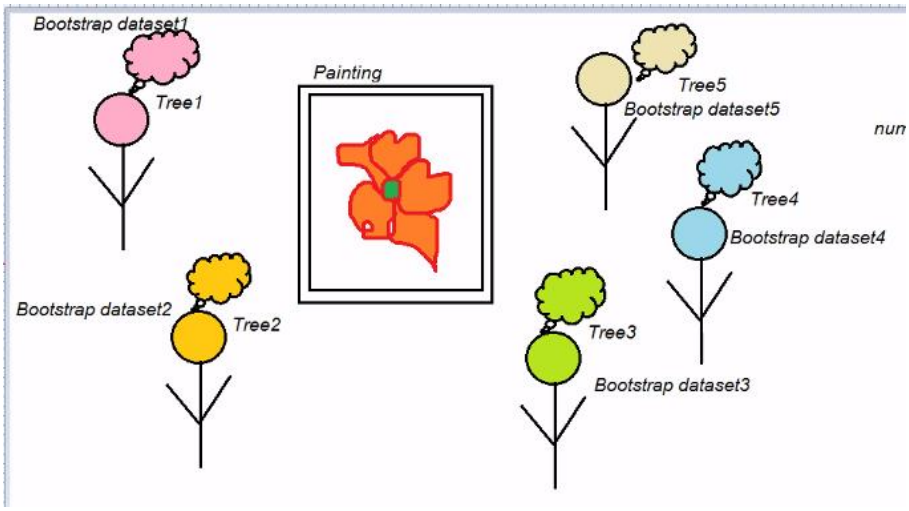




Overall Dataset will be same

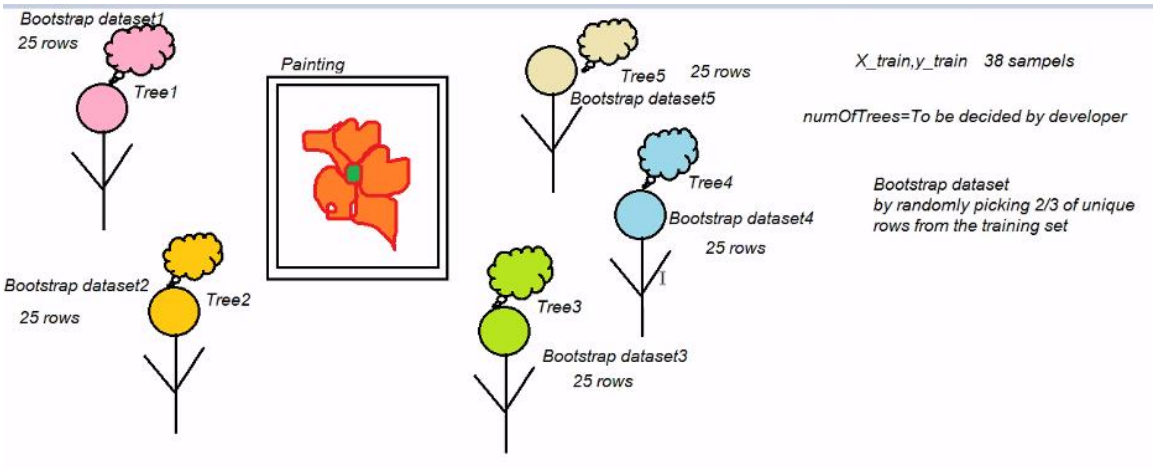
Number of trees = to be decided by developer

Each tree will have its own bootstrap dataset



How is the bootstrap dataset is constructed?

- Method to split the dataset: Bootstrap.
- Without replacement and its pure random
- By randomly picking 2/3 of unique rows from the training dataset.
- Each tree is different as it picks up different bootstrap dataset.



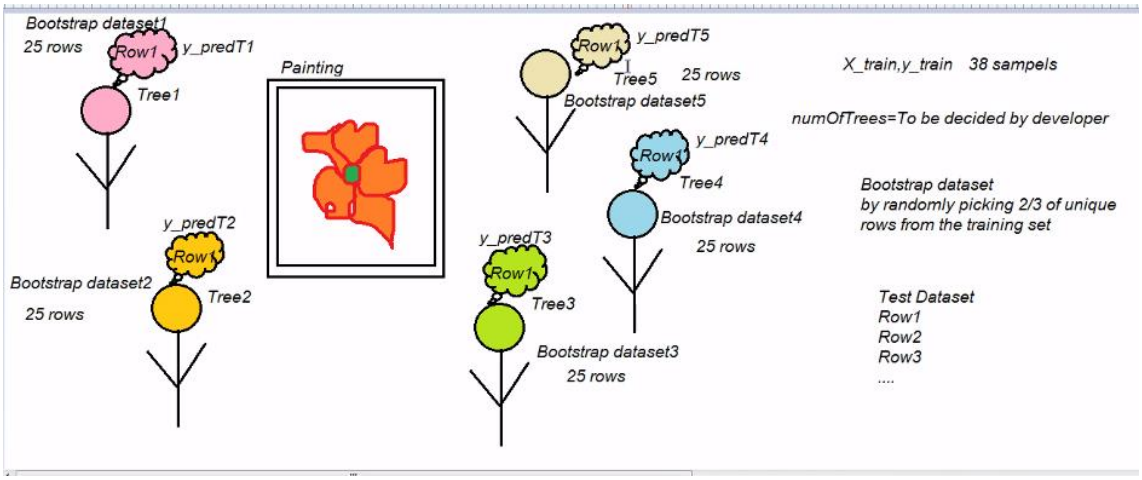
Test Dataset:

Row 1

Row 2

Row 3

…….



What am I going to do with all these predictions?

Test Dataset

Row1 = y_pred = avg(y_predT1,y_predT2,y_predT3,y_predT4,y_predT5)

Row2 = y_pred = avg(y_predT1,y_predT2,y_predT3,y_predT4,y_predT5)

Row3 = y_pred = avg(y_predT1,y_predT2,y_predT3,y_predT4,y_predT5)

....

```
NumOfTrees = To be decided by developer

RandomForest Tree Creation          (X_train, y_train)          38 samples

Step1:    => BootStrap Dataset Creation i.e. Random Row selection from the given training set
          => BootStrap dataset will be SAME SIZE as Training Dataset
                              e.g. X_train,y_train          38 samples
                                BootStrap Dataset           38 Samples

          => But Bootstrap Dataset      doesnt use all 38 values in the training set to create the BootStrap dataset
                    it just uses 2/3 of the total values in the training set to create the BootStrap dataset

                    2/3(training set)    ----------> 66% rows,  Bagging, remainder is called out of Bag Records
                    2/3(38 rows)         ----------> 25 rows random pick values from training set (66%)

          => BootStrap dataset after taking 2/3 unique rows the remainder rows are filledup to match the training set size by
                    duplicating rows from 2/3 volume

RandomForest (10 trees)

Tree1     BootStrap Dataset1
Tree2     BootStrap Dataset2
Tree3     BootStrap Dataset3
Tree4     BootStrap Dataset4
.............
Tree10    BootStrap Dataset10
```

Training Sample	
Row1	pattern1
Row2	pattern2
Row3	pattern1
Row4	pattern1
Row5	pattern2
Row6	pattern1
Row7	pattern3
Row8	pattern2
Row9	pattern2
Row10	pattern1
Row11	pattern2
BootStrap Dataset1	
Row1	pattern1
Row2	pattern2
Row3	pattern1
Row4	pattern1
Row5	pattern2
Row6	pattern1
Row7	pattern3
Row7	pattern3
Row7	pattern3
Row6	pattern1
Row6	pattern1

It’s a random pick. For understanding, done as systematic pick.

2/3 volume is the pickup volume.

Training Sample

Row1	pattern1
Row2	pattern2
Row3	pattern1
Row4	pattern1
Row5	pattern2
Row6	pattern1
Row7	pattern3
Row8	pattern2
Row9	pattern2
Row10	pattern1
Row11	pattern2

BootStrap Dataset1

Row1	pattern1
Row2	pattern2
Row3	pattern1
Row4	pattern1
Row5	pattern2
Row6	pattern1
Row7	pattern3
Row7	pattern3
Row7	pattern3
Row6	pattern1
Row6	pattern1

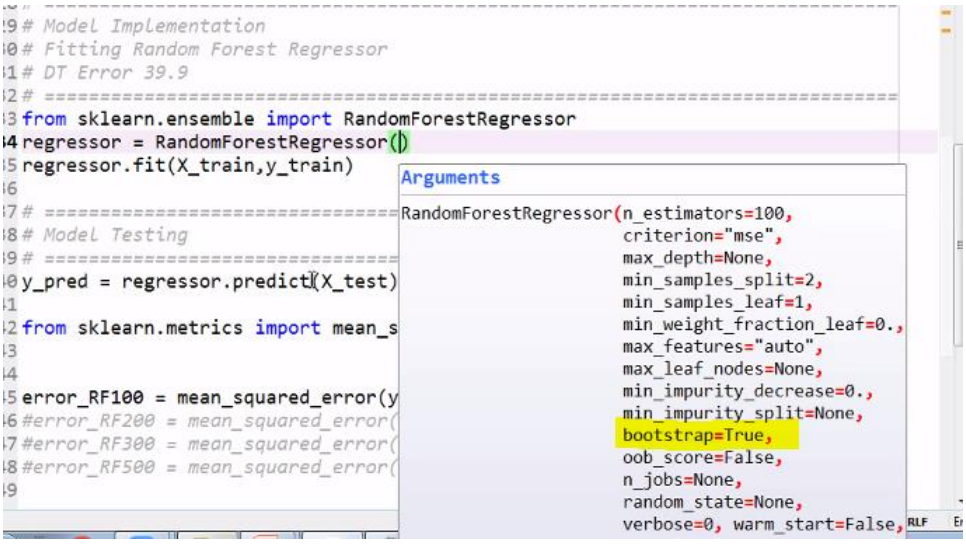
Randomly pick up from bootstrap dataset to fill the gap.

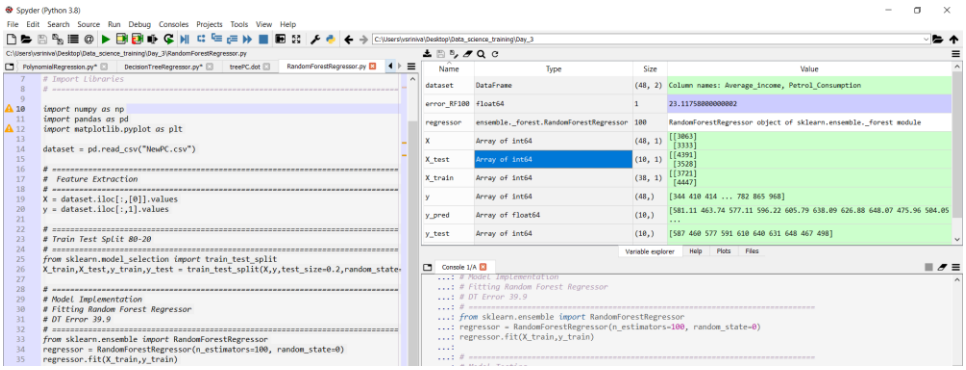
Bootstrap and training dataset size should be same.

BootStrap Dataset1

Row1	pattern1
Row2	pattern2
Row3	pattern1
Row4	pattern1
Row5	pattern2
Row6	pattern1
Row7	pattern3
Row7	pattern3
Row7	pattern3
Row6	pattern1
Row6	pattern1

Randomness won’t be effective after certain number of trees.





error_RF100	float64	1	23.11758000000002
error_RF200	float64	1	22.971057500000065
error_RF300	float64	1	23.582036666666742
error_RF500	float64	1	23.001594500000017

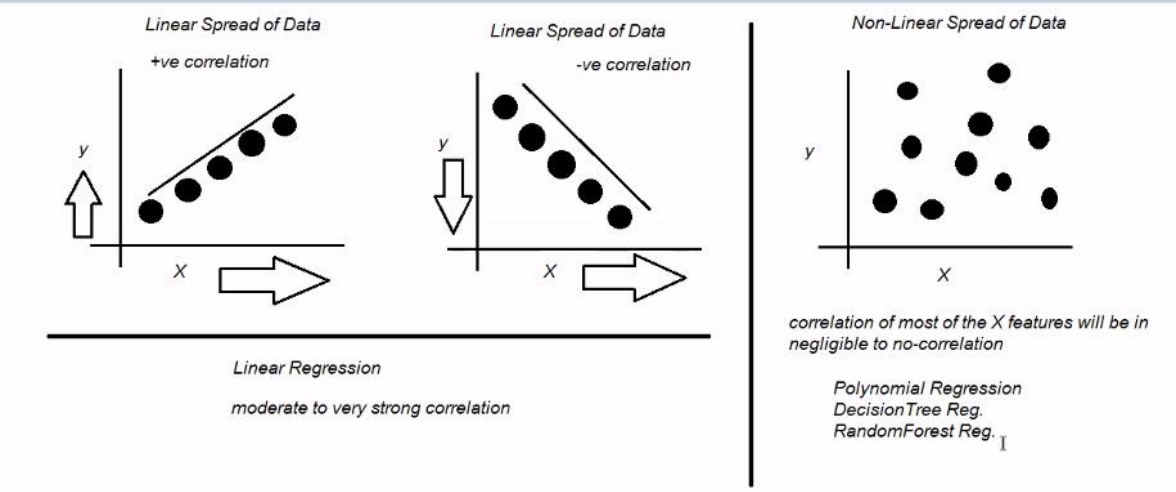
Observation:

Exhausted the error value after 100. You can see the same similarity.

Try it between 120 to 150:

error_RF100	float64	1	23.11758000000002
error_RF120	float64	1	21.793486111111214
error_RF150	float64	1	21.8920755555556
error_RF200	float64	1	22.971057500000065
error_RF300	float64	1	23.582036666666742
error_RF500	float64	1	23.001594500000017

Regression overview:



Which regression supports what kind of spread of data?

SUPERVISED LEARNING	Input Data & Output Data	Remember & Generalize
Regression	Value to Predict is Continuous	
Linear Regression		Linear spread of data
Polynomial Regression		non-Linear spread of data
DecisionTree Regressor		non-Linear spread of data
RandomForest Regressor		non-Linear spread of data
Classification	Value to Predict is Discrete	
Logistic Regression		
Support Vector Machine		
DecisionTree Classifier		
RandomForest Classifier		
Naive Bayes		

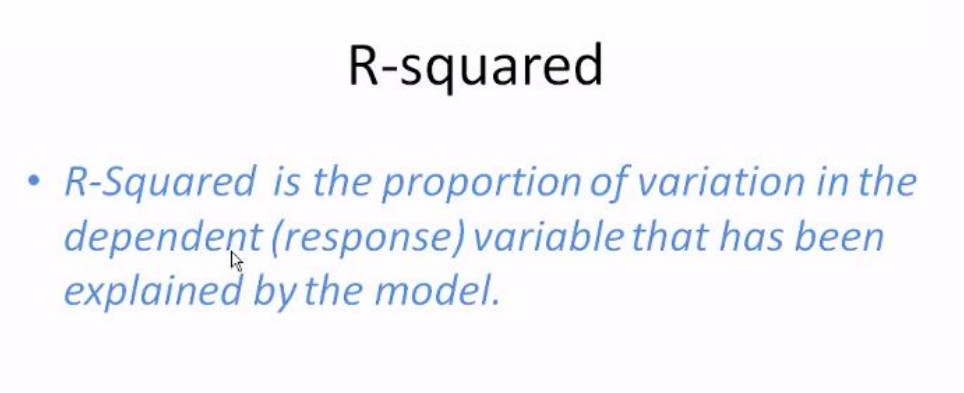
How to decide on which regression to pick up?

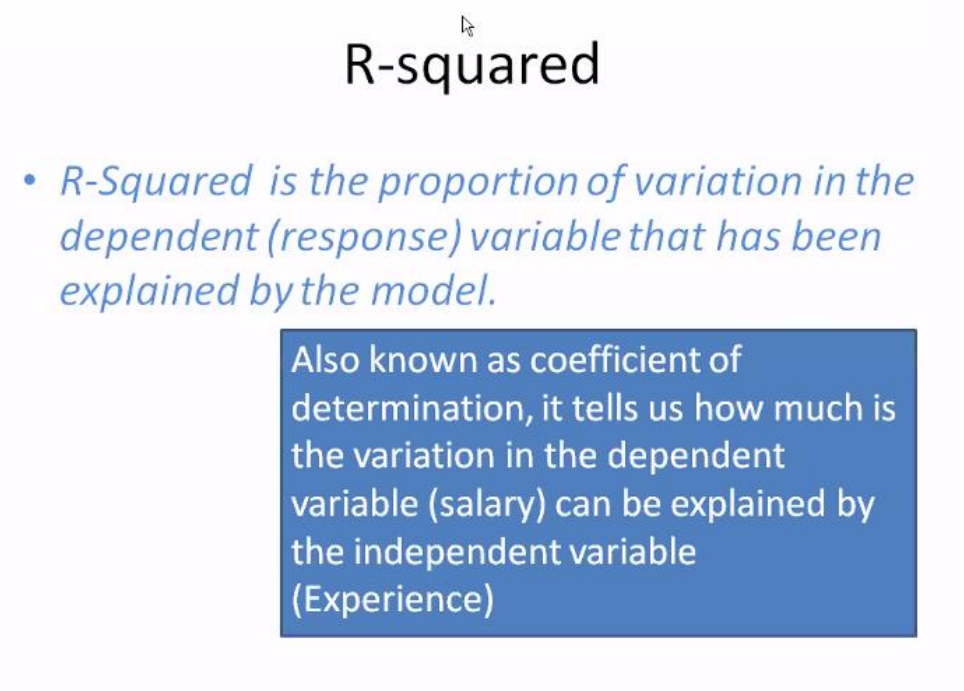
If most of the X features (80%) are moderate to very strongly correlated to Y feature
Then Choose LINEAR REGRESSION

If most of the X features (80%) are no correlation to negligible correlation to Y feature
Then Choose Poly Reg. , DT , RF

R2 score:

Tell us how much variation in y that can be explained by all x features.

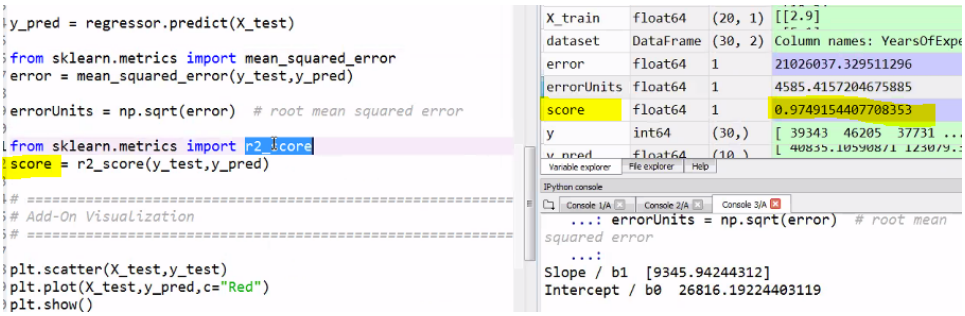




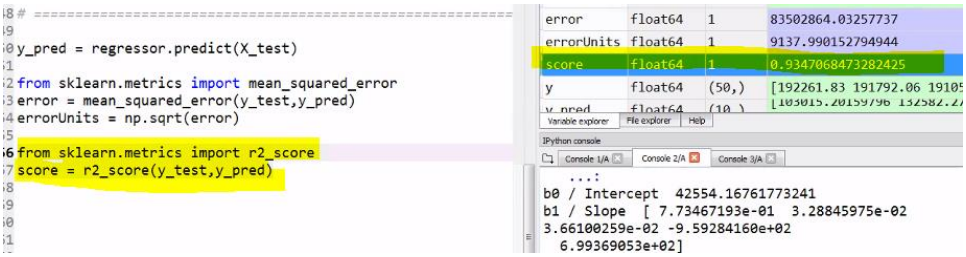
R2 score of 80 and above is healthly.

```
51 from sklearn.metrics import r2_score
52 score = r2_score(y_test,y_pred)
53
```

Simple regression:



Multiple regression:



```
y_pred = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
error = mean_squared_error(y_test,y_pred)
errorUnits = np.sqrt(error)

from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred)
```

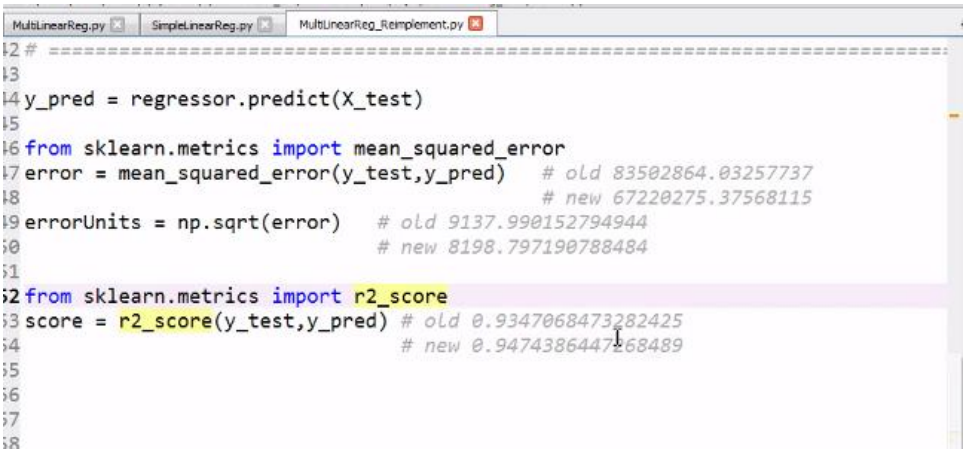
```
error       float64   1   83502864.03257737
errorUnits  float64   1   9137.990152794944
score       float64   1   0.9347068473282425
y           float64   (50,)   [192261.83 191792.06 191
y_pred      float64   (10,)   [183015.20159796 132582.27
```

```
...:
b0 / Intercept  42554.16761773241
b1 / Slope  [ 7.73467193e-01  3.28845975e-02
3.66100259e-02 -9.59284160e+02
  6.99369053e+02]
```

Pearsonr score:

```
datasetEval = pd.read_csv("02Companies.csv")

datasetEval = pd.get_dummies(data=datasetEval, columns=["State"], drop_first=True)

from scipy.stats import pearsonr

pearsonr(datasetEval['RNDSpend'], datasetEval['Profit'])
pearsonr(datasetEval['Administration'], datasetEval['Profit'])
pearsonr(datasetEval['MarketingSpend'], datasetEval['Profit'])
pearsonr(datasetEval['State_Florida'], datasetEval['Profit'])
pearsonr(datasetEval['State_New_York'], datasetEval['Profit'])
```

Taking all features vs taking specific features:



MultiLinearReg.py | SimpleLinearReg.py | MultiLinearReg_Reimplement.py

```
y_pred = regressor.predict(X_test)

from sklearn.metrics import mean_squared_error
error = mean_squared_error(y_test,y_pred)   # old 83502864.03257737
                                            # new 67220275.37568115
errorUnits = np.sqrt(error)   # old 9137.990152794944
                              # new 8198.797190788484

from sklearn.metrics import r2_score
score = r2_score(y_test,y_pred) # old 0.9347068473282425
                                # new 0.9474386447268489
```

Comparison of Linear Regression and Multiple regression:

error_LR	float64	1	556.0375904395161
error_RF	float64	1	21.8920755555556

Classification:

Use cases:

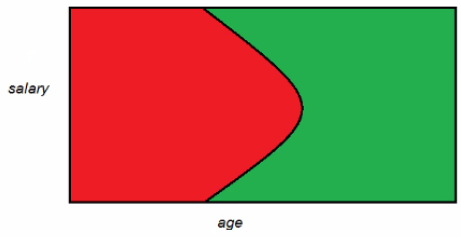
User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000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

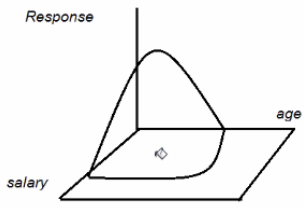
Discrete Value Prediction, Measure of model performance is Accuracy percentage

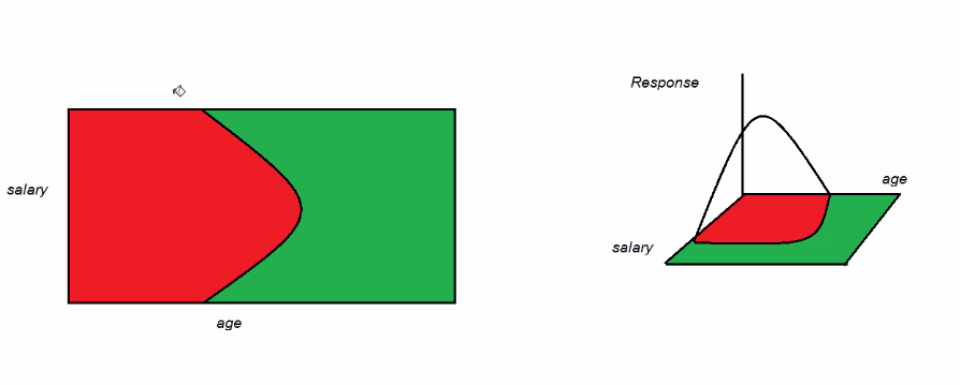
Age	ActualResponse	PredictedResponse
20	0	0
30	1	1
21	0	0
35	1	0
40	1	1
45	1	1
50	0	1
18	0	0
19	1	0

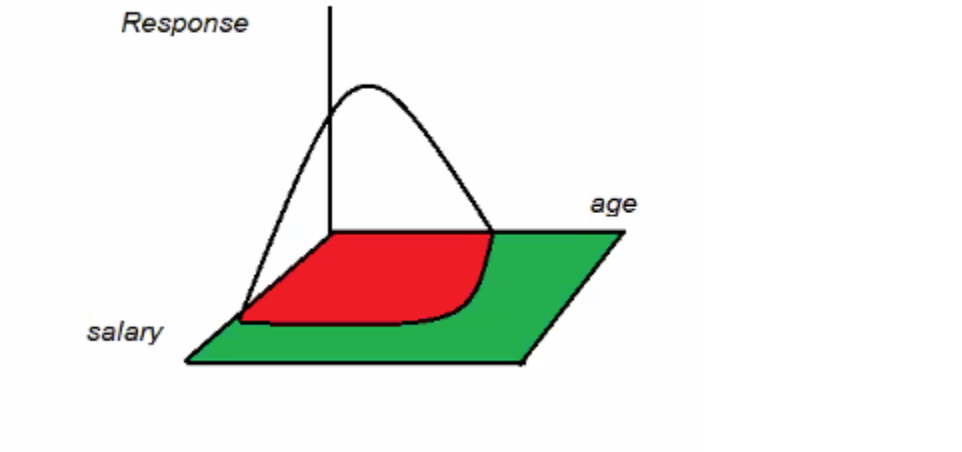
6/9*100=66% accuracy score

How am I going to generalize here?









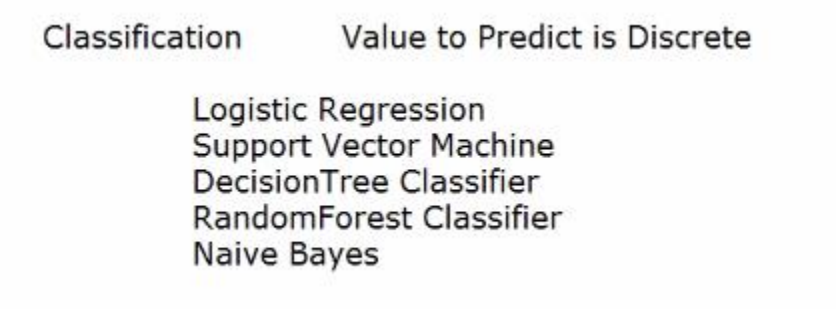
For Classification, accuracy is in the range 0 to 1. We can't take MSE here.

Consider multiple test sets and get accuracy.

Do we have a good training set?

K-fold validation – It is used for Evaluation.

Statistical calculation.



SVM: Support Vector Machine

Dataset:

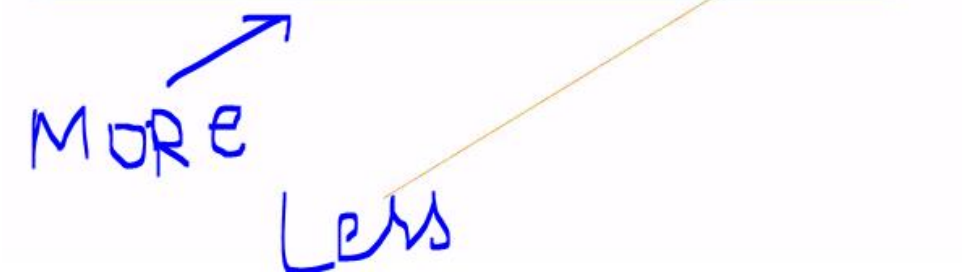
dataset - DataFrame

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0

Scaling:

Age	Salary	BankBalance	CreditScore	OwnHouse
30	24000	50000	650	1

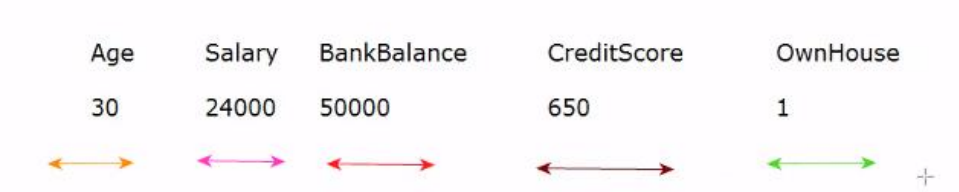
Age	Salary	BankBalance	CreditScore	OwnHouse
30	24000	50000	650	1



Bank balance: Contributes more in calculation

Own House: Contributes less in calculation

Each has its own scale



Machine doesn't understand its scale.

So, get everything under one scale

Take example: Common scale – range –2 to 2

To take balanced data

To avoid the imbalance

There are scalar available to perform this operations.

Standard scaler – good if your dataset doesn't have many outlier

Min and max scalar – good to use when your dataset has more outlier

Max absolute scalar

Robust scalar

Most popular standard, and min and max scalar

Standardization StandardScaler *(if the dataset doesnot have too many outliers)*

$$X_{stand} = \frac{X - mean(X)}{standard\ deviation\ (x)}$$

MinMaxScaler *(if the dataset has too many outliers)*

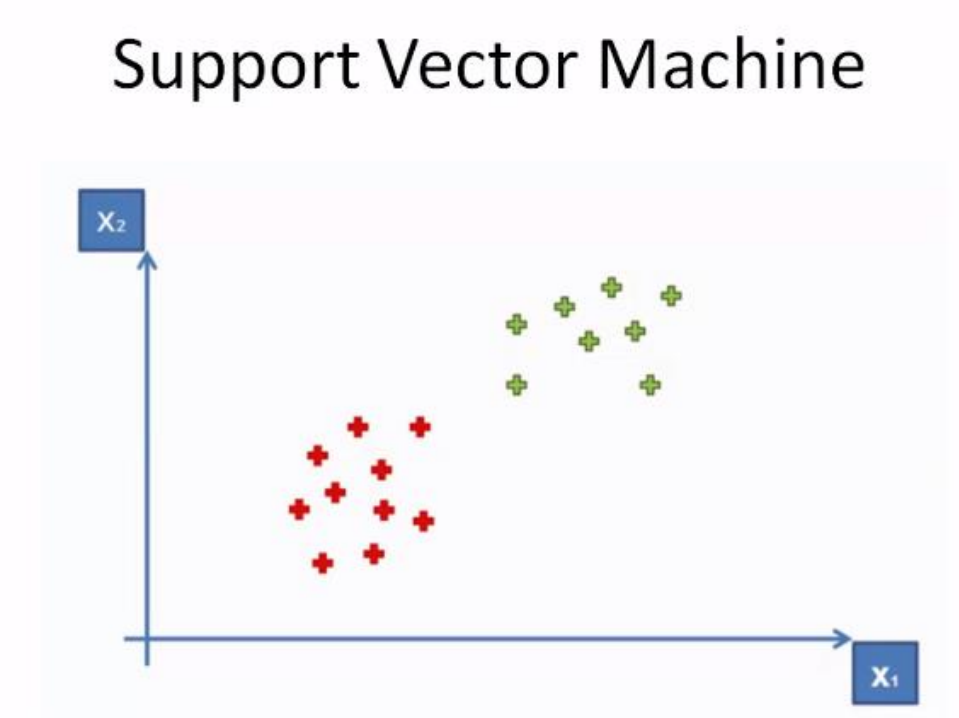
$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

We won't having scaling in regression as we didn't face imbalance situation there.

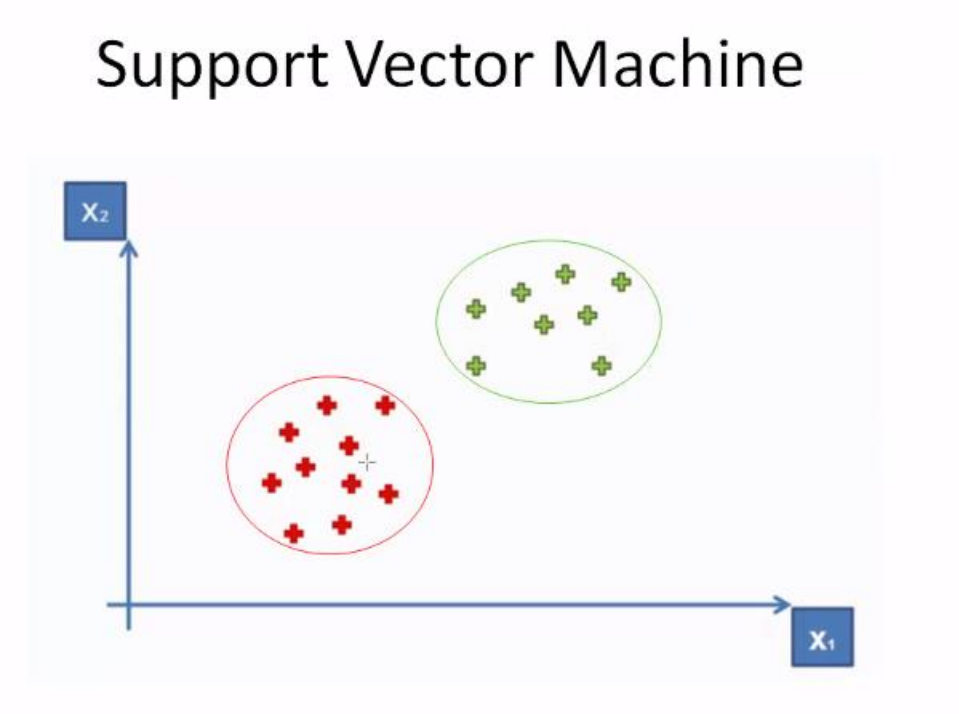
```
In [6]: np.min(X_train)
Out[6]: -1.9931891594584856

In [7]: np.max(X_train)
Out[7]: 2.3315320031817324

In [8]:
```

Pretty balanced! Not too low or high.

SVM:

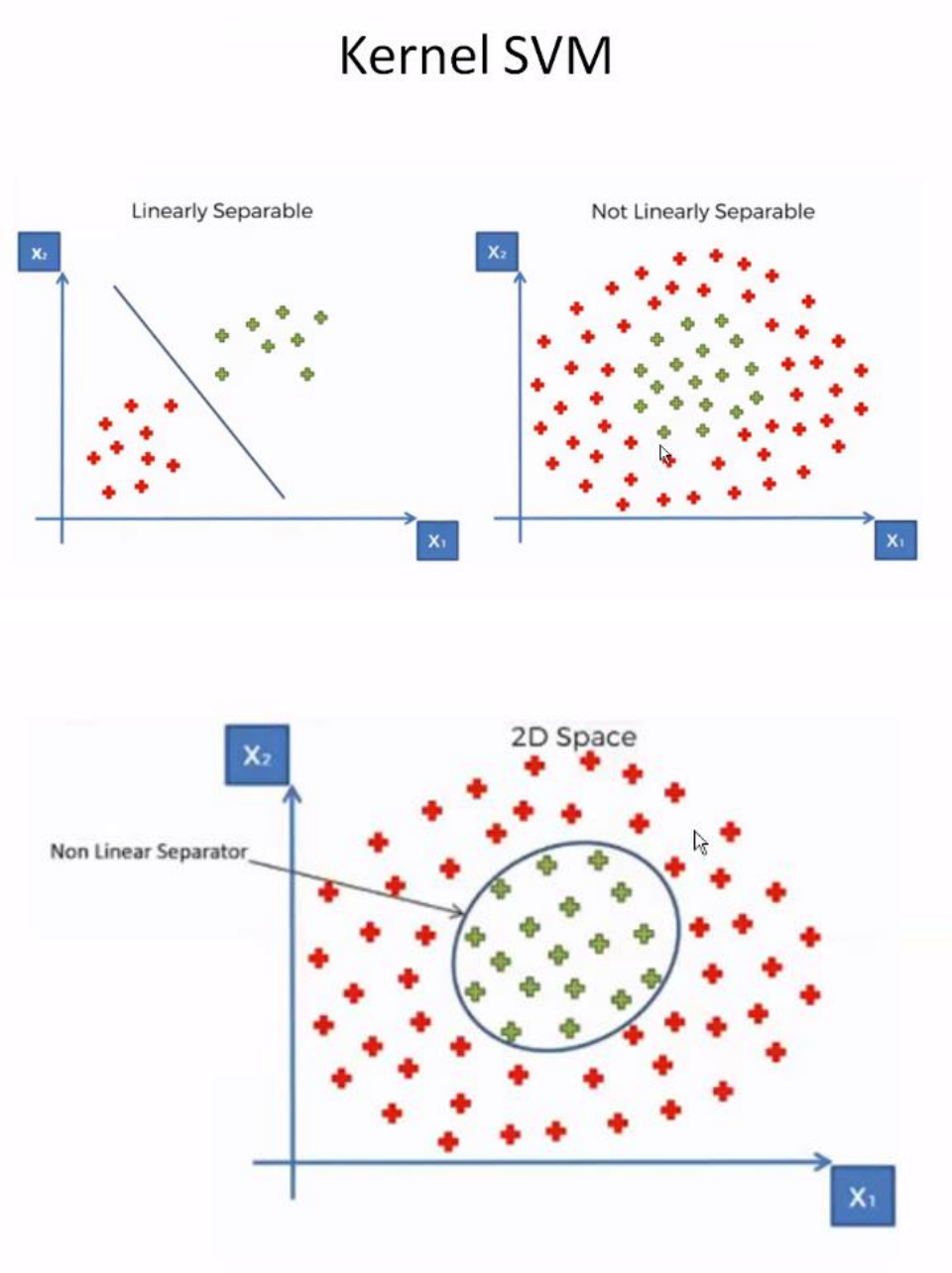


Learn by looking into the similarity and dissimilarity of the data.

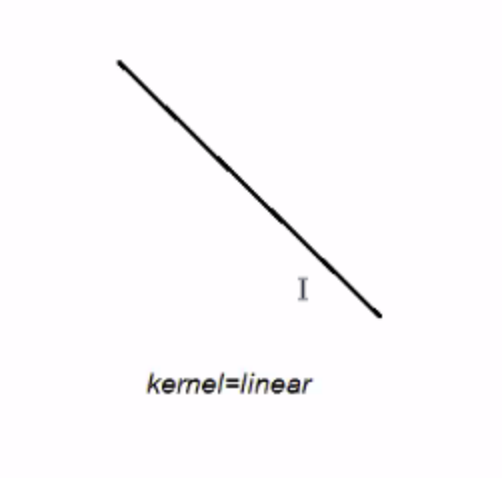


At boundaries, you can see data will not be similar.

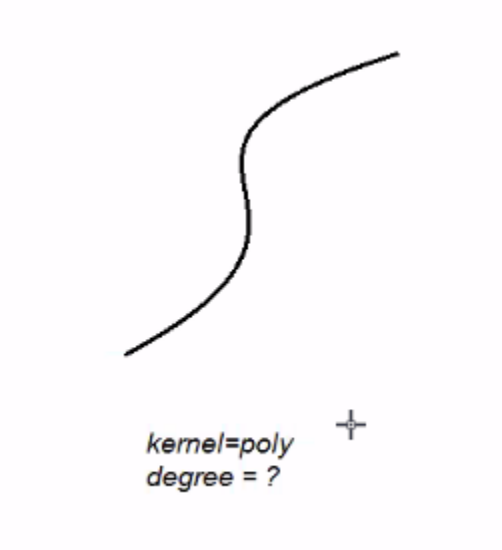
We will draw the hyperplane



Simple problem:

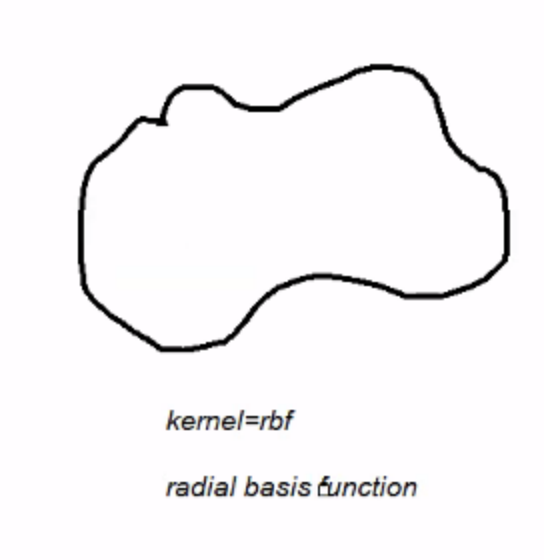


For complex problem: poly kernel



Didn’t serve long as data is more complex in real world

Radial basis function



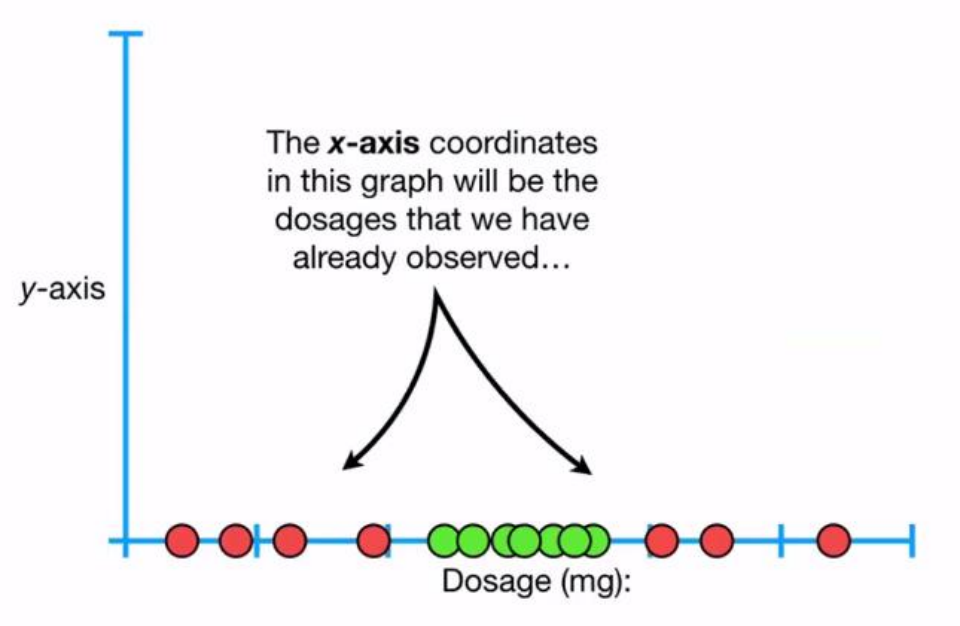
What is the math behind them?

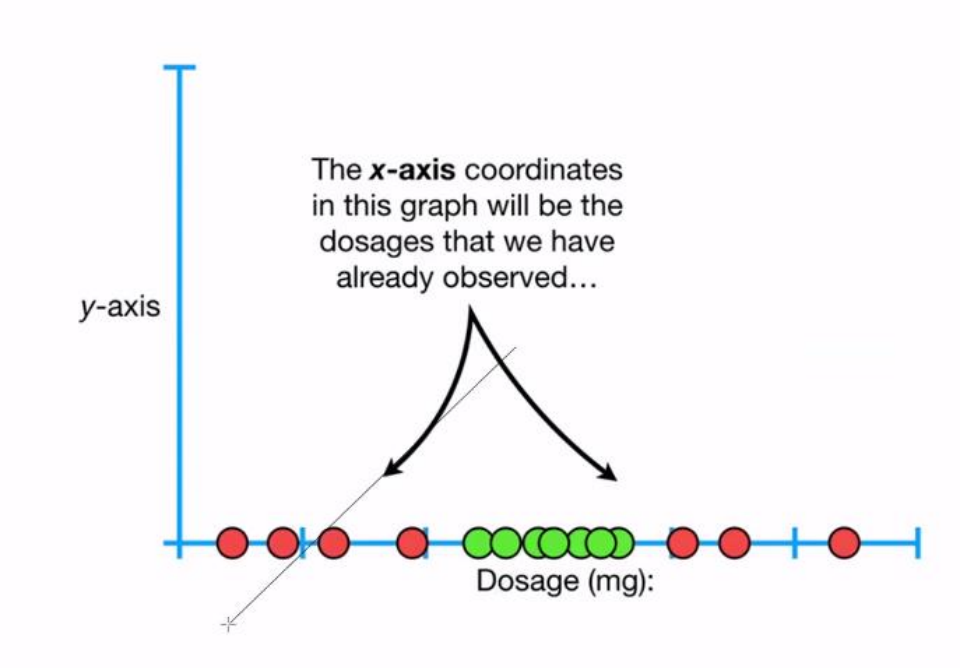
Example: Doctor prescribed the medicines with different dosages

Green – cured

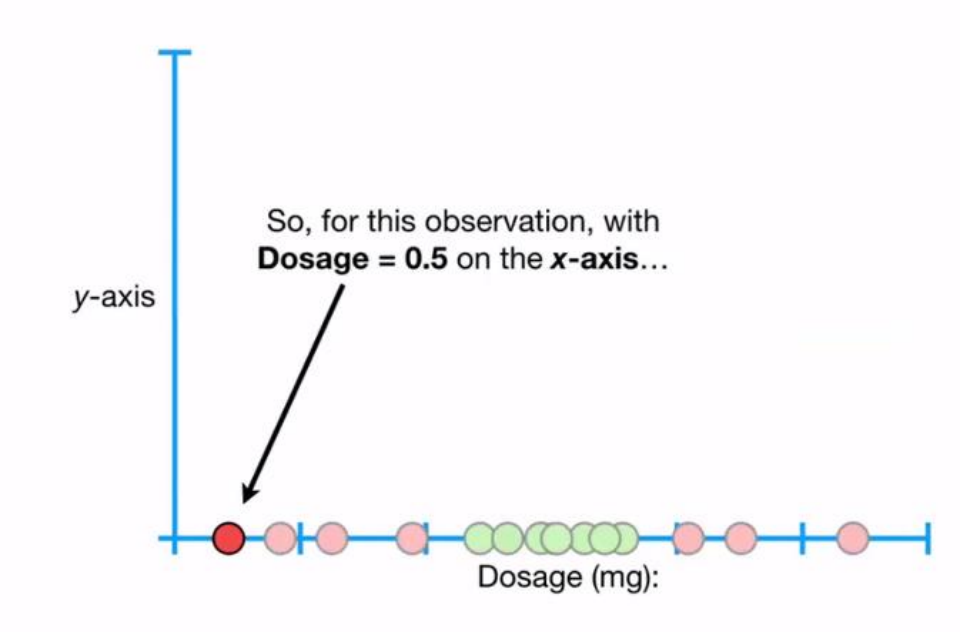
Red – not cured.

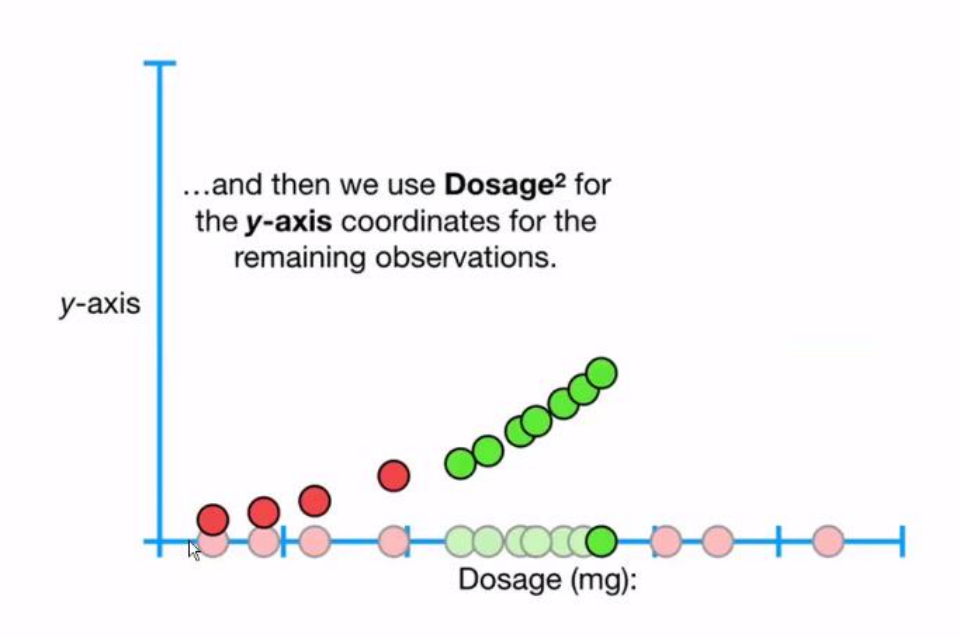
We need to draw the best fit line

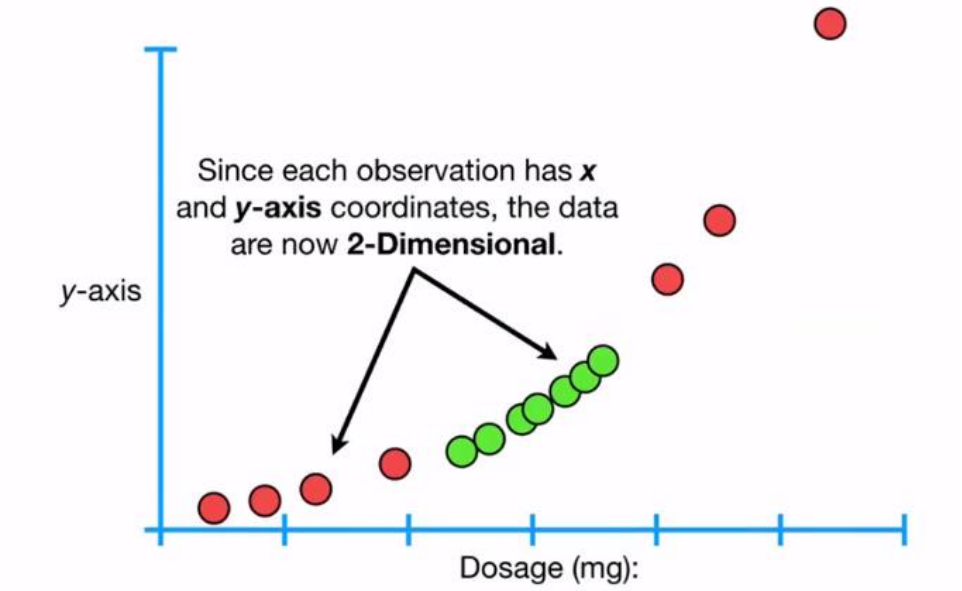


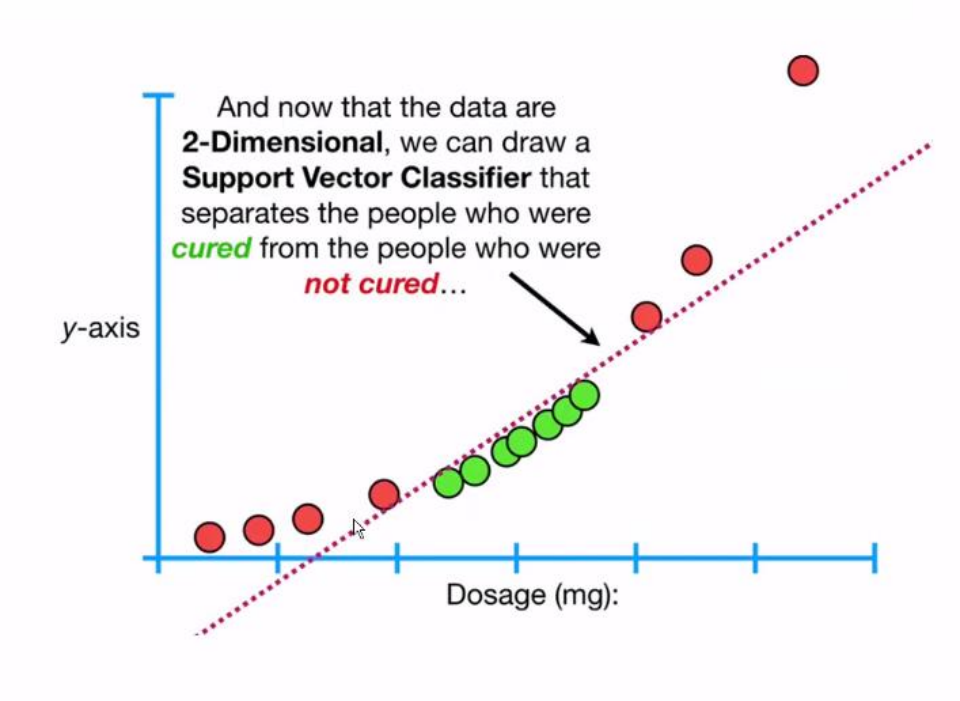


Pull into higher dimension

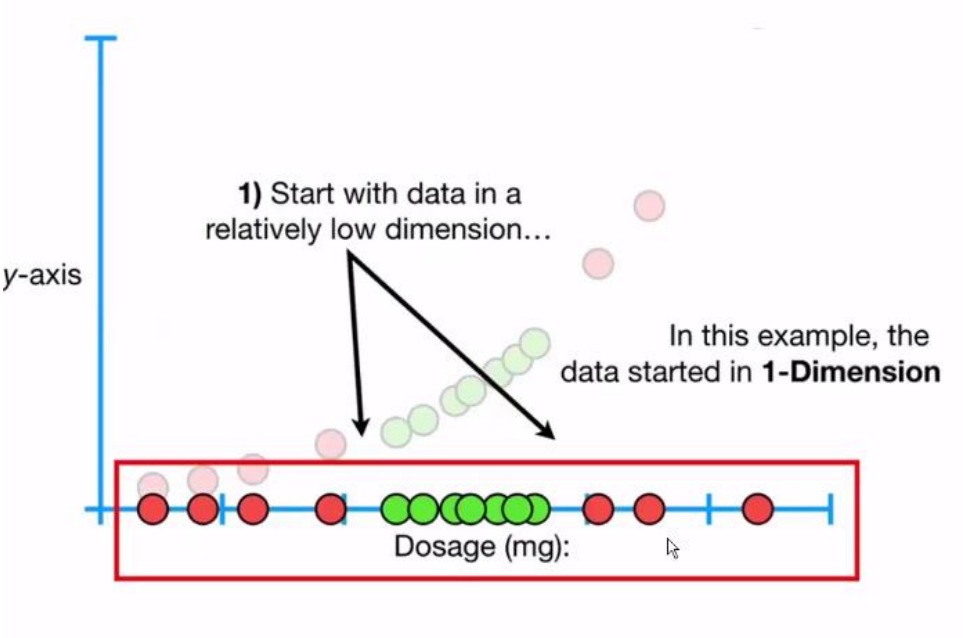


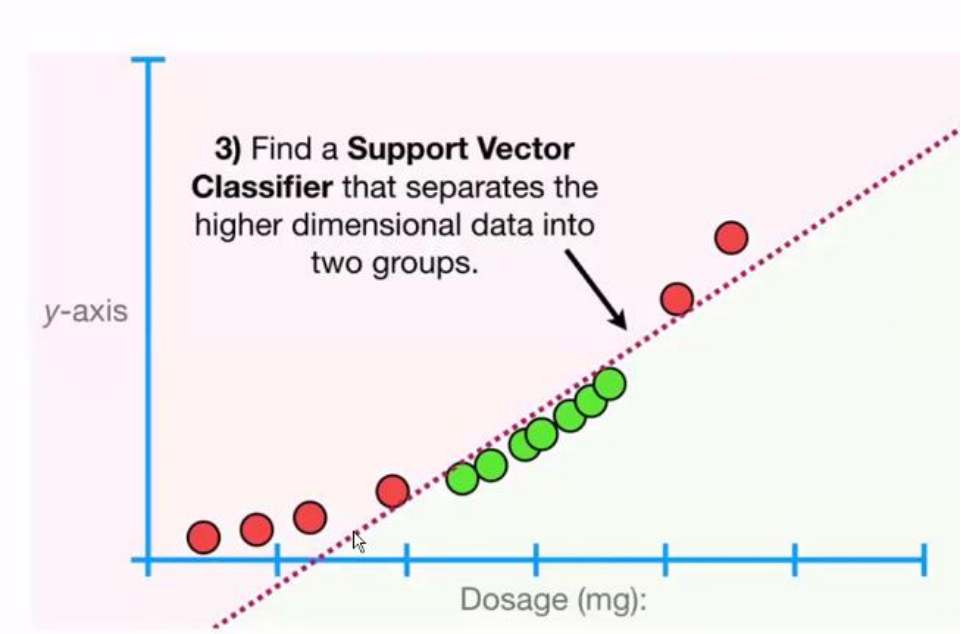

…and then we use **Dosage²** for the ***y*-axis** coordinates for the remaining observations.
y-axis
Dosage (mg):

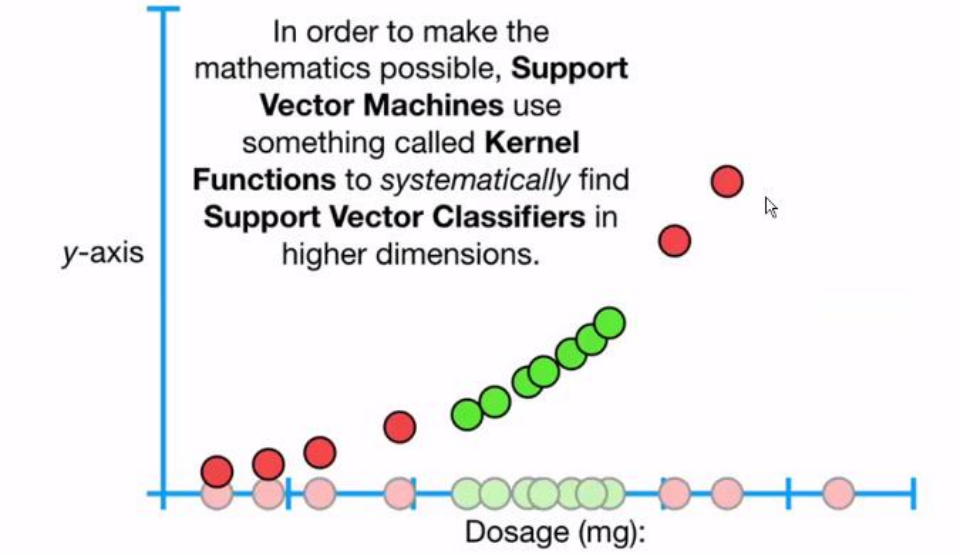

Since each observation has ***x*** and ***y*-axis** coordinates, the data are now **2-Dimensional**.
y-axis
Dosage (mg):

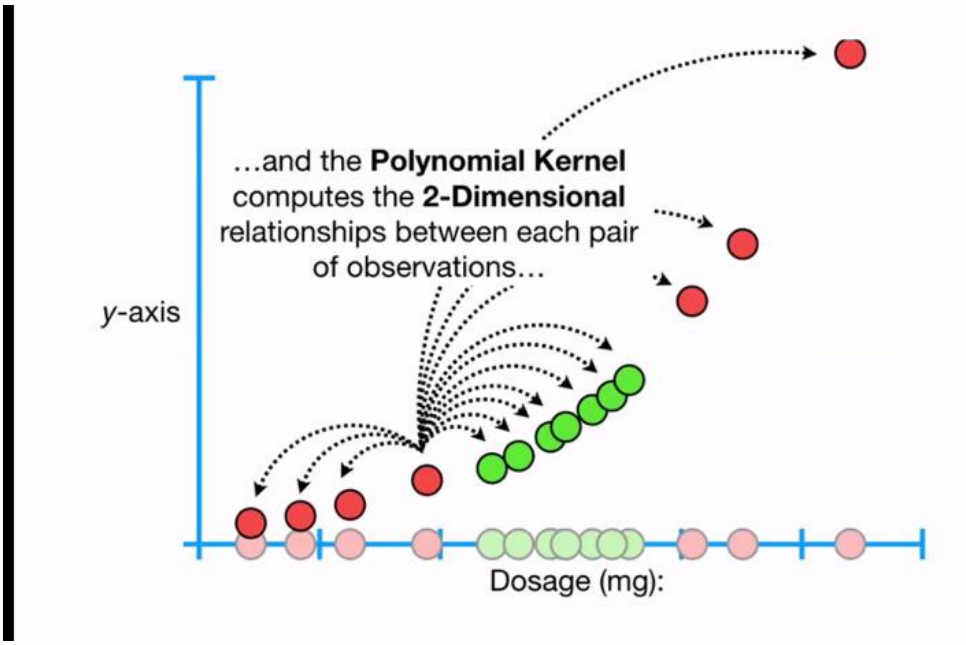


Taking the points which are very boarder of it. We will try to draw the lines (hyperplane)




3) Find a **Support Vector Classifier** that separates the higher dimensional data into two groups.
y-axis
Dosage (mg):


In order to make the mathematics possible, **Support Vector Machines** use something called **Kernel Functions** to *systematically* find **Support Vector Classifiers** in higher dimensions.
y-axis
Dosage (mg):



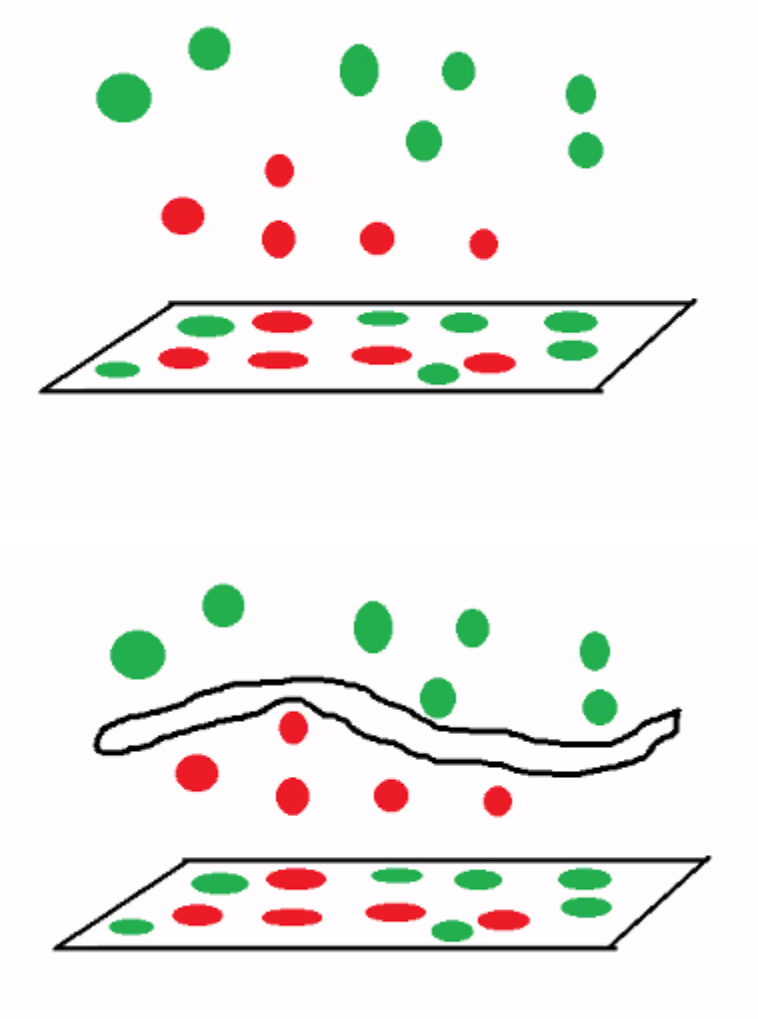
Another very commonly used **Kernel** is the **Radial Kernel**, also known as the **Radial Basis Function (RBF) Kernel**.

Radial Kernel finds **Support Vector Classifiers** in *infinite dimensions*

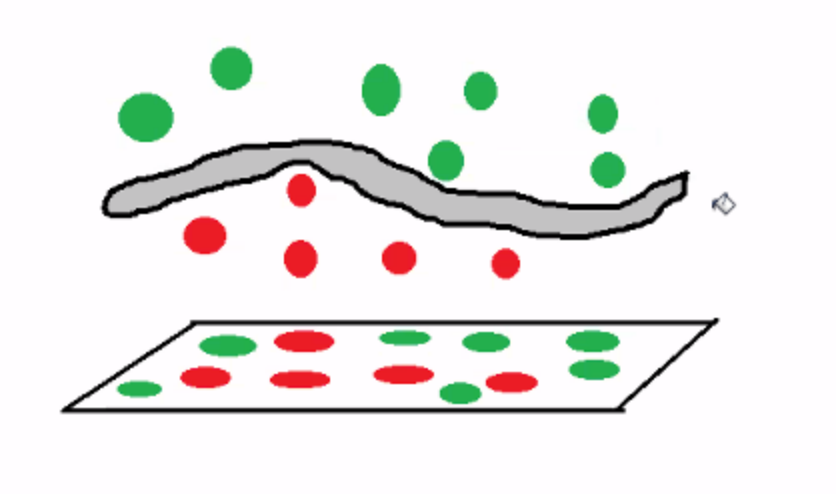
Datapoints:

Pull them into higher dimension space:

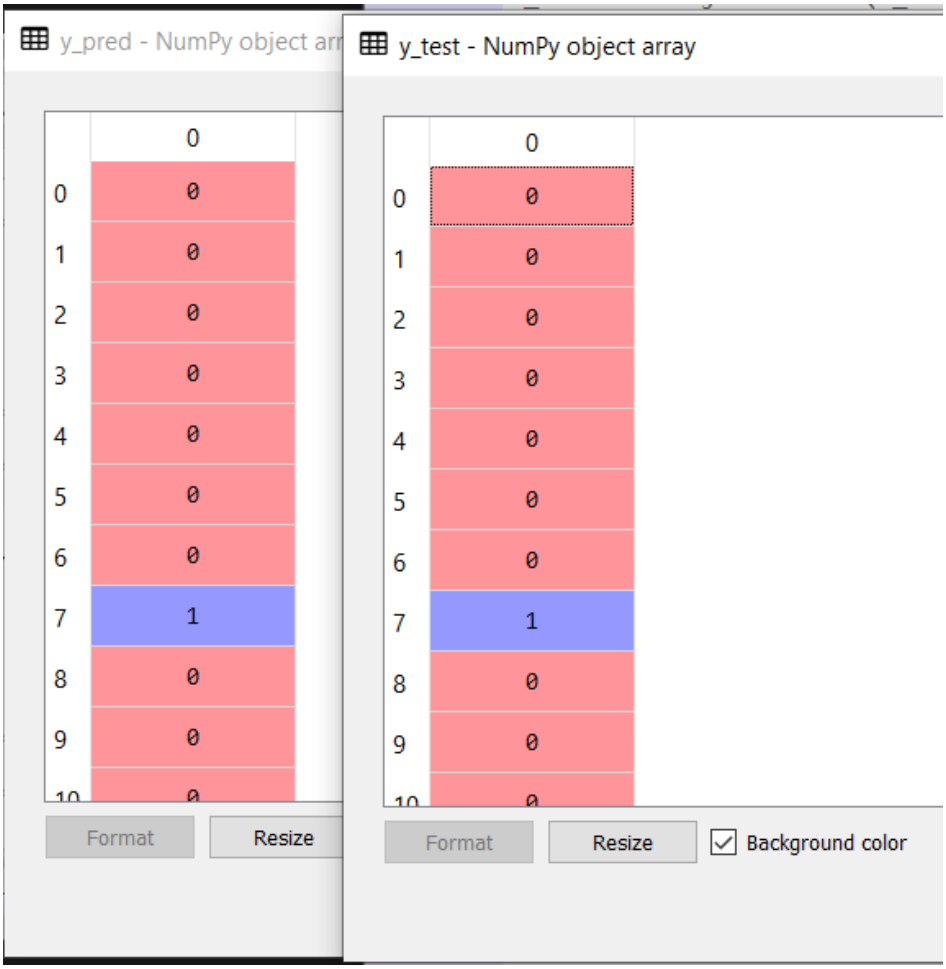
Floating in the air



Radial basis function: (RDF)



Default kernel is RBF.

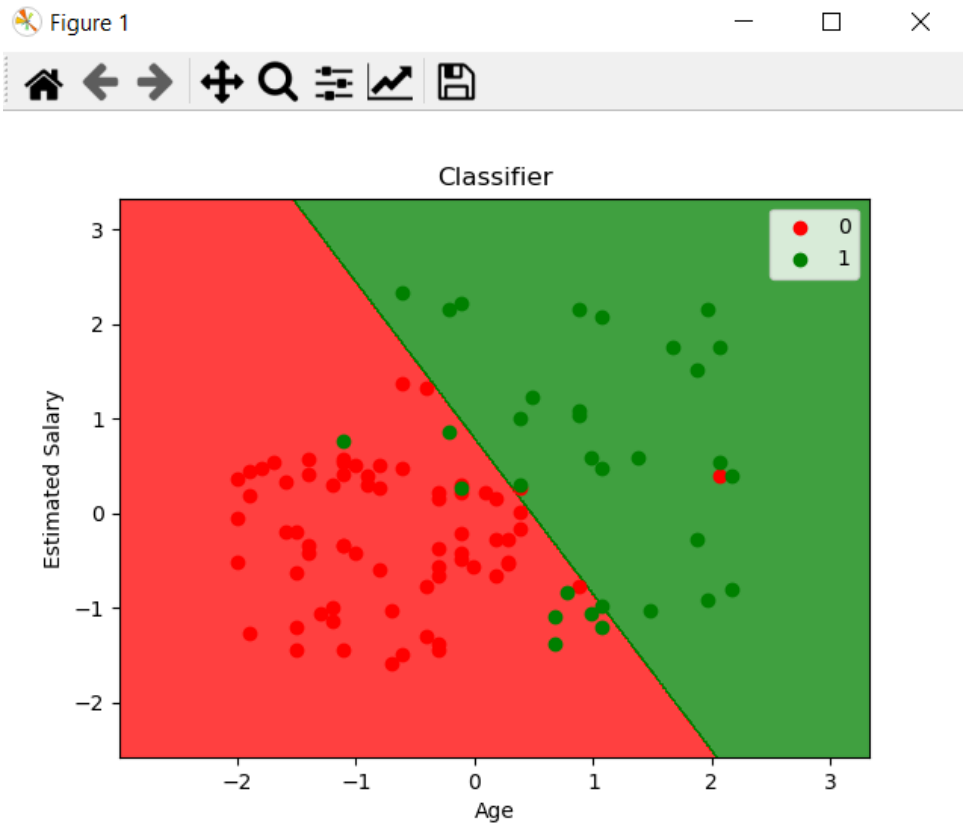


Accuracy:

acc	float64	1	0.9
classifier	svm._classes.SVC	1	SVC object of sklearn.svm._classes module

Visualization:

Linear:



Polynomial with degree 5:

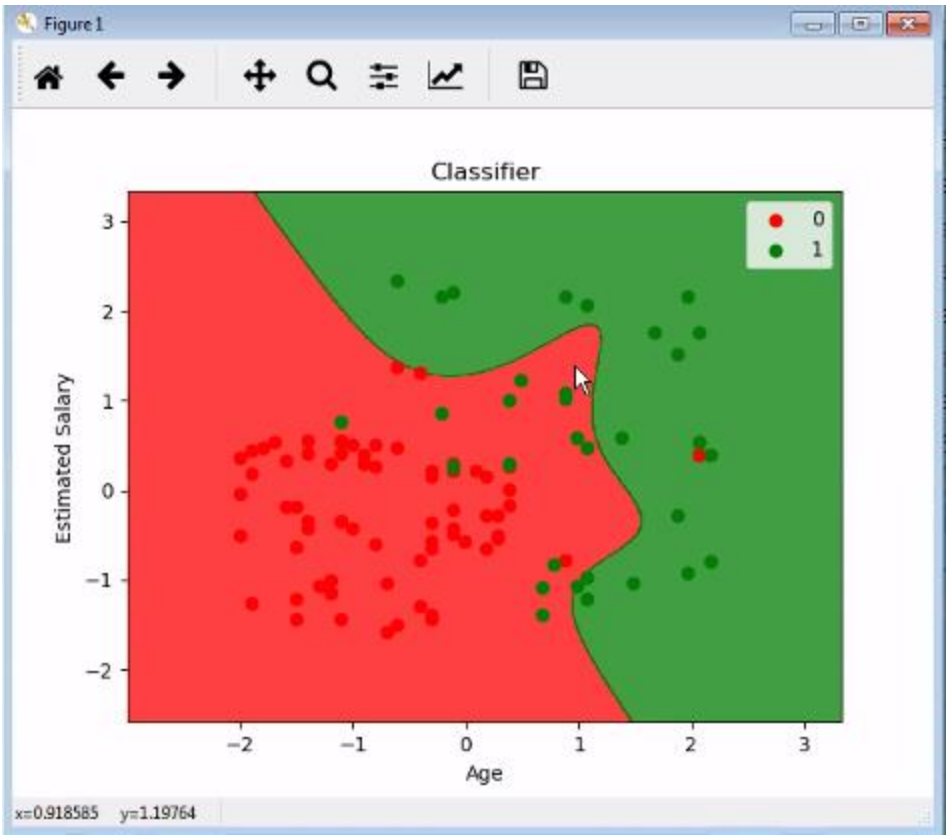


Figure 1
Classifier
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3
x=0.918585 y=1.19764

RBF:

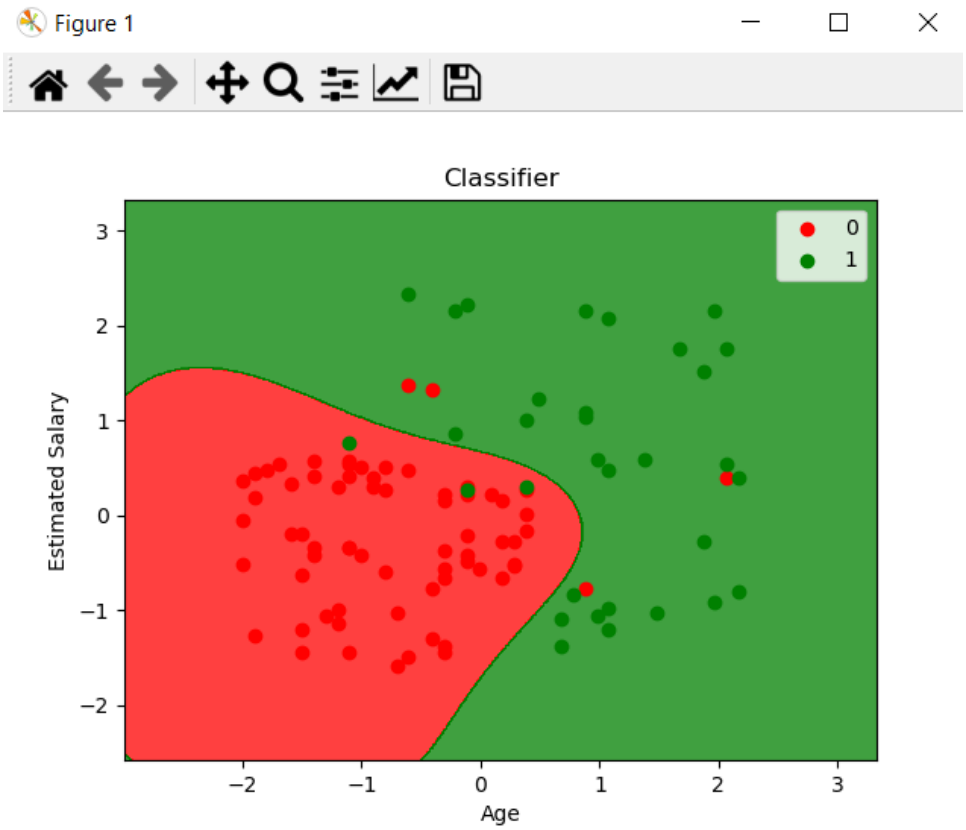


Figure 1
Classifier
0
1
Estimated Salary
Age
3
2
1
0
−1
−2
−2
−1
0
1
2
3

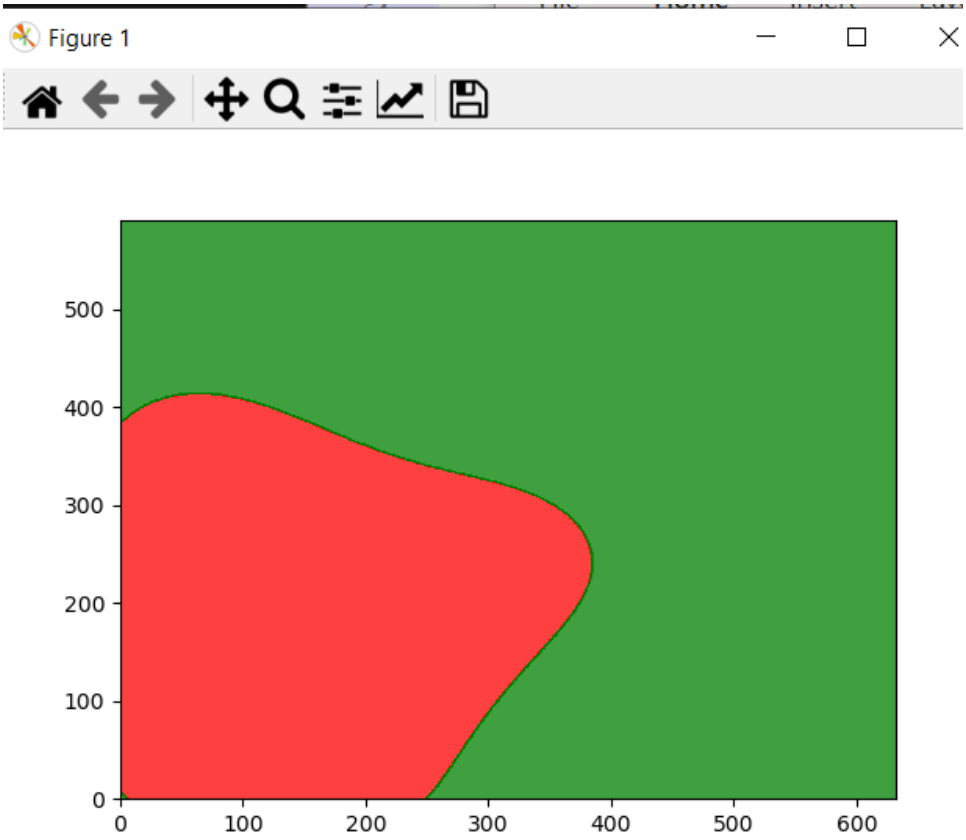
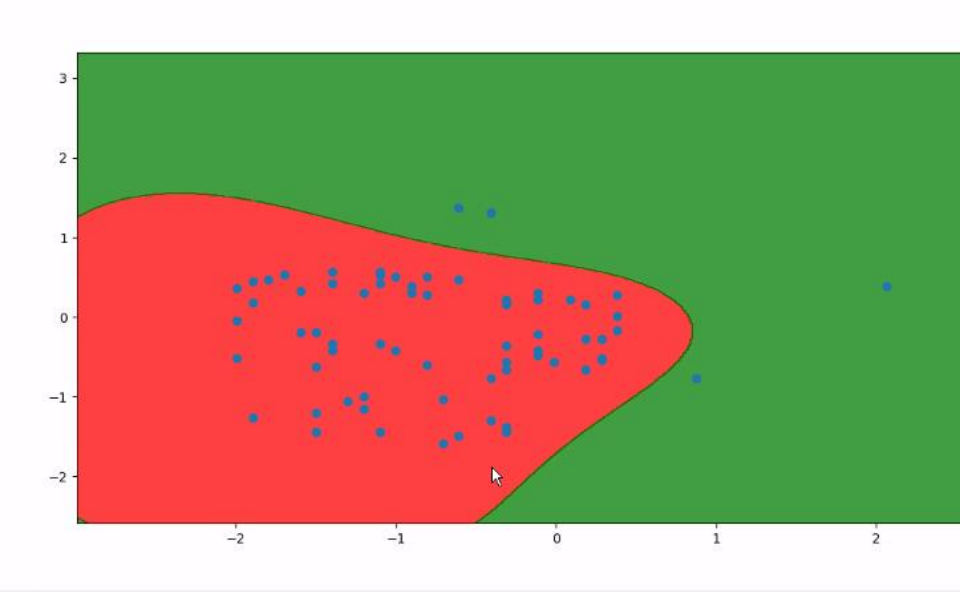


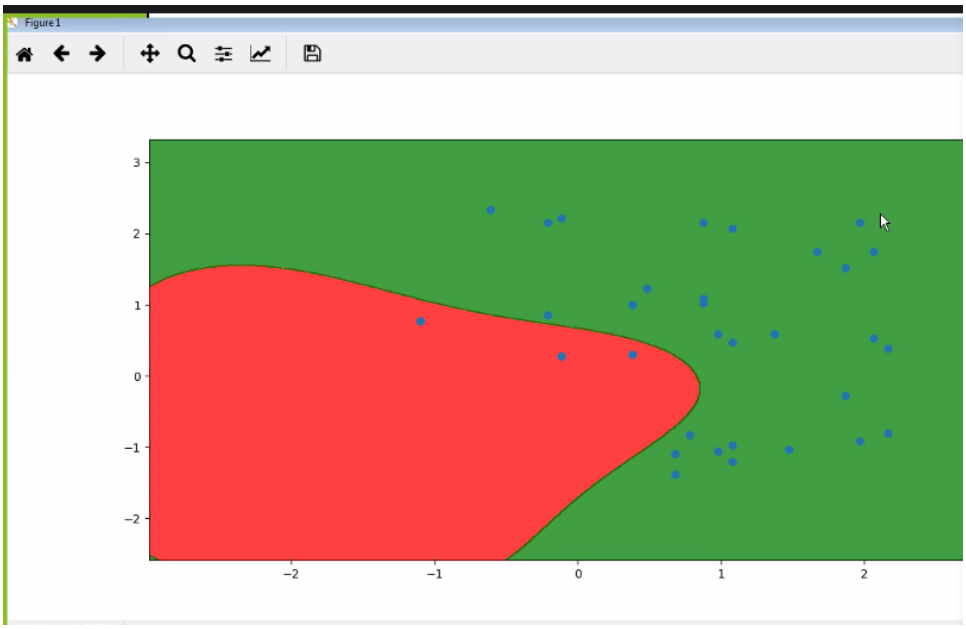
Figure 1
500
400
300
200
100
0
0
100
200
300
400
500
600

False cases:

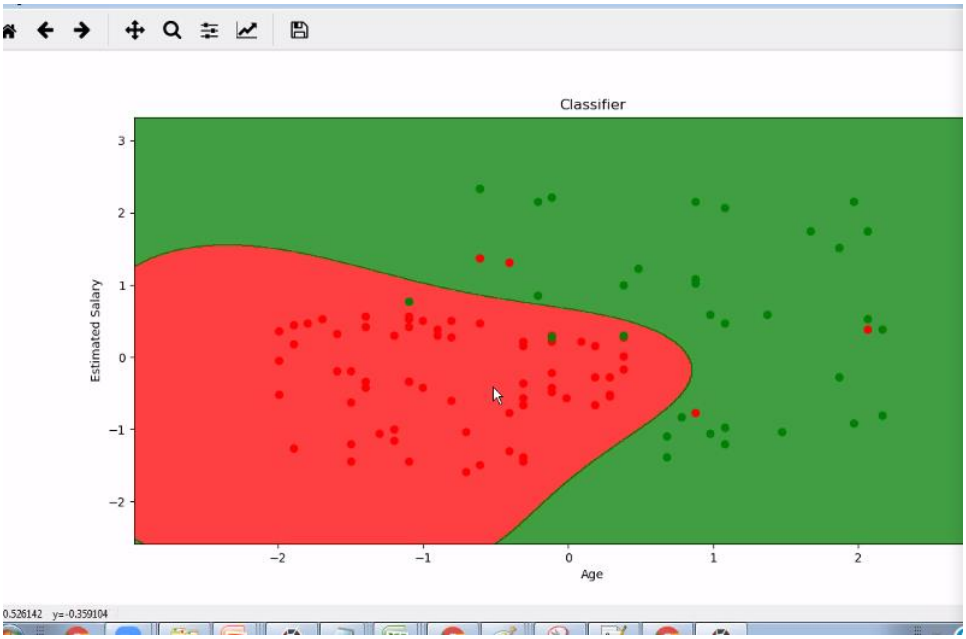



```
X1, X2 = np.meshgrid(np.ara ge(start=X_set[:,0].min()-1,stop=X_set[:,1].max()+1,step=0 01),
                     np.ara ge(start=X_set[:,1].min()-1,stop=X_set[:,1].max()+1,step=0 01))

plt.contourf(X1,X2,
             classifier.pre ict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape)
             alpha = 0.75 , cmap = ListedColormap(('red','green')))

plt.scatter(X_test[y_test== ,0],X_test[y_test==1,1])
plt.show()

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i, j in enumerate(np.un que(y_set)):
    plt.scatter(X_set[y_set == j , 0], X_set[y_set == j, 1],
                c=ListedCol rmap(('red','green'))(i), label = j)
plt.title('Classifier')
plt.xlabel('Age')
plt.ylabel('Estimated Salar ')
plt.legend()
plt.show()
```

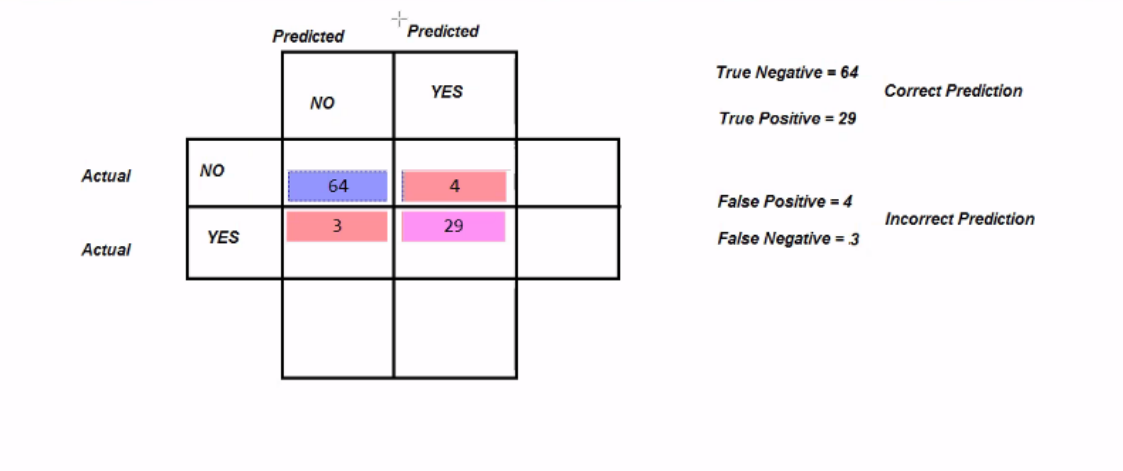
True cases:



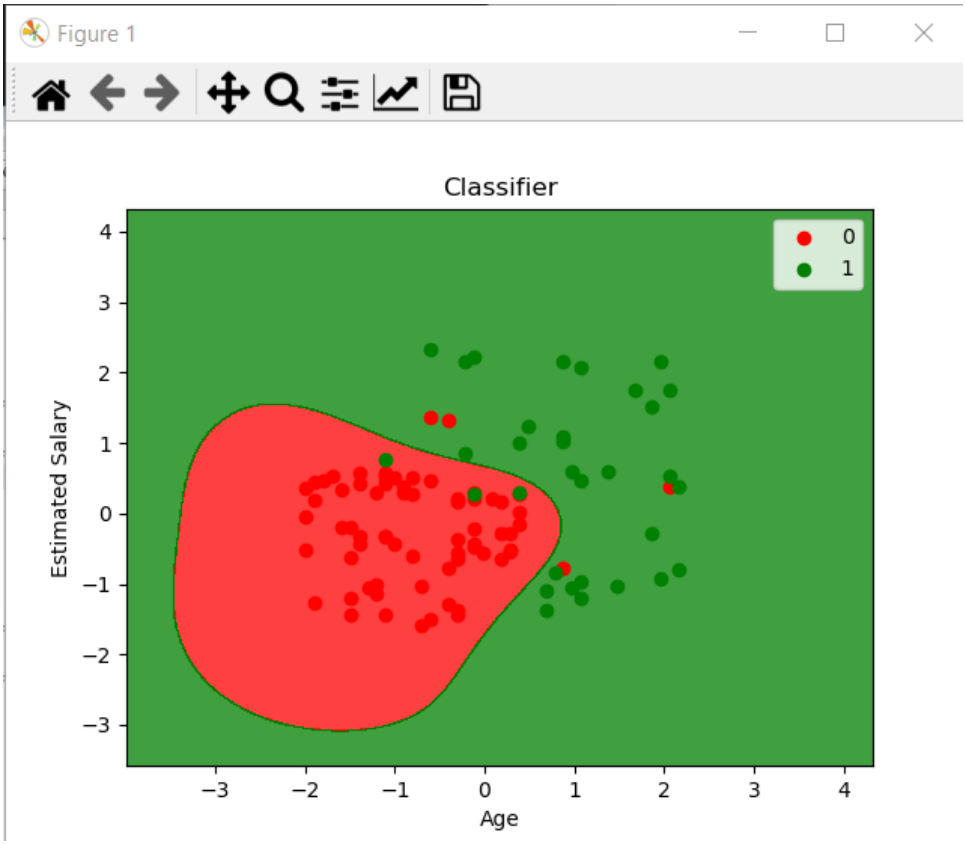
All cases:

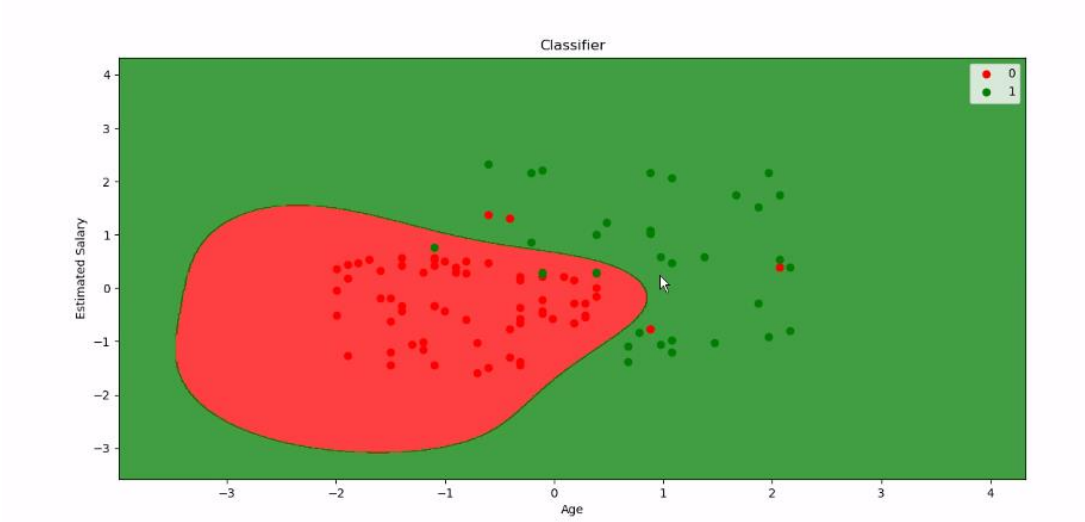


Confusion matrix:



Increased the scale:

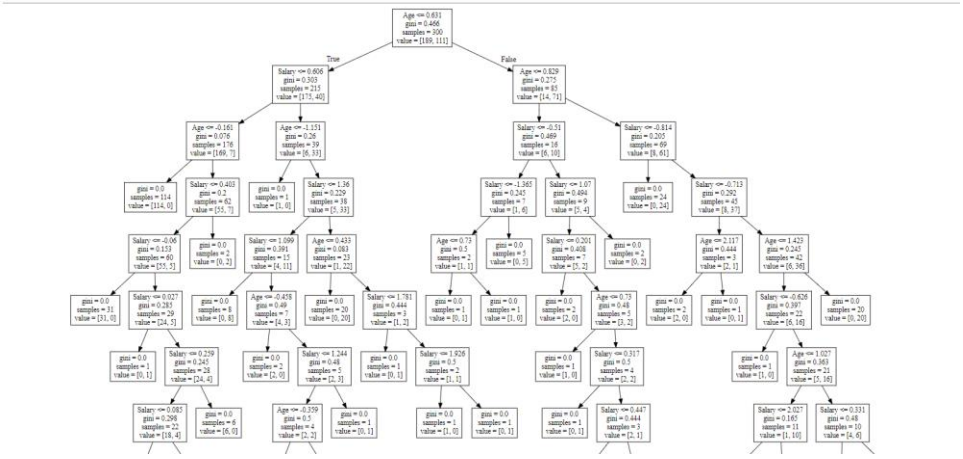




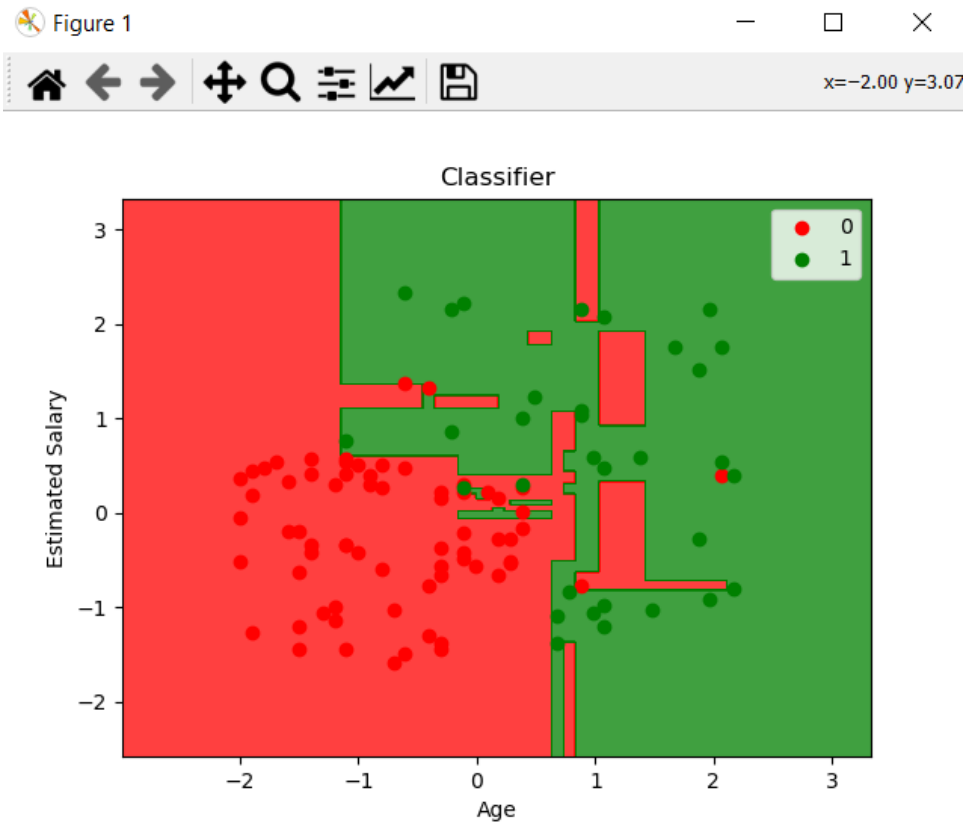

```
from matplotlib.colors import ListedColormap

X_set, y_set = X_test,y_test

X1, X2 = np.meshgrid(np.arange(start=X_set[:,0].min()-1,stop=X_set[:,1].max()+1,step=0.01),
                     np.arange(start=X_set[:,1].min()-1,stop=X_set[:,1].max()+2,step=0.01))

plt.contourf(X1,X2,
             classifier.predict(np.array([X1.ravel(),X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75 , cmap = ListedColormap(('red','green')))

plt.xlim(X1.min(),X1.max())
plt.ylim(X2.min(),X2.max())

for i, j in enumerate(np.unique(y_set)):
```

Decision Tree Classifier:

Impurity calculation tells how well it perform no and yes response – Gini index

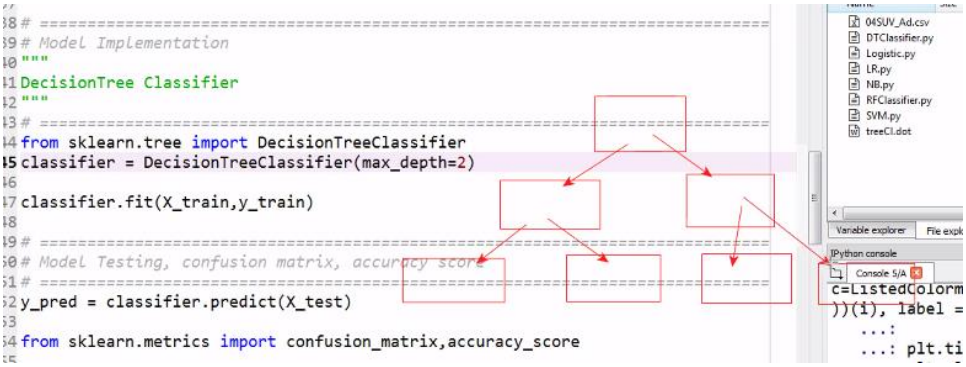


Visualization:

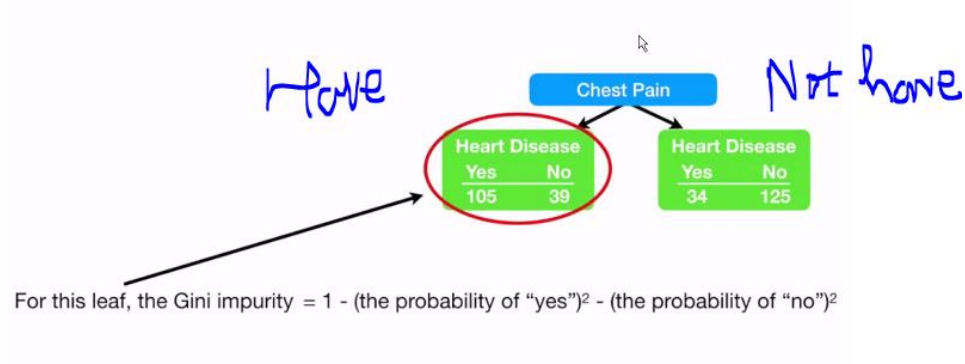


Boundary is not generalized. To control this, we will have minimum split => max depth

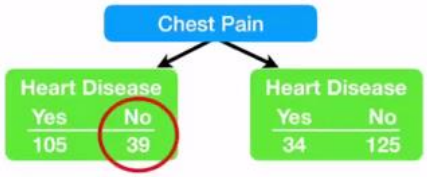
Control the depth here



Gini impurity calculation:

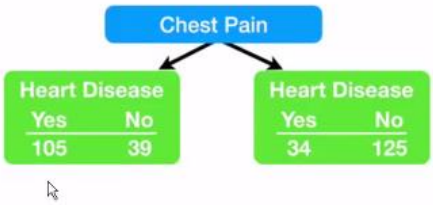


For this leaf, the Gini impurity = 1 - (the probability of "yes")^2 - (the probability of "no")^2



For this leaf, the Gini impurity = 1 - (the probability of "yes")^2 - (the probability of "no")^2

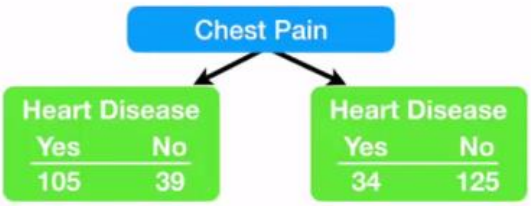
$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$



For this leaf, the Gini impurity = 1 - (the probability of "yes")^2 - (the probability of "no")^2

$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$

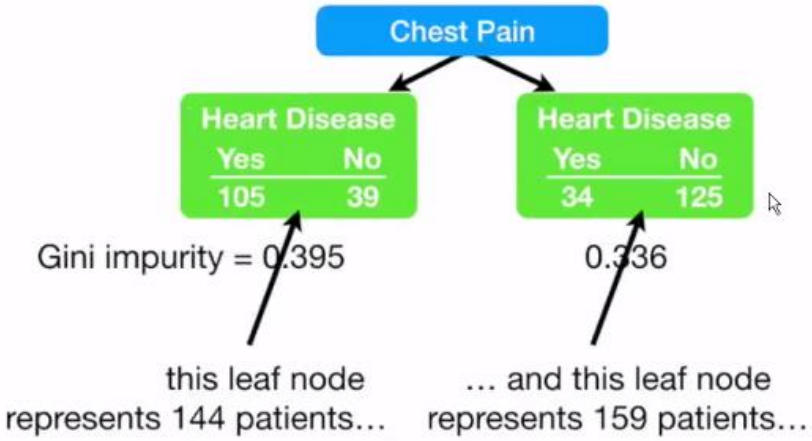
$$= 0.395$$



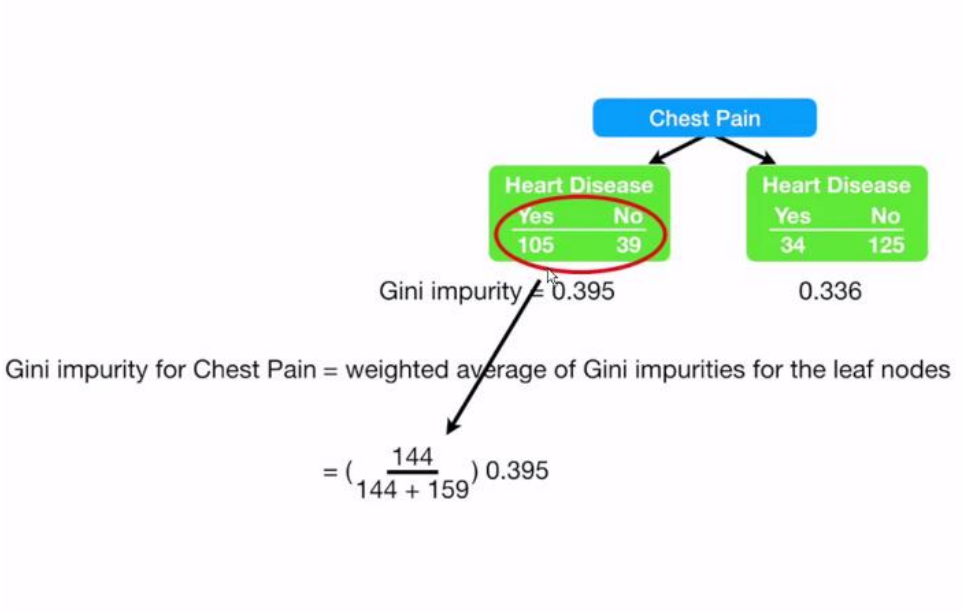
= 1 - (the probability of "yes")^2 - (the probability of "no")^2

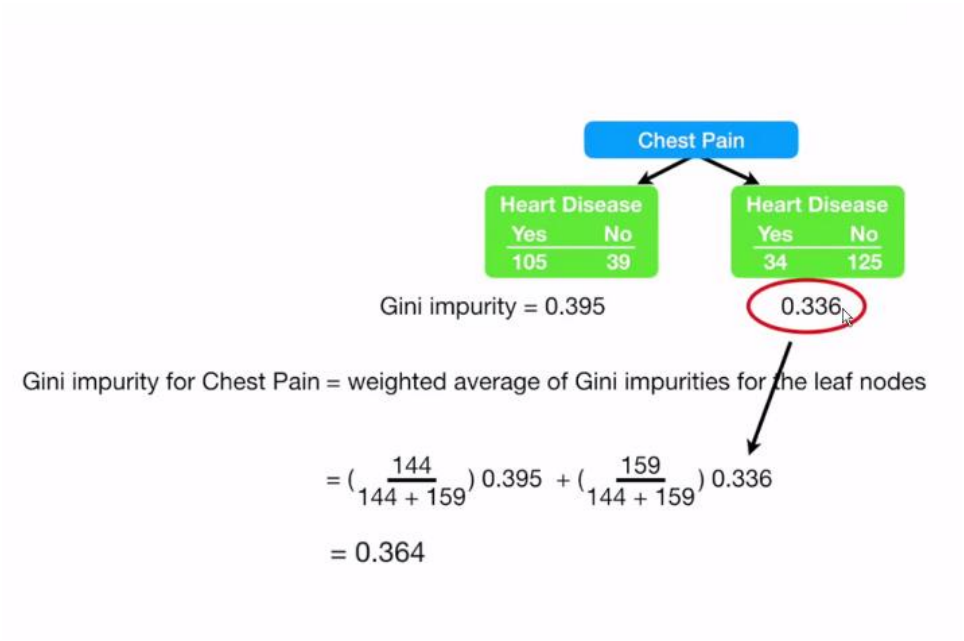
$$= 1 - \left(\frac{34}{34 + 125}\right)^2 - \left(\frac{125}{34 + 125}\right)^2$$

= 0.336

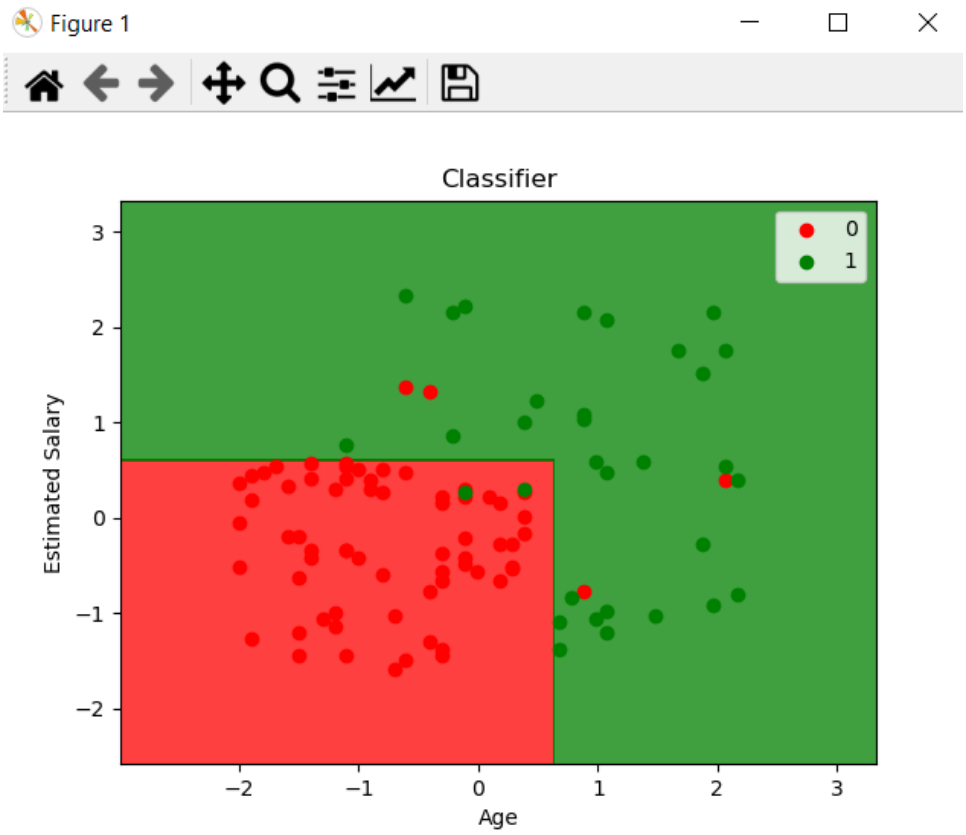


…the leaf nodes do not represent the same number of patients.



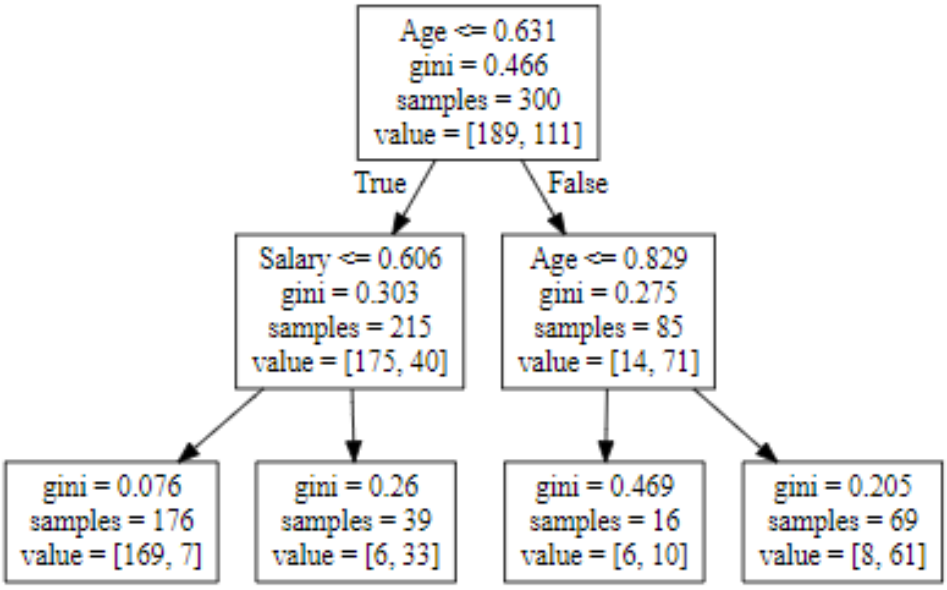


Increase the max depth = 2:



Boundary seems to be generalized.

Good decision tree:



Classification Value to Predict is Discrete

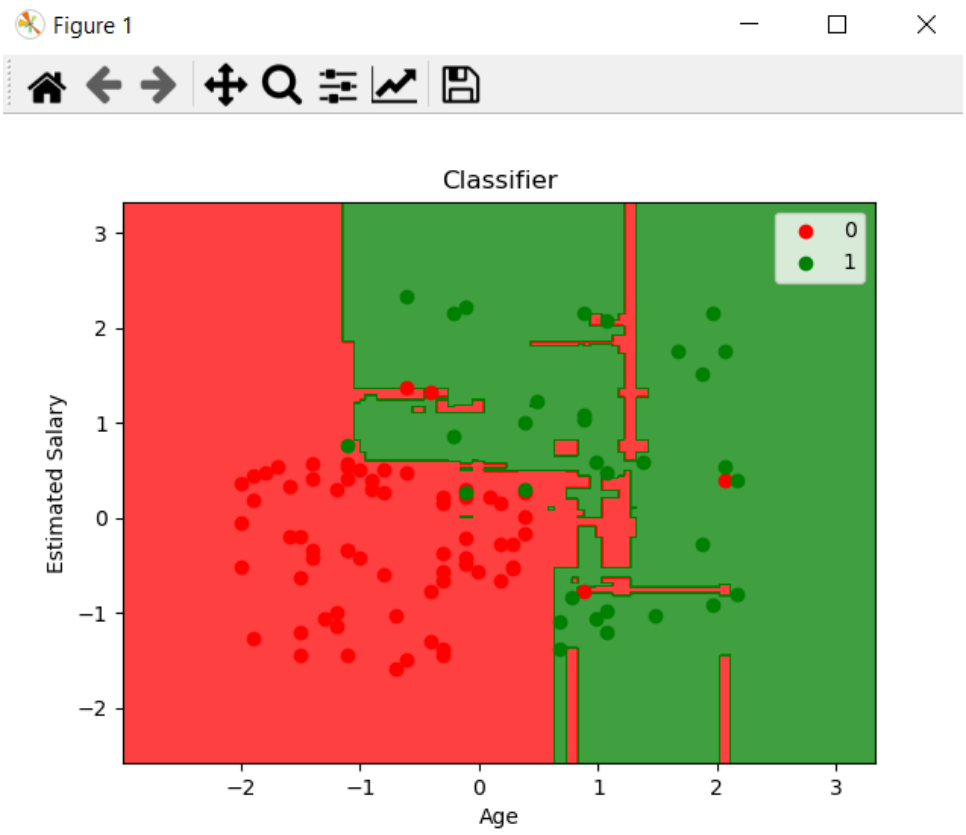
Logistic Regression		
Support Vector Machine	93%	
DecisionTree Classifier	94%	max_depth=2
RandomForest Classifier		
Naive Bayes		

Random Forest Classifier:

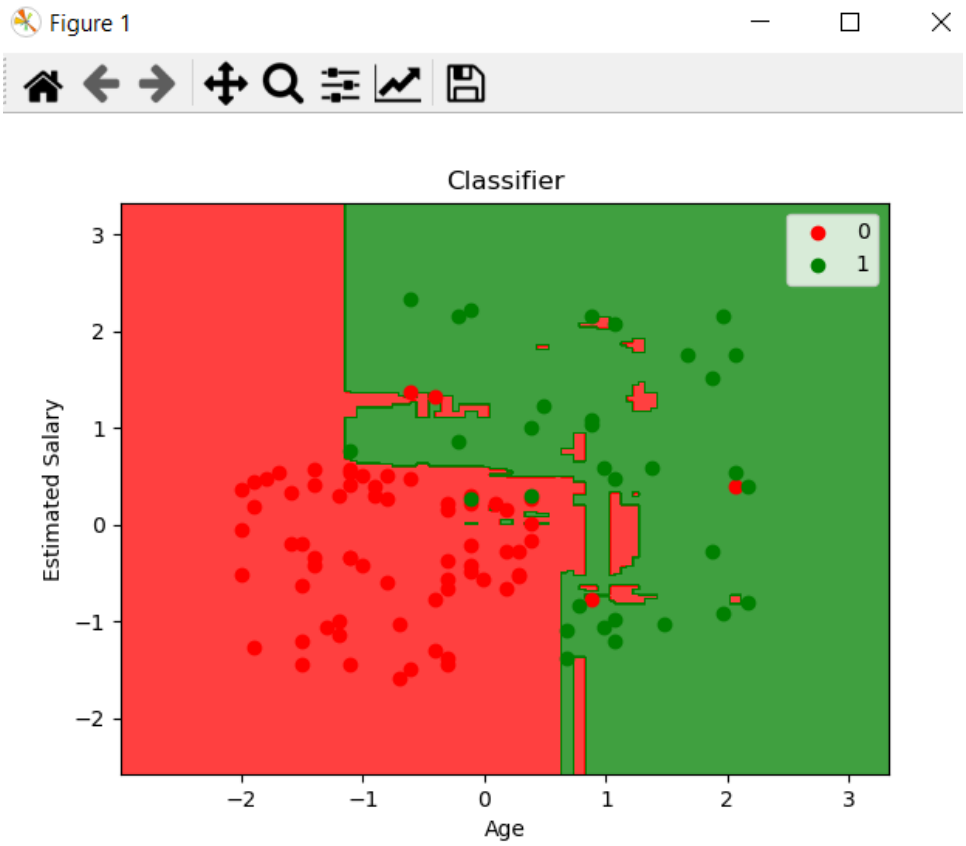
Name	Type	Size	Value
acc	float64	1	0.92
classifier	ensemble._forest.RandomForestClassifier	10	RandomForestClassif…

Visualization:

Overfitting issue!



Change it to 100 trees:

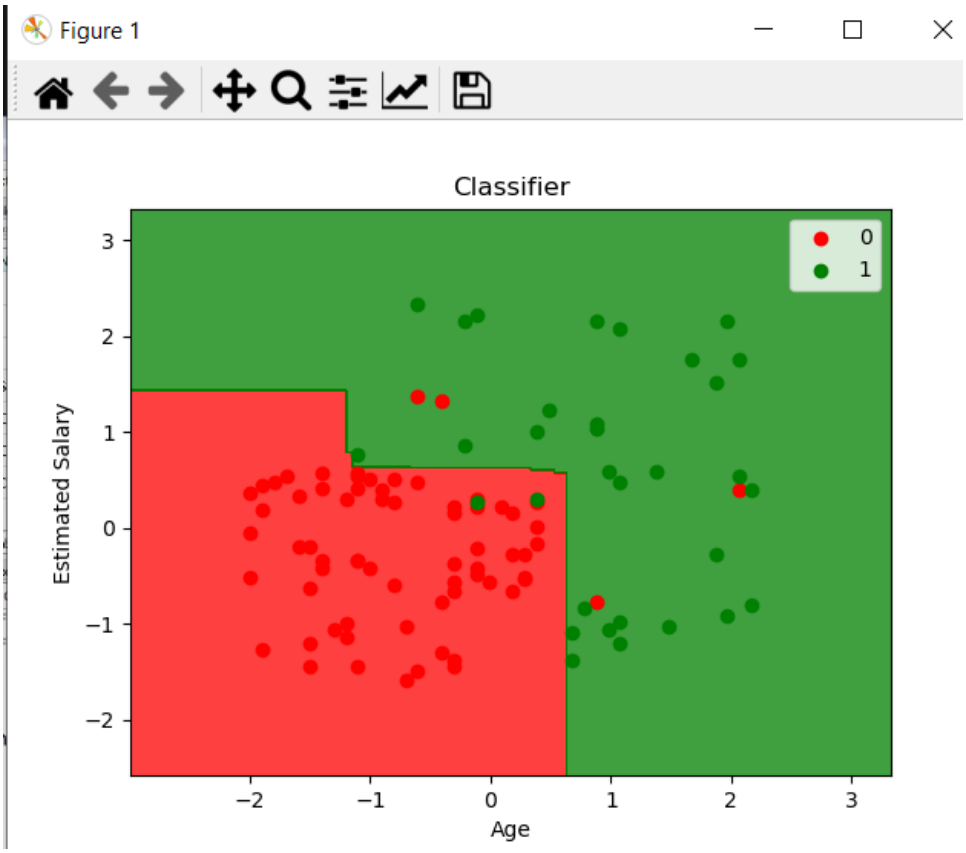


Name	Type	Size	Value
acc	float64	1	0.92
classifier	ensemble._forest.RandomForestClassifier	100	RandomForestClass

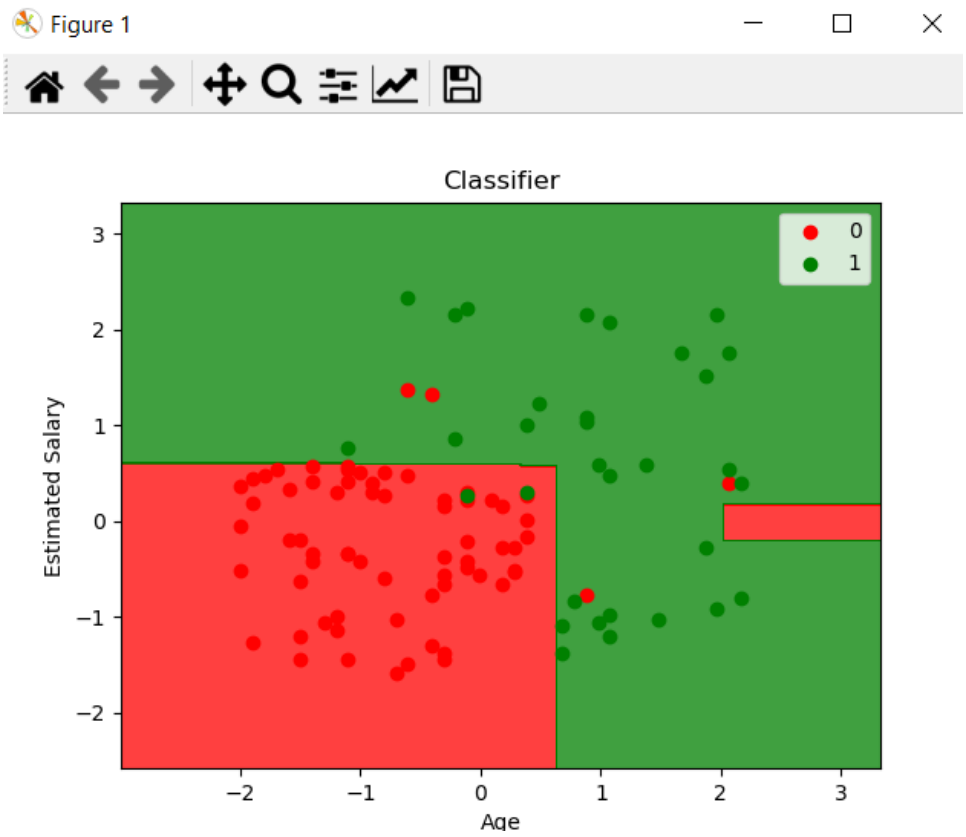
Not helping! Same accuracy

Try to include max_depth=2:

With 100 trees:



With 10 trees:



Name	Type	Size	Value
acc	float64	1	0.94
classifier	ensemble._forest.RandomForestClassifier	100	RandomForestClass…

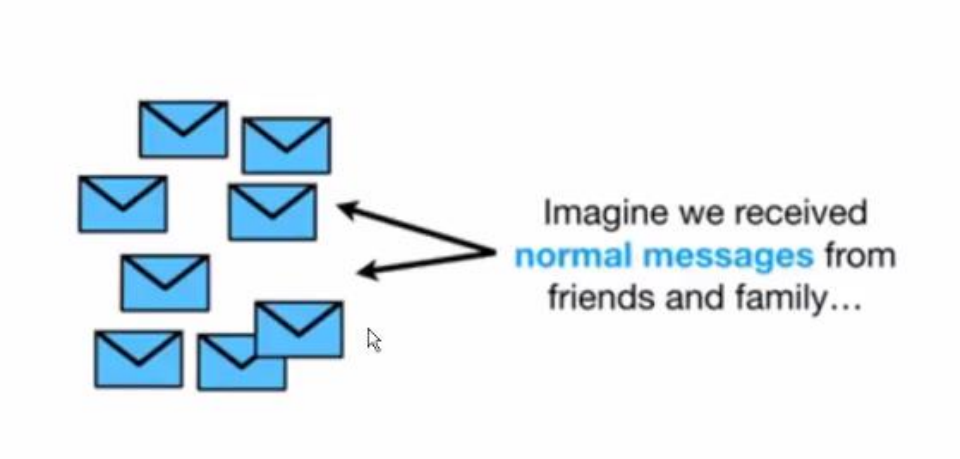
Name	Type	Size	Value
acc	float64	1	0.94
classifier	ensemble._forest.RandomForestClassifier	10	RandomForestClass…

Classification Value to Predict is Discrete

Logistic Regression		
Support Vector Machine	93%	
DecisionTree Classifier	94%	max_depth=2
RandomForest Classifier	94%	max_depth=2 ,Trees=10
Naive Bayes		

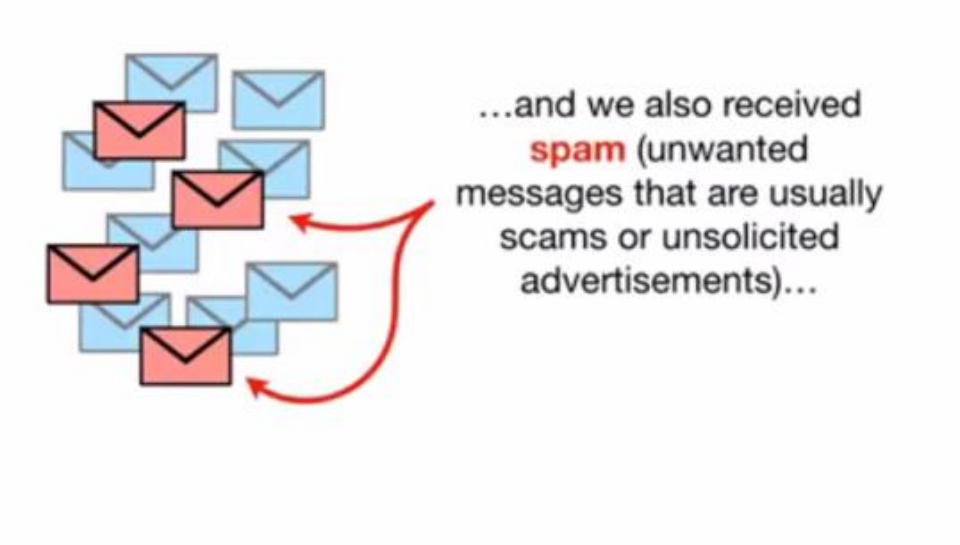
Naïve Bayes:

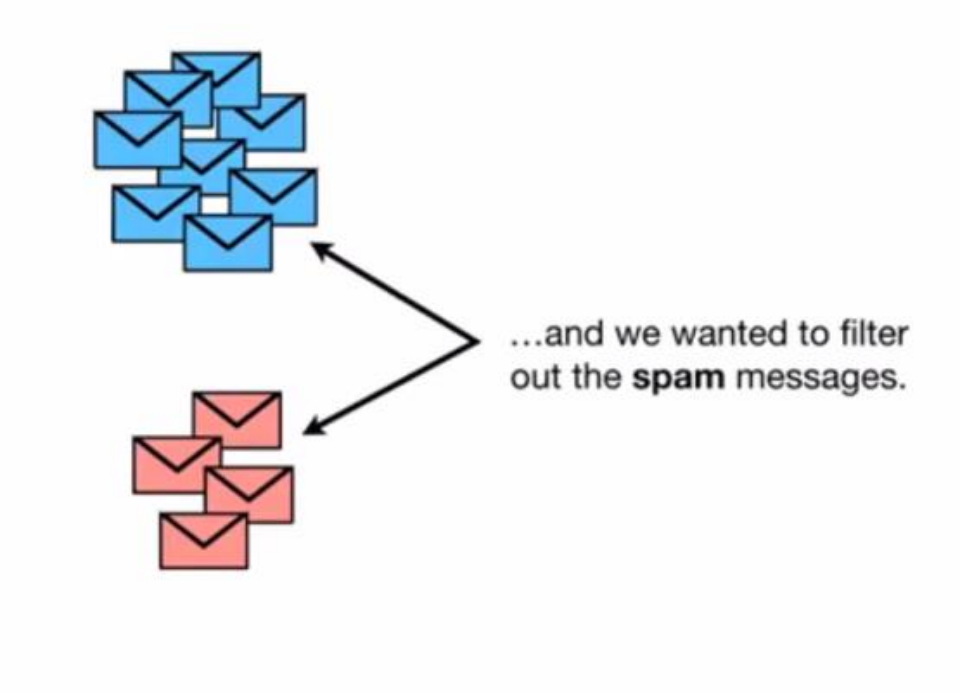
Learned based on probabilities.



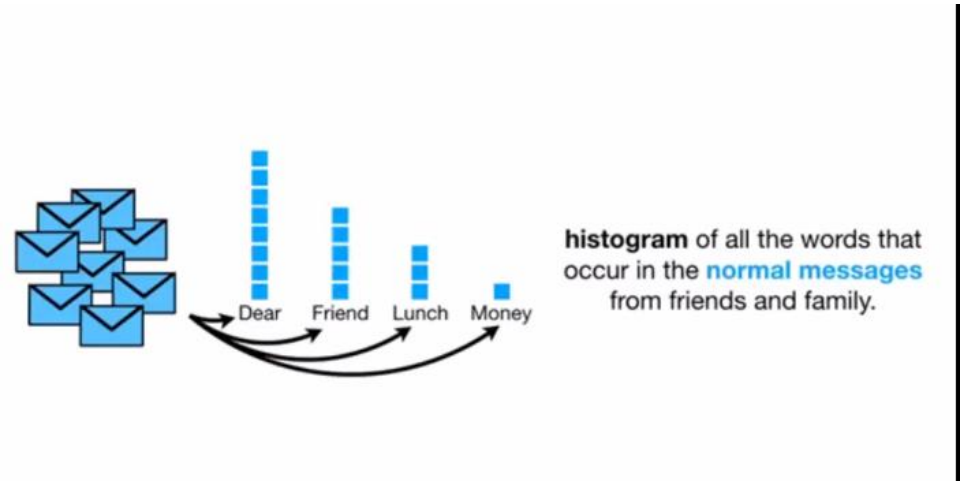
Create spam filter

Segreate normal and spam messages.

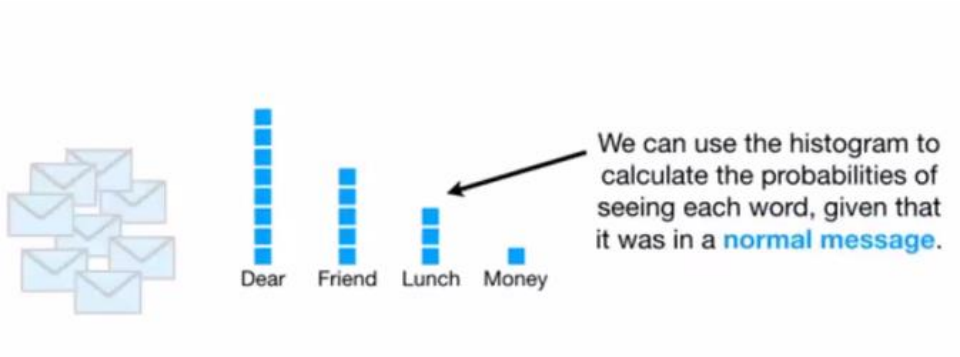


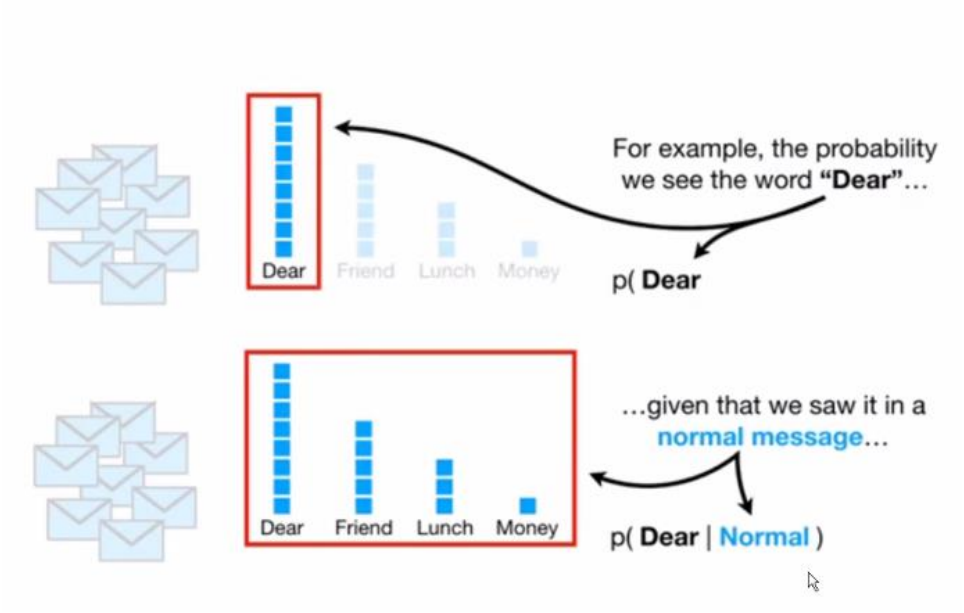


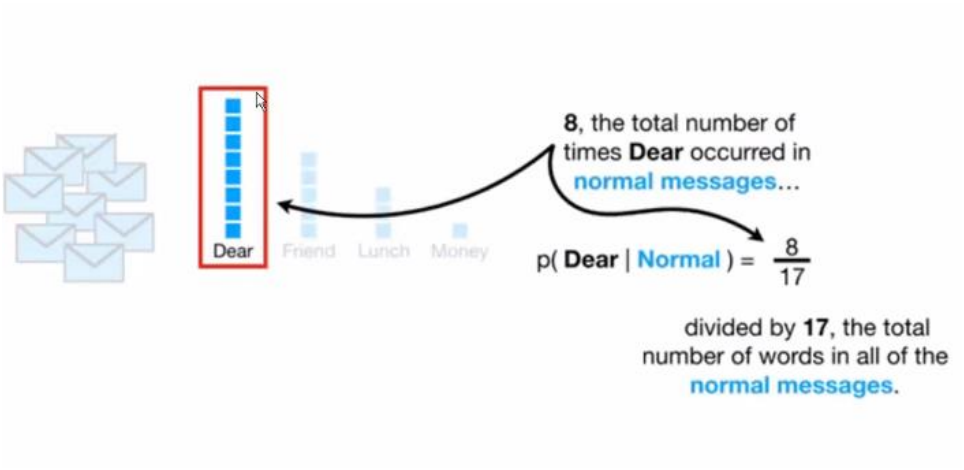
Find probabilities for each words:

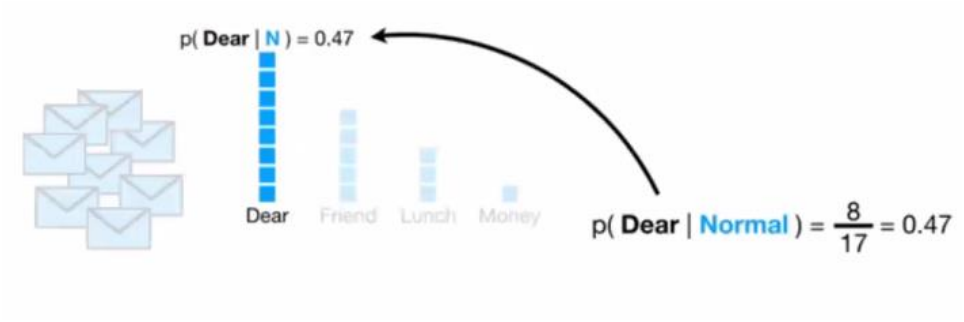


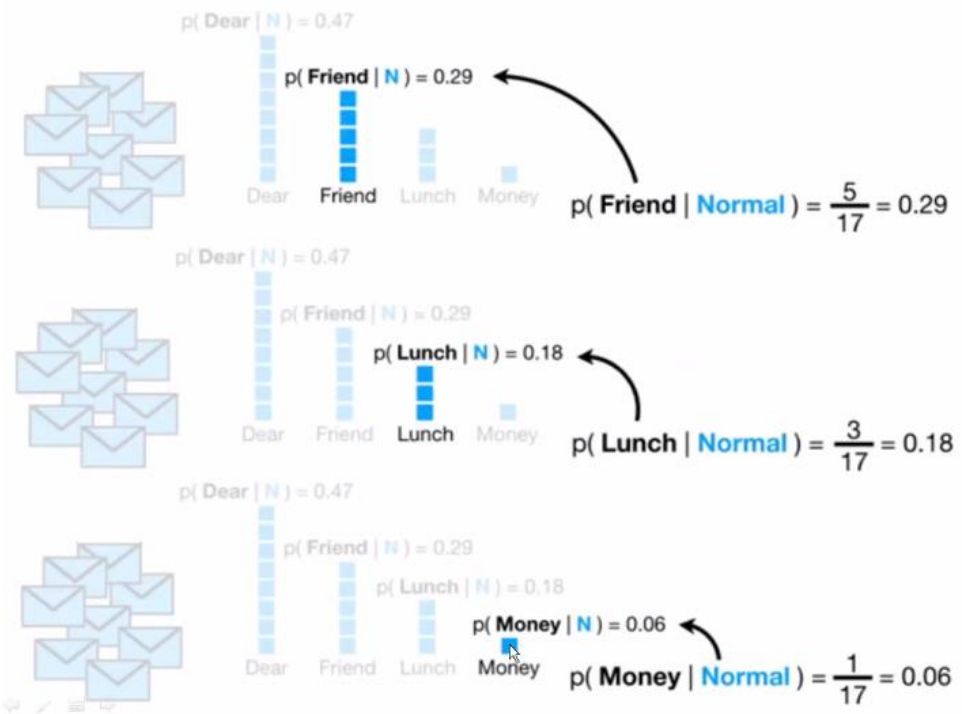
Find the probabilities of words given the visibilities of normal messages



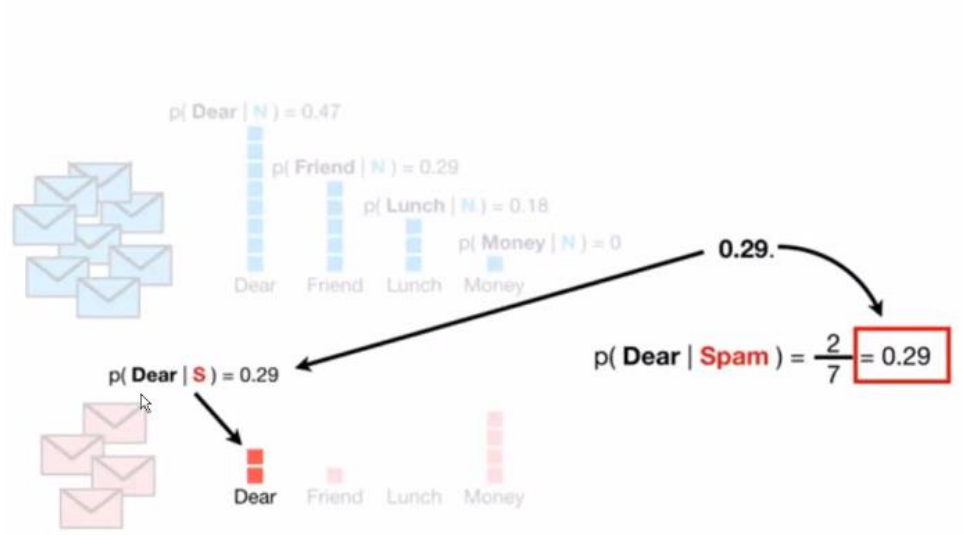

Dear
Friend
Lunch
Money
For example, the probability we see the word "Dear"...
p(Dear
Dear
Friend
Lunch
Money
...given that we saw it in a normal message...
p(Dear | Normal)

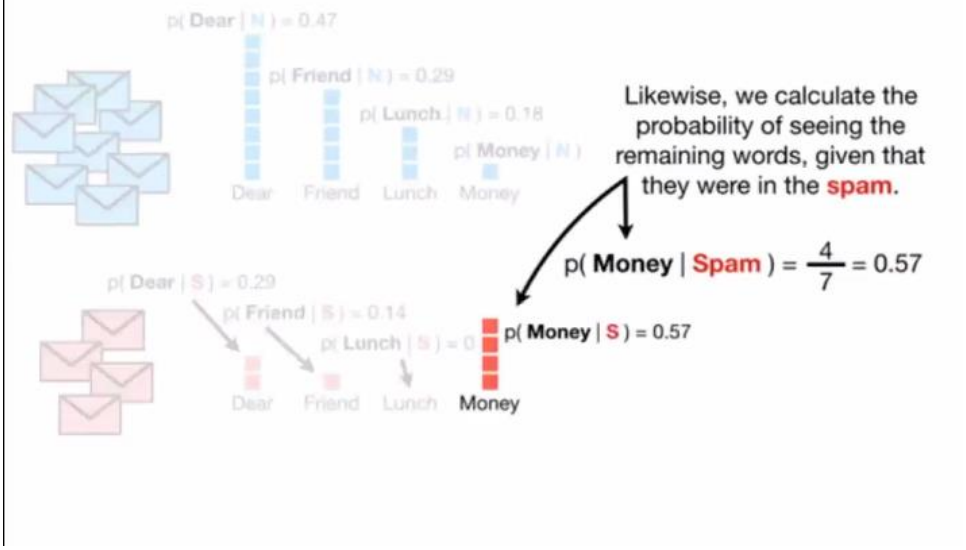

Dear
Friend
Lunch
Money
8, the total number of times Dear occurred in normal messages...
p(Dear | Normal) = 8/17
divided by 17, the total number of words in all of the normal messages.


p(Dear | N) = 0.47
Dear
Friend
Lunch
Money
p(Dear | Normal) = 8/17 = 0.47


p(Dear | N) = 0.47
p(Friend | N) = 0.29
Dear
Friend
Lunch
Money
p(Friend | Normal) = 5/17 = 0.29
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
Dear
Friend
Lunch
Money
p(Lunch | Normal) = 3/17 = 0.18
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
Dear
Friend
Lunch
Money
p(Money | Normal) = 1/17 = 0.06

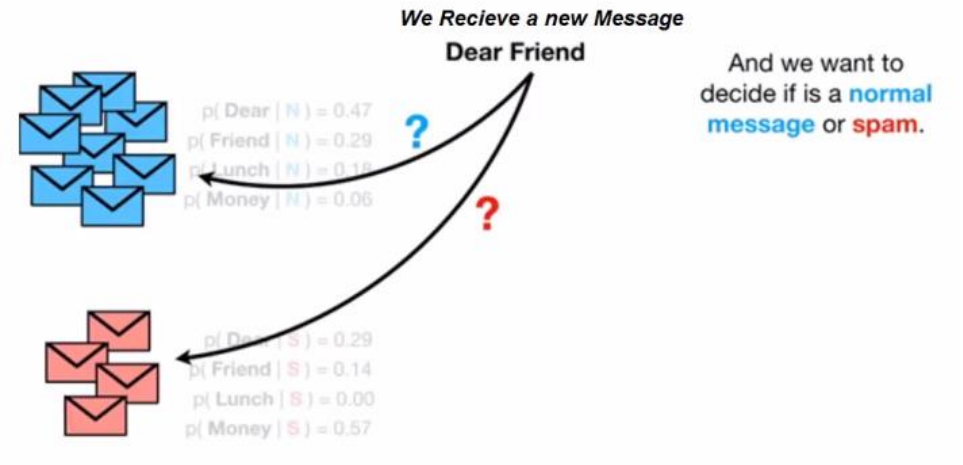

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N)
Dear
Friend
Lunch
Money
Now we make a histogram of all the words that occur in the spam...
Dear
Friend
Lunch
Money

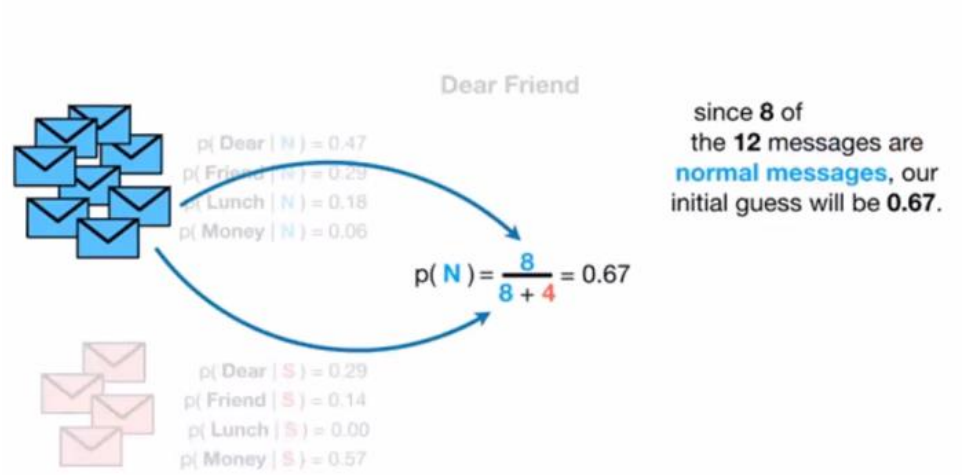

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0
Dear
Friend
Lunch
Money
0.29.
p(Dear | Spam) = 2/7 = 0.29
p(Dear | S) = 0.29
Dear
Friend
Lunch
Money


p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N)
Dear
Friend
Lunch
Money
Likewise, we calculate the probability of seeing the remaining words, given that they were in the spam.
p(Money | Spam) = 4/7 = 0.57
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0
p(Money | S) = 0.57
Dear
Friend
Lunch
Money

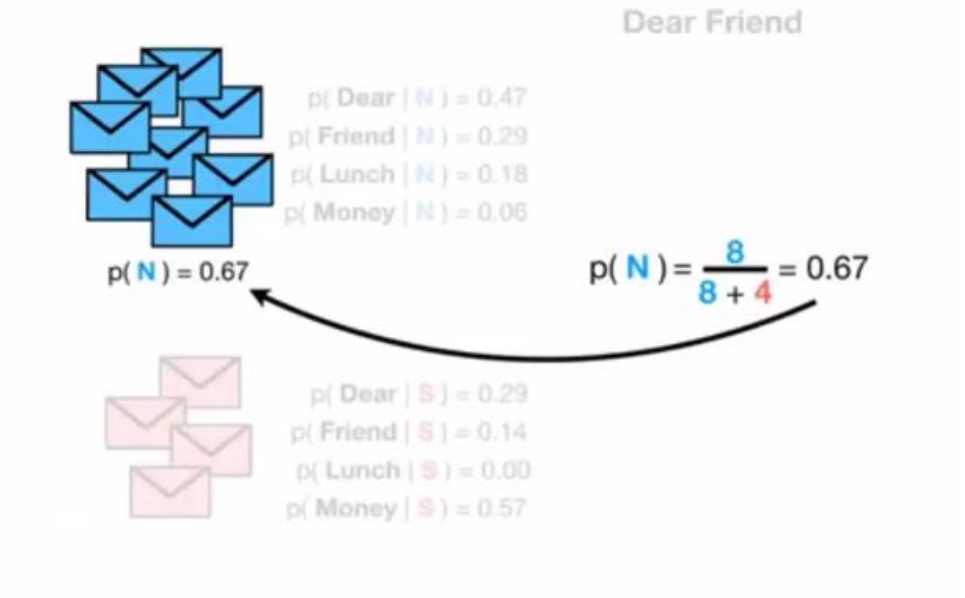
p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57





since **8** of the **12** messages are **normal messages**, our initial guess will be **0.67**.



p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

Now we multiply that initial guess by probability that the word **Dear** occurs in a **normal message**...

...and the probability that the word **Friend** occurs in a **normal message**.

p(N) × p(**Dear** | N) × p(**Friend** | N)

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

Now we multiply that initial guess by probability that the word **Dear** occurs in a **normal message**...

...and the probability that the word **Friend** occurs in a **normal message**.

p(N) × p(**Dear** | N) × p(**Friend** | N)

0.67 × 0.47 × 0.29 = 0.09

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

Now we multiply that initial guess by probability that the word **Dear** occurs in a **normal message**...

...and the probability that the word **Friend** occurs in a **normal message**.

p(N) × p(**Dear** | N) × p(**Friend** | N)

0.67 × 0.47 × 0.29 = 0.09

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

In a simple way, we can think of **0.09** as the score that **Dear Friend** gets if it is a **Normal Message**.

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

Now we multiply that initial guess by probability that the word **Dear** occurs in a **normal message**...

...and the probability that the word **Friend** occurs in a **normal message**.

p(N) x p(**Dear** | N) x p(**Friend** | N)

$0.67 \times 0.47 \times 0.29 = 0.09$ p(N | **Dear Friend**)

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

In a simple way, we can think of **0.09** as the score that **Dear Friend** gets if it is a **Normal Message**.

p(N | Dear Friend) ∝ 0.09 Dear Friend

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

$p(S) = \frac{4}{4+8} = 0.33$

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(N | Dear Friend) ∝ 0.09

Dear Friend

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

p(S) × p(**Dear** | S) × p(**Friend** | S)

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(S) = 0.33

p(N | Dear Friend) ∝ 0.09

Dear Friend

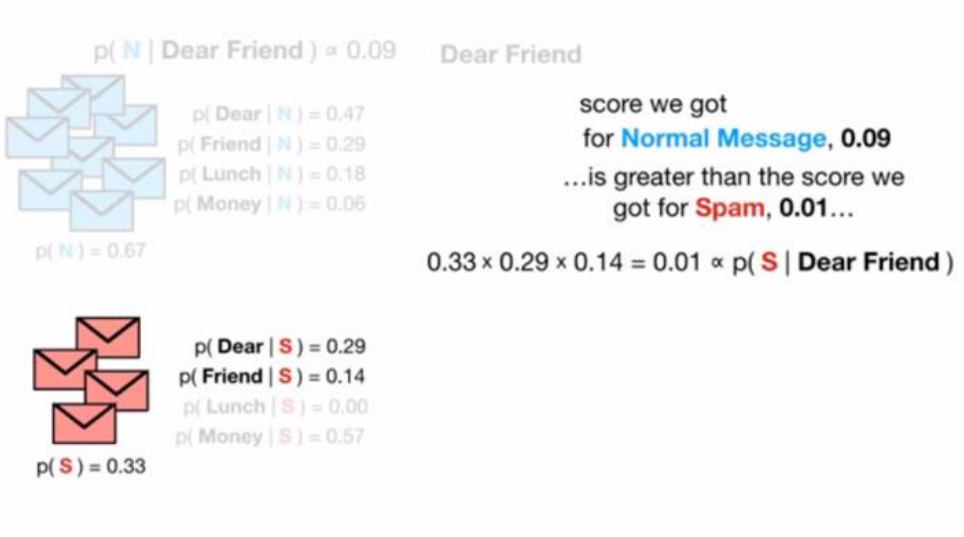
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

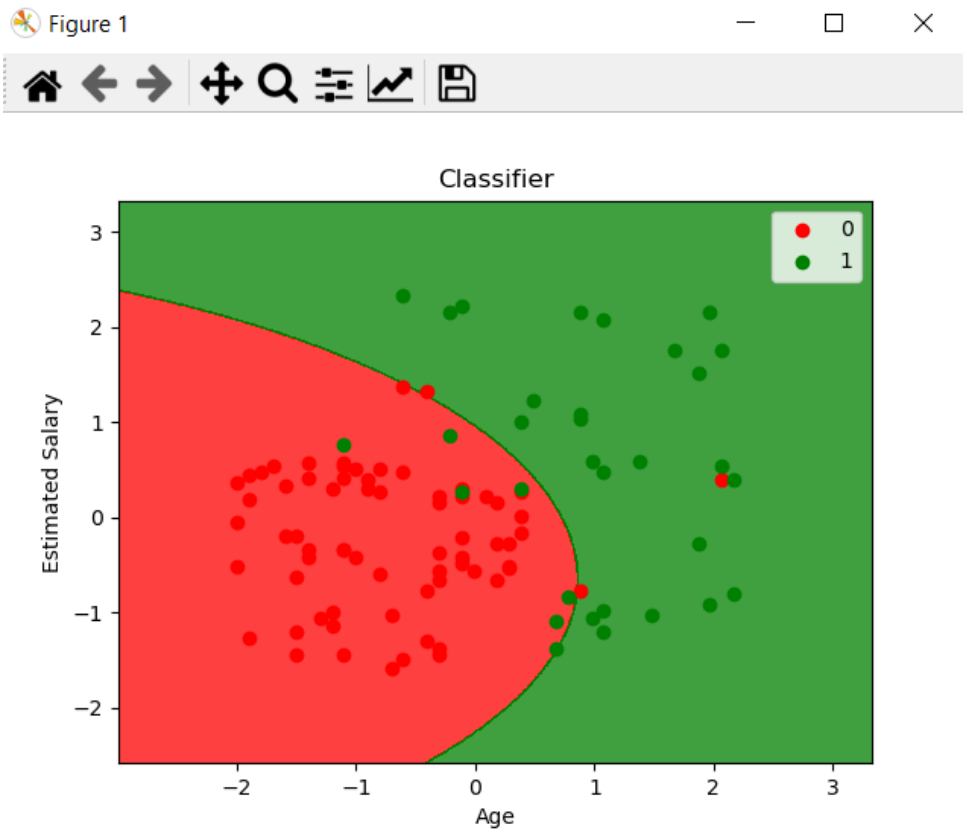
0.33 × 0.29 × 0.14 = 0.01 ∝ p(S | **Dear Friend**)

p(**Dear** | S) = 0.29
p(**Friend** | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(S) = 0.33



Visualization:



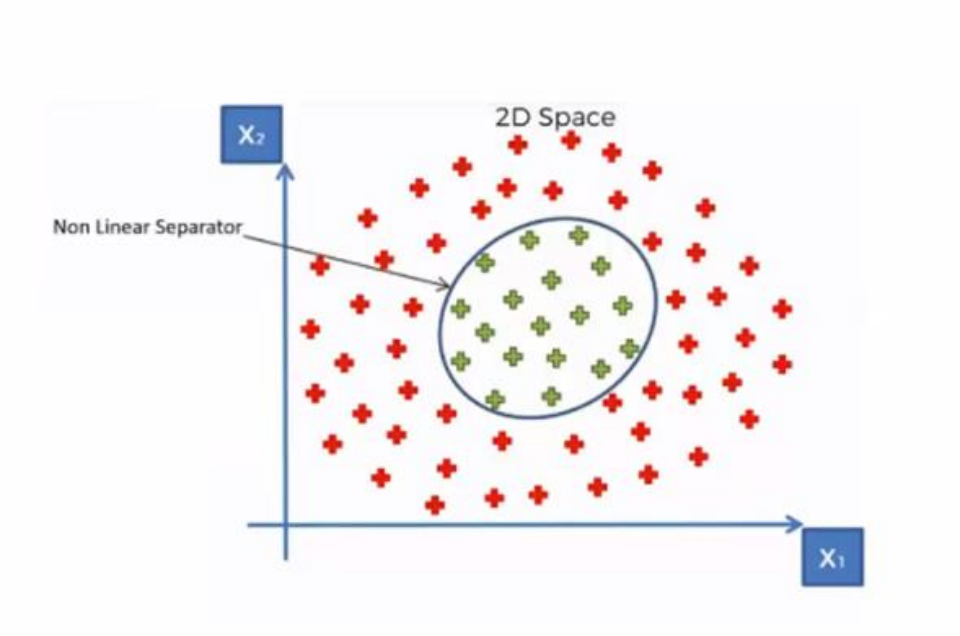
Accuracy:

Name	Type	Size	Value
acc	float64	1	0.9
classifier	naive_bayes.GaussianNB	1	GaussianNB object c

Classification	Value to Predict is Discrete		
	Logistic Regression		
	Support Vector Machine	93%	
	DecisionTree Classifier	94%	max_depth=2
	RandomForest Classifier	94%	max_depth=2 ,Trees=10
	Naive Bayes	90%	

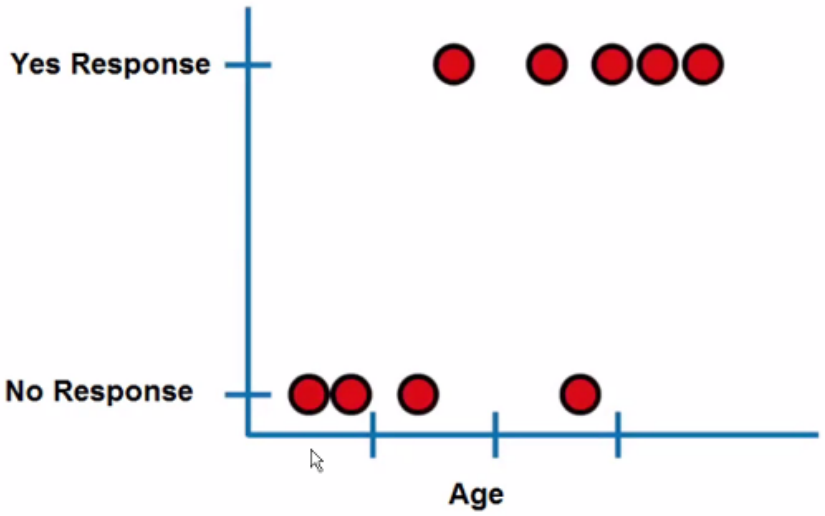
Logistic Regression:

How the formula is going to give to results to classify the problem

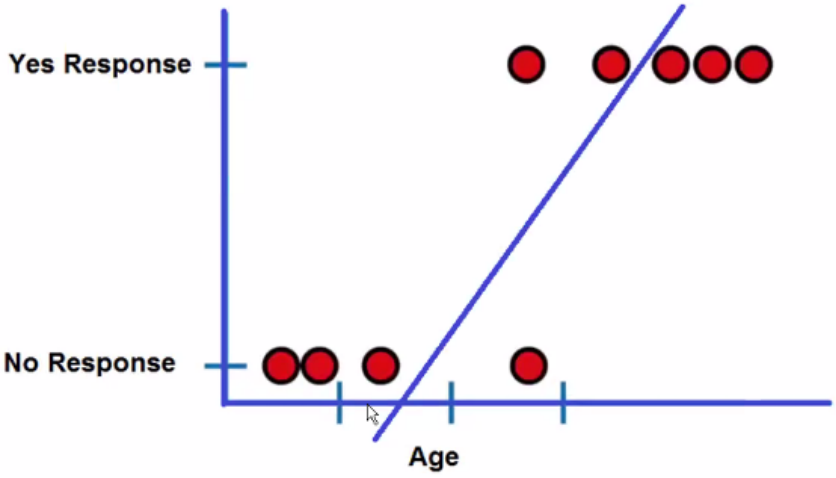


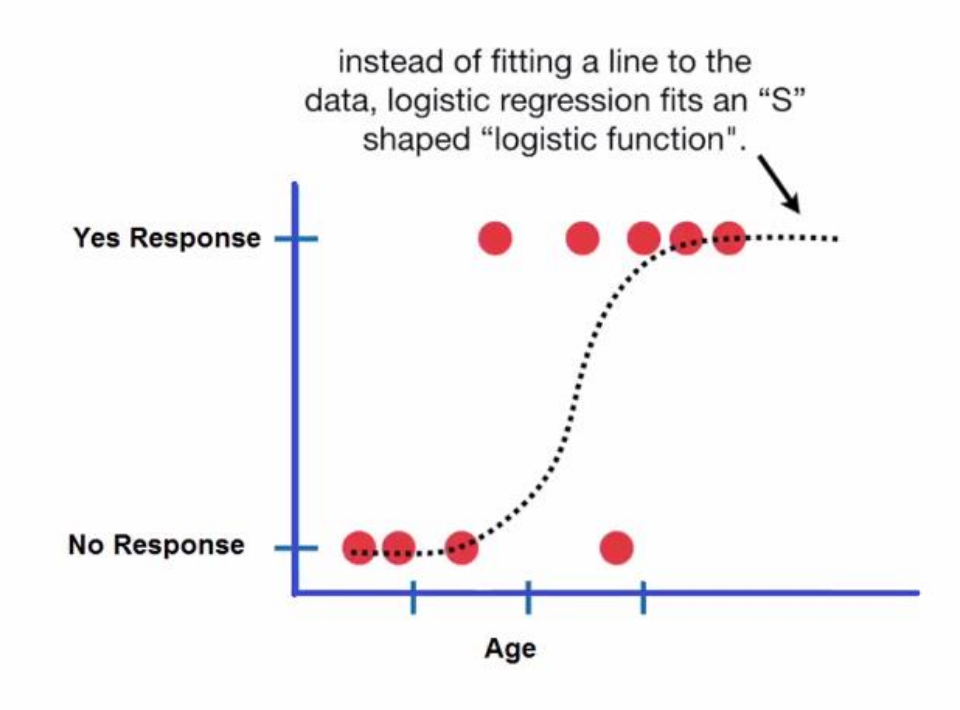
How the straight line fit => Regression

Logistic Regression

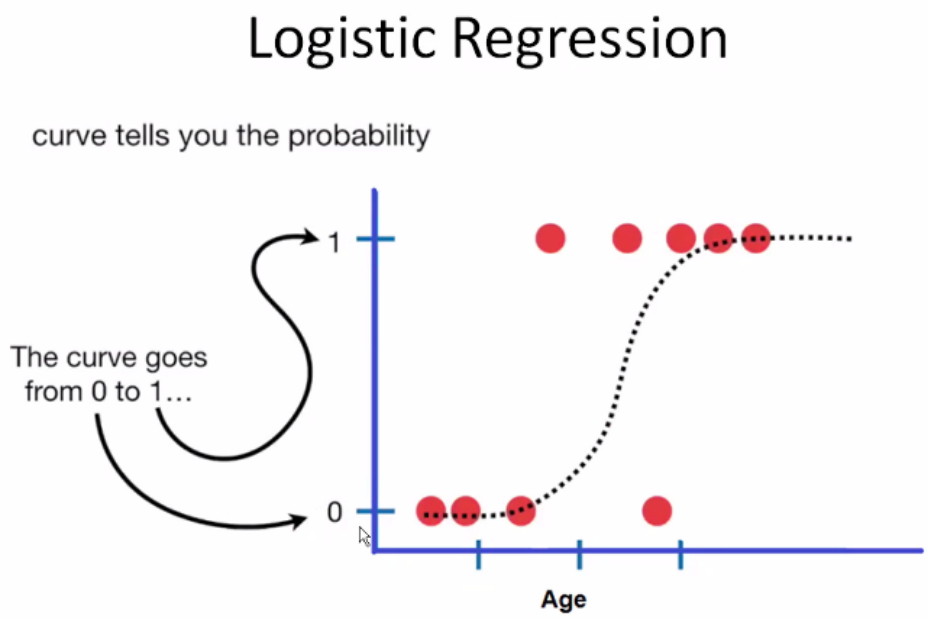


Logistic Regression

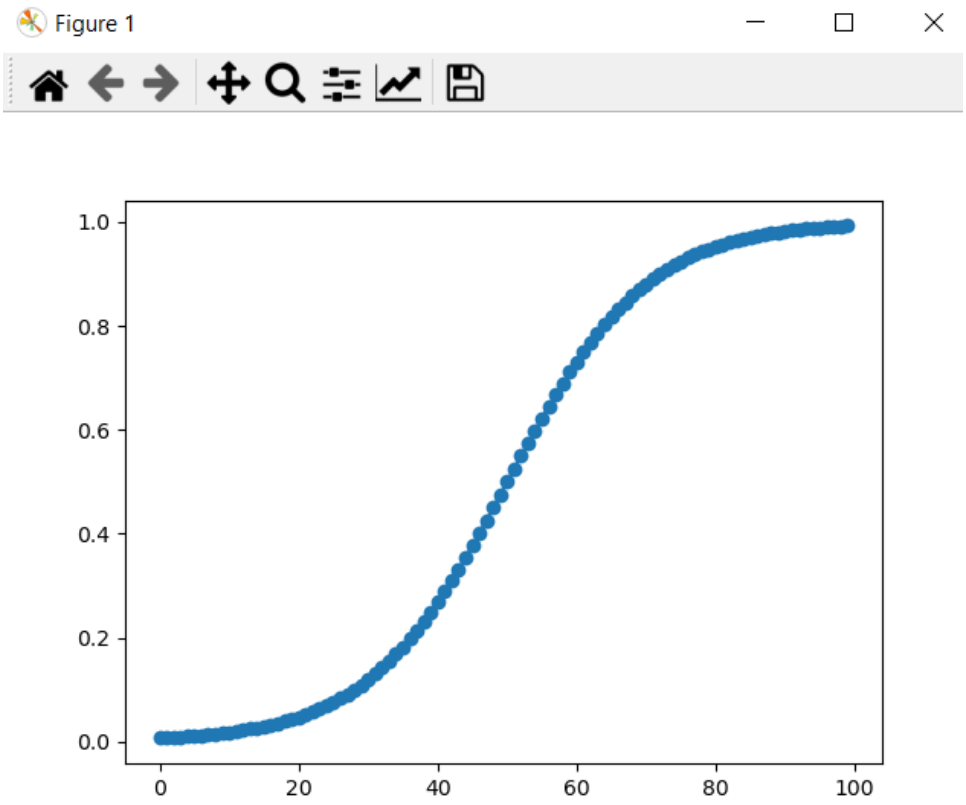




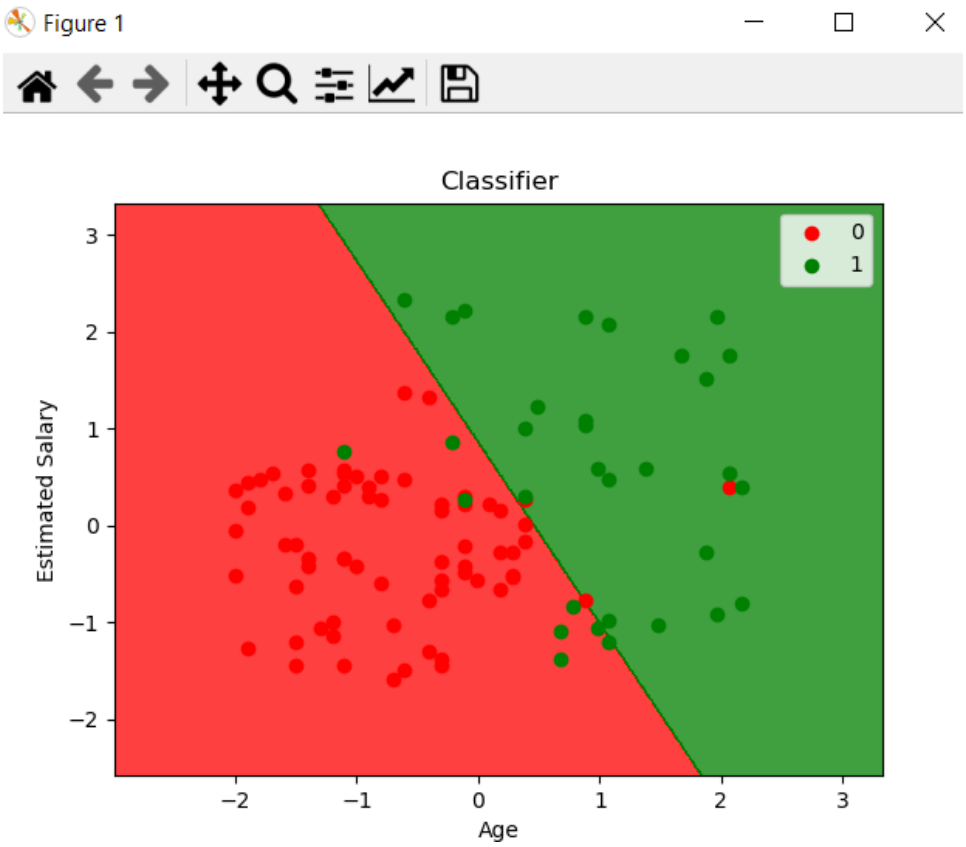
Bring Logistic function



Logistic function is also called as sigmoid functions



Visualization:

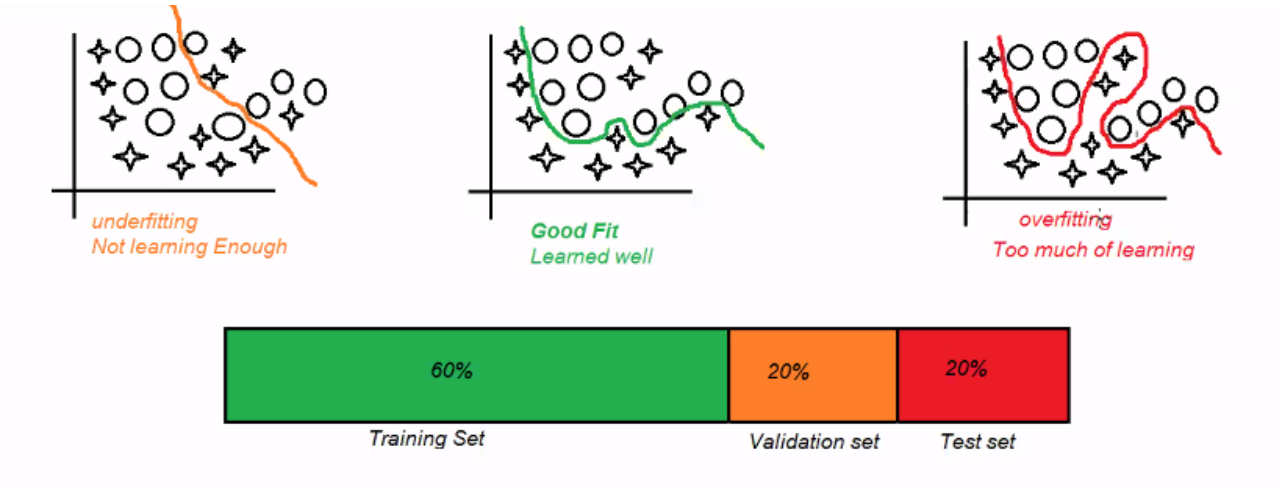


Accuracy:

Name	Type	Size	Value
acc	float64	1	0.89
classifier	linear_model._logistic.LogisticRegression	1	LogisticRegress…

Classification Value to Predict is Discrete

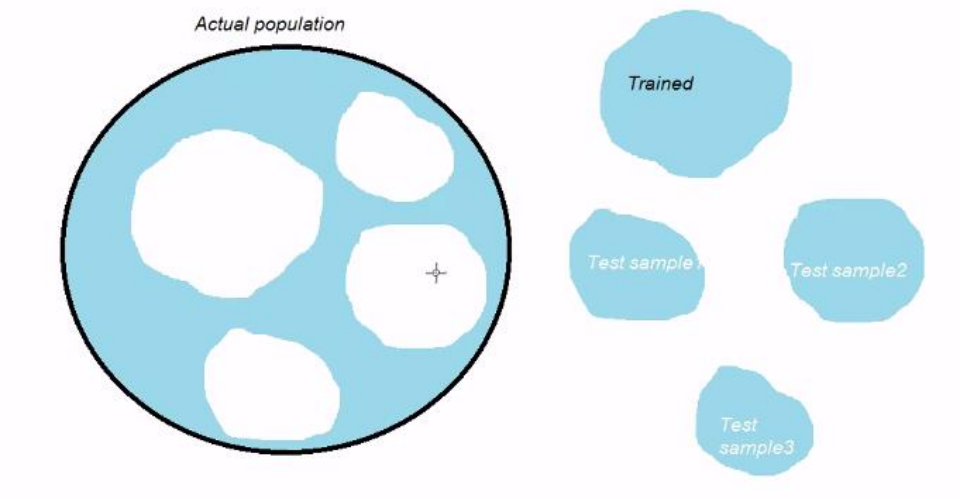
Logistic Regression	89%	
Support Vector Machine	93%	
DecisionTree Classifier	94%	max_depth=2
RandomForest Classifier	94%	max_depth=2 ,Trees=10
Naive Bayes	90%	



Training Set	*we train the model on training data*
Validation Set	*After training the model, we check how well it performs on the validation set*
Test Set	*When we have final Model (model that has performed well on both training and validation). We evaluate it on test set to go and get unbiaded estimation of its performance*

Stability check:

Test the model with various test dataset



K-Fold Cross Validation:

X_train, y_train – divide it into K-Folds

K = 5 to 10 folds

100 rows – dataset

Fold1	20 rows
Fold2	20 rows
Fold3	20 rows
Fold4	20 rows
Fold5	20 rows

Same as bootstrapping – to have equal folds.

Cross validation – 90%-10% Train – Test

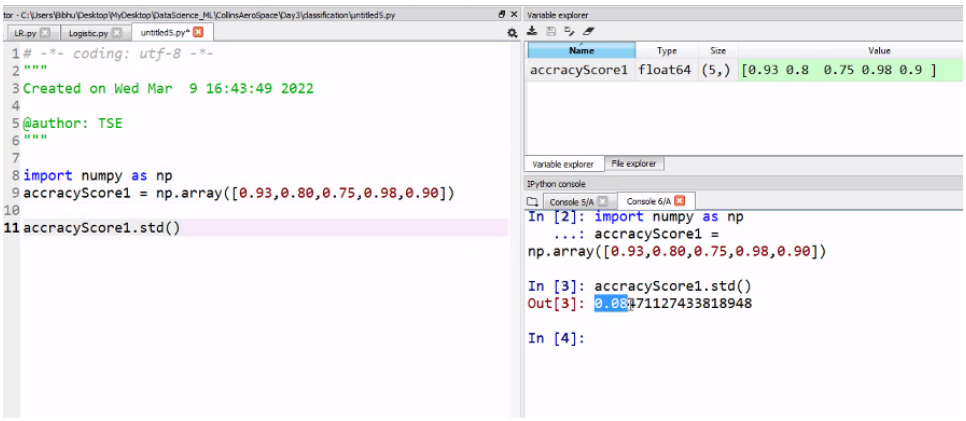
K-Fold Cross Validation
X_train,y_train divide it into K-Folds k=5
100 rows

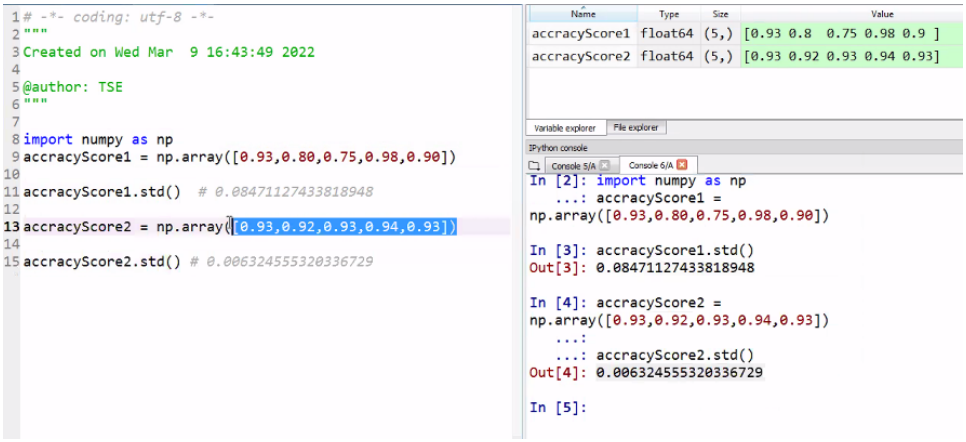
Fold1	20 rows
Fold2	20 rows
Fold3	20 rows
Fold4	20 rows
Fold5	20 rows

Cross Validation - 90%-10% Train-Test

Training	Testing	Accuracy
Fold1-2-3-4	Fold-5	93%
Fold-1-2-3-5	Fold-4	80%
Fold-1-2-4-5	Fold-3	75%
Fold-1-3-4-5	Fold-2	98%
Fold-2-3-4-5	Fold-1	90%

High variance
Accuracy variation > 1% Overfiiting issues
Accuracy variation == 1% Good Fit





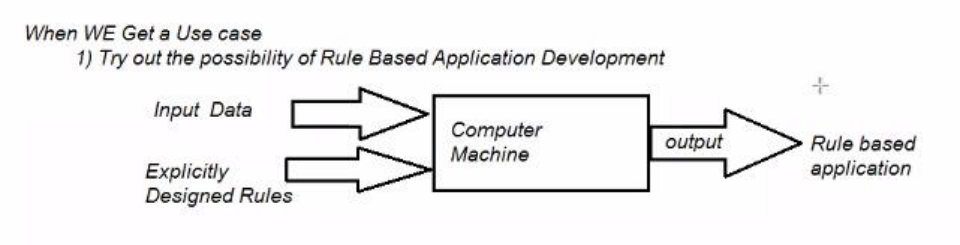
Data Science Training – Day 4

Topic: Neural Network / Deep Learning

- ANN
- CNN
- RNN

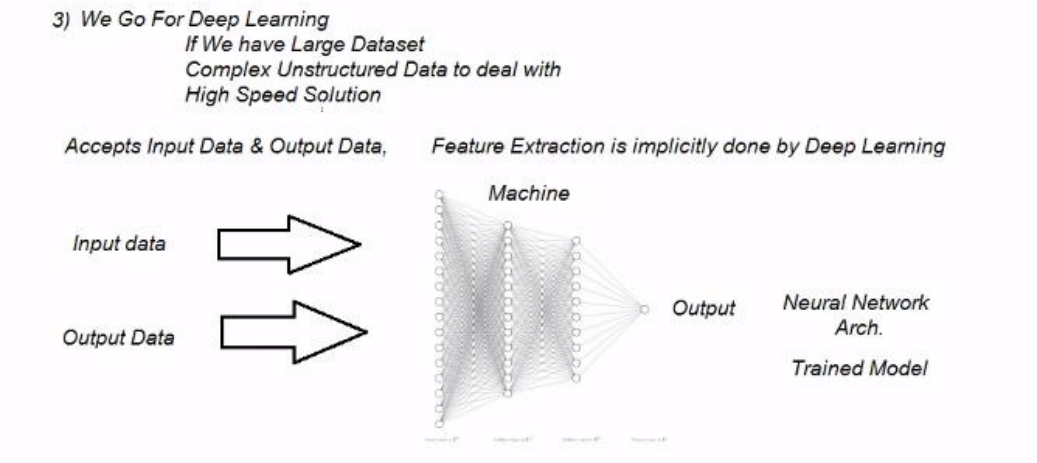
Use case:

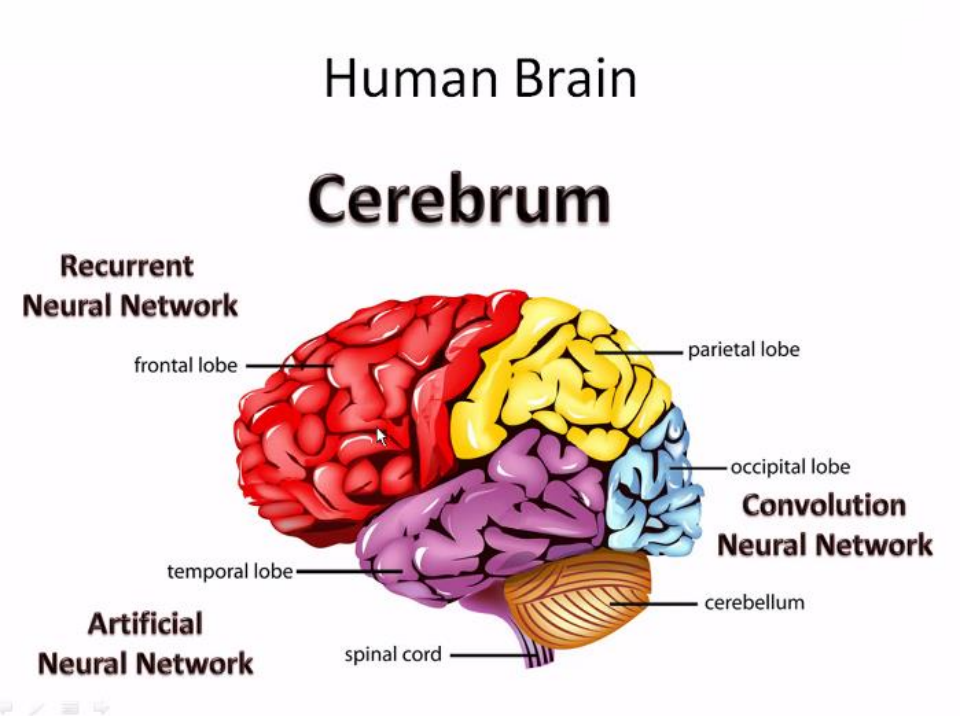
Explicit coding the logic



Develop the solution with explicit defined the rule-based application

Independent variable	Dependent variable
Input Data	Output Data
YrsExp	Salary
Investment, Tax, Revenue	Profit
Age, Salary, City	Response





CNN - human interpretation by seeing pictures.

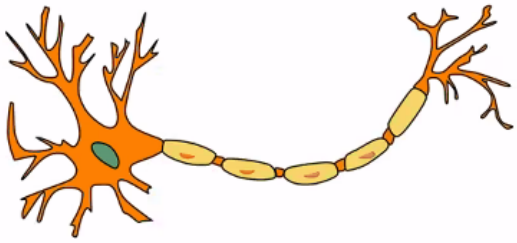
ANN -

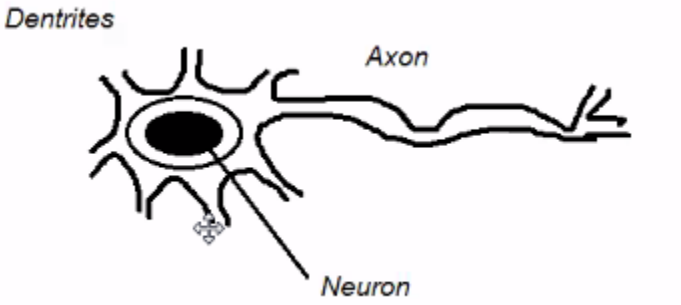
RNN – What did you have for breakfast? As time goes, you start forgetting it. On Thursday, what did you had for breakfast?

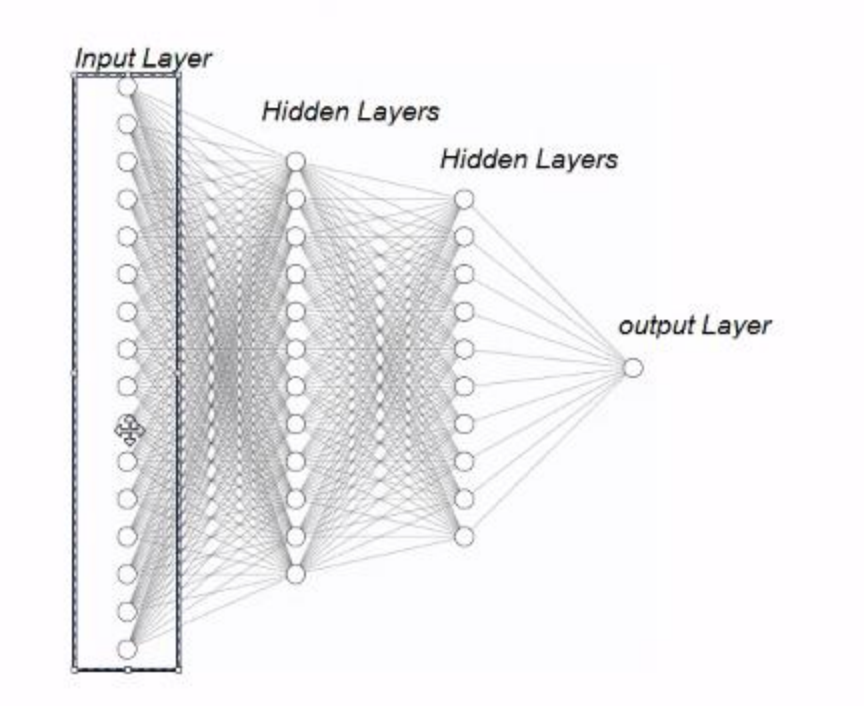
Process data in sequence, as time goes in sequence, your brain starts forgetting things.

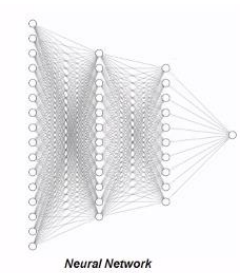
Deep Learning

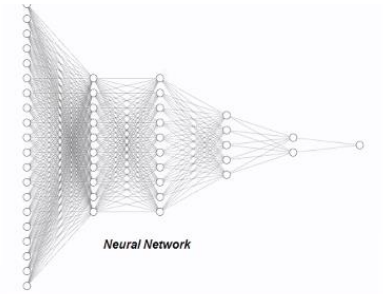
- In deep learning, these layered representations are learned via models called "neural networks", structured in literal layers stacked one after the other.
- The term "neural network" is a reference to neurobiology.

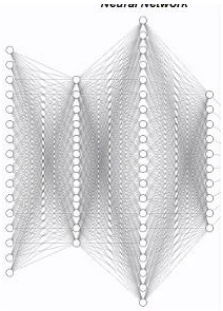


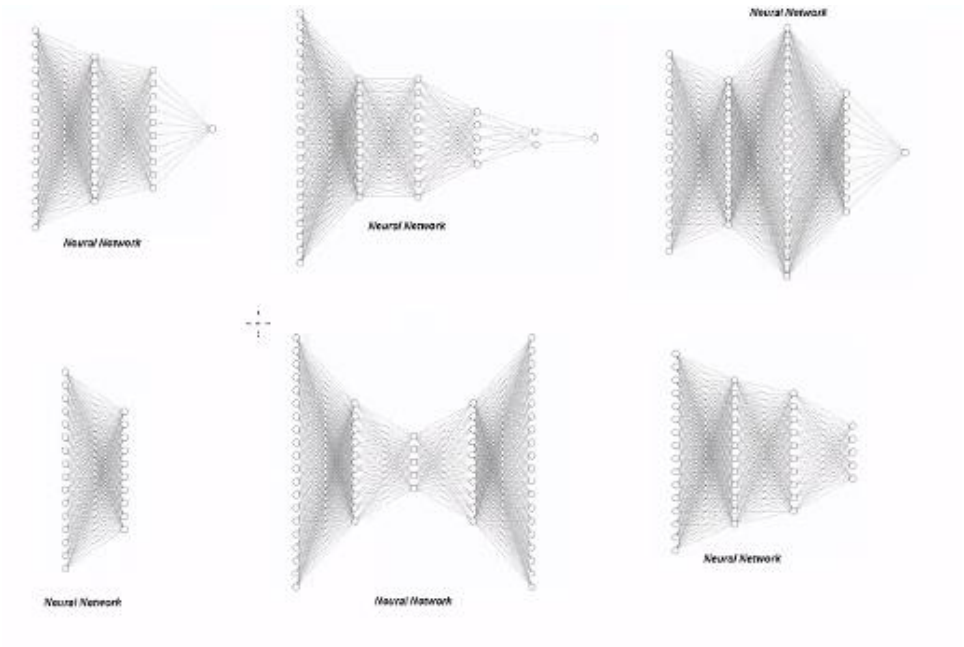


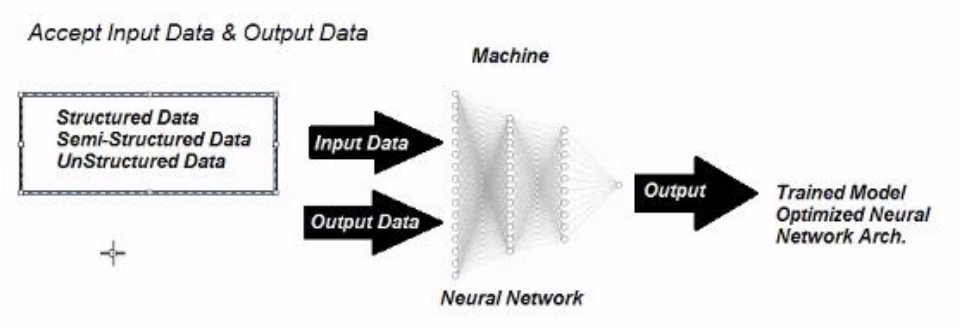

Input Layer
Hidden Layers
Hidden Layers
output Layer


Neural Network


Neural Network







Layered structured

Represented as neuron

Input Layer

Passing Input Data to NN
Each Neuron in Input Layer
represents X feature /column

Hidden Layer

Capture Patterns in the data
All calculation that needs to be done on input to transform it into desired output is done here

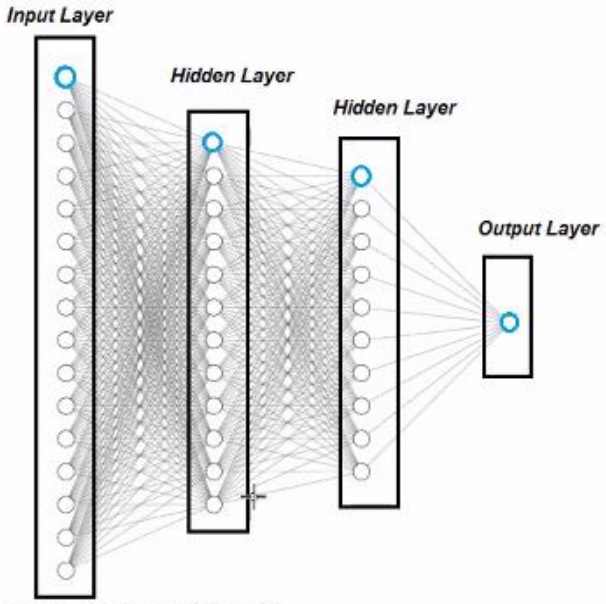
Each neuron in HL captures a pattern

NumOfNeurons in HL == Trial & Error

3 neurons = 3 patterns
30 neurons = 30 patterns
300 neurons = 300 patterns

Output Layer

Return output



.....

min
1 2 3 4 5 6 .....

12 24 48 .....

underfitting
Not learning enough

Error 27k 10k 5k 1k 500

Training error High
Testing error High

Overfitting issues

Training error Low
Testing error High

.....

min
1 2 3 4 5 6 .....

12 24 48 .....

underfitting
Not learning enough

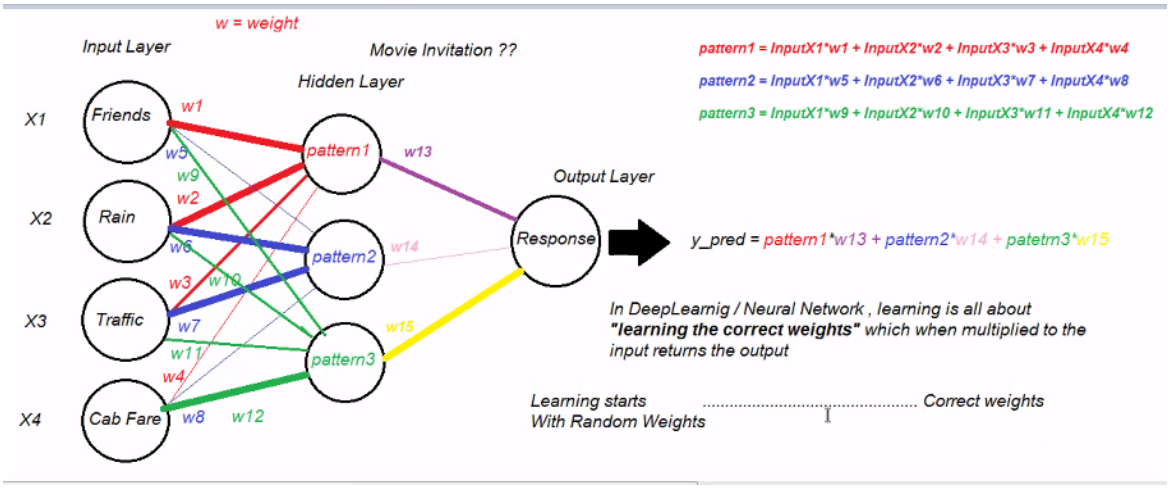
Error 27k 10k 5k 1k 500

Training error High
Testing error High

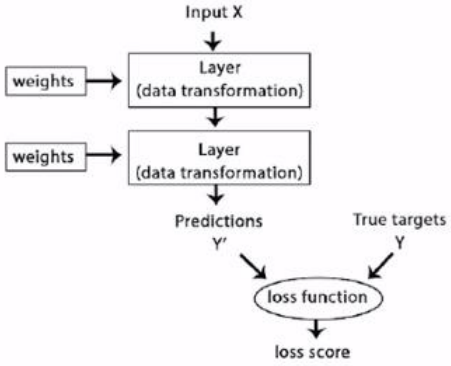
Overfitting issues

Training error Low
Testing error High

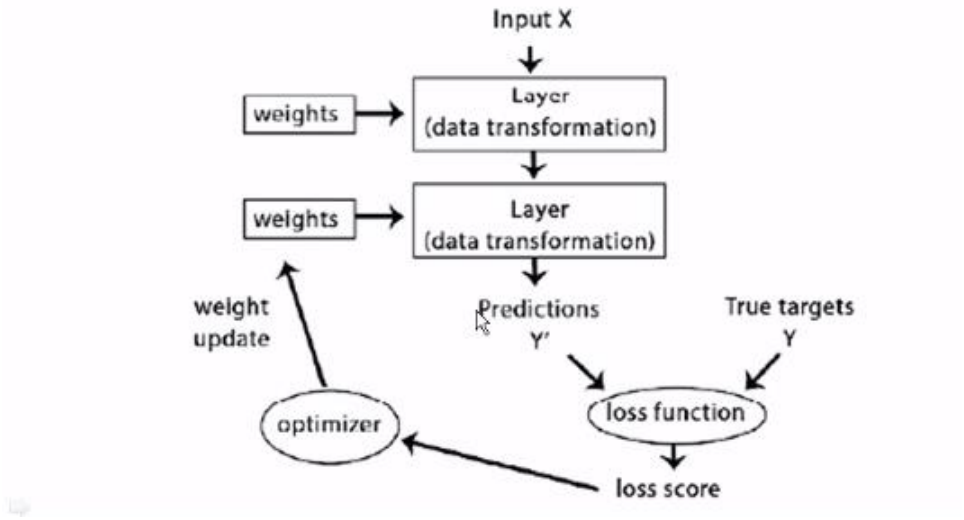
Regression	**Continuous value Single Pred (1 neuron)**
Binary Classification	**Discrete Binary value 0/1 Pred (1 neuron)**
Multi-Class Classification	**Discrete Value (more than 1 neuron)**

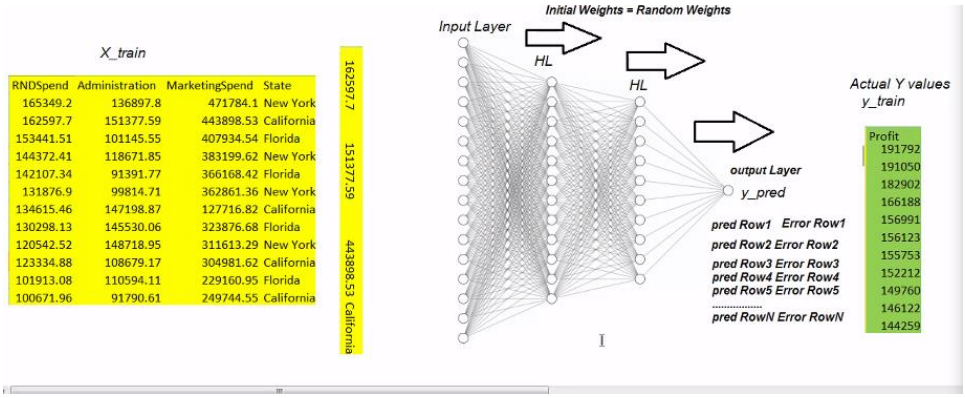


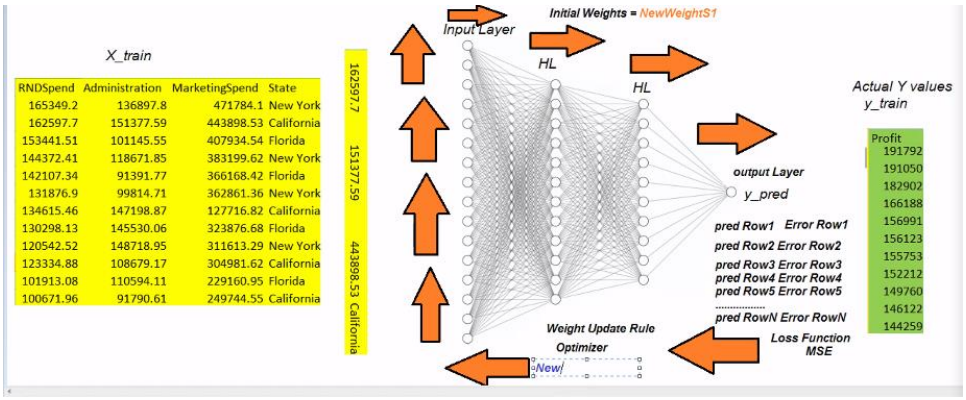
- To control the output of a neural network, you need to be able to measure how far this output is from what you expected.
- This is the job of the **"loss function"** of the network, also called **"objective function".**
- The loss function takes the predictions of the network and the true target , and computes a distance score, capturing how well the network has done on this specific example.

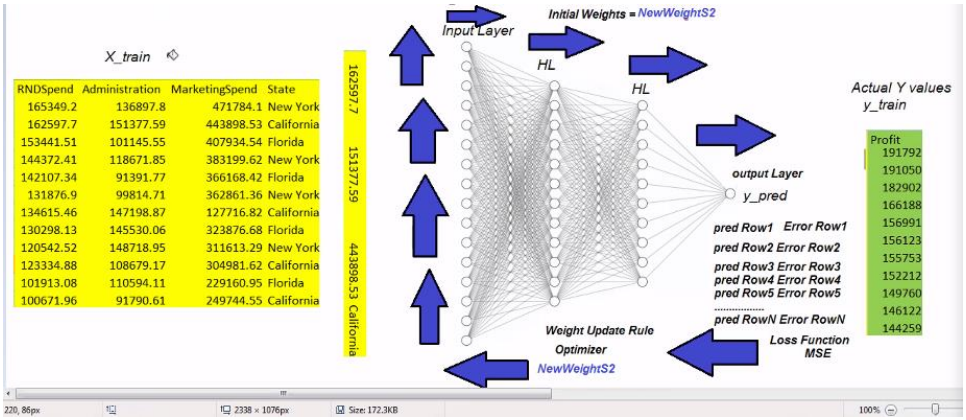


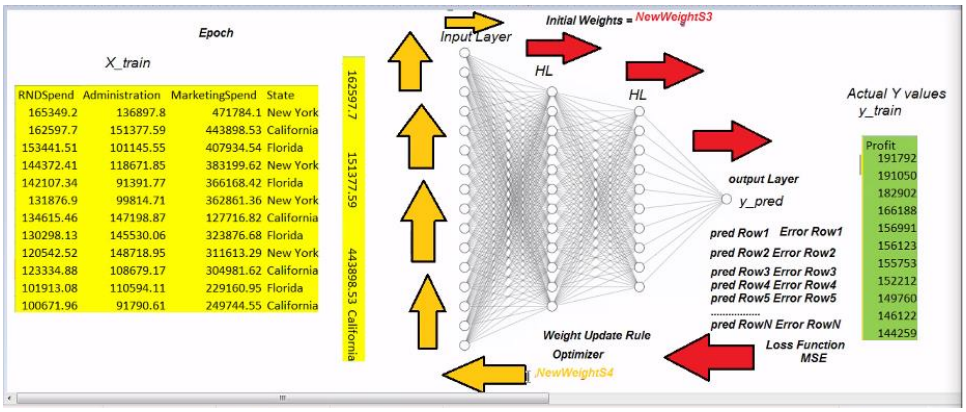
- The fundamental trick in deep learning is to use the loss score as a feedback signal to adjust the value of the weights by a little bit, in a direction that would lower the loss score for the current example.
- This adjustment is the job of the "optimizer", which implements what is called the **"backpropagation"** algorithm, the central algorithm in deep learning.











UnderFitting	Not Learning Enough
Good Fit	Learned well to generalize
OverFitting	Too much of Learning

TrainingAcc	TestingAcc	
60%	60%	Low Acc Underfitting
90%	90%	Good Fit
90%	95%	Good Fit
90%	75%	OverFitting

```
numOfEpochs = To be decided by developer (Trial & Error)

Step1: Initial Weights = Random Weight Allocation (All rows in training set is used against the initial weight)
Epoch1    y_pred = InputX1*weight1 + InputX2*weight2 + InputX3*weight3    .............+InputXn*weightn
          Error = lossFunction(y_pred,y_actual)
          WeightUpdate = optimizer ( weight update Rule)
                         NewWeightS1

Step2: Initial Weights = NewWeightS1
Epoch2    y_pred = InputX1*weight1 + InputX2*weight2 + InputX3*weight3    .............+InputXn*weightn
          Error = lossFunction(y_pred,y_actual)
          WeightUpdate = optimizer ( weight update Rule)
                         NewWeightS2

Step3: Initial Weights = NewWeightS2
Epoch3    y_pred = InputX1*weight1 + InputX2*weight2 + InputX3*weight3    .............+InputXn*weightn
          Error = lossFunction(y_pred,y_actual)
          WeightUpdate = optimizer ( weight update Rule)
                         NewWeightS3
.................
StepN: Initial Weights = NewWeightSN-1
EpochN    y_pred = InputX1*weight1 + InputX2*weight2 + InputX3*weight3    .............+InputXn*weightn
          Error = lossFunction(y_pred,y_actual)
          WeightUpdate = optimizer ( weight update Rule)
                         NewWeightSN   => Final Weights giving max accuracy and lowest error / loss
```

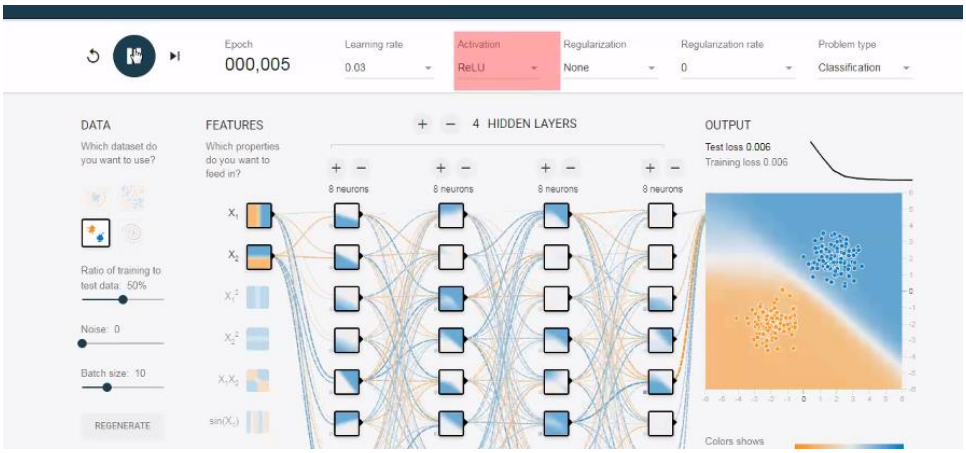
```
Formula at the heart of Each Neuron [Linear Spread of Data]

y_pred = ActivationFunction(bias + InputX1*weight1 + InputX2*weight2 + InputX3*weight3.....+InputXn*weightn)


Activation Function

Linear Activation Functions                    (Linear Spread of data)

Non-Linear Activation Functions                (Linear Spread of data & Non-Linear Spread of data)
          Sigmoid , Tanh, ReLu,LeakyReLu,ExpReLu


Linear Regression - Linear Spread of Data

y_pred = b0 +  InputX1*b1 + InputX2*b2 + InputX3*b3    .............+InputXn*bn
```

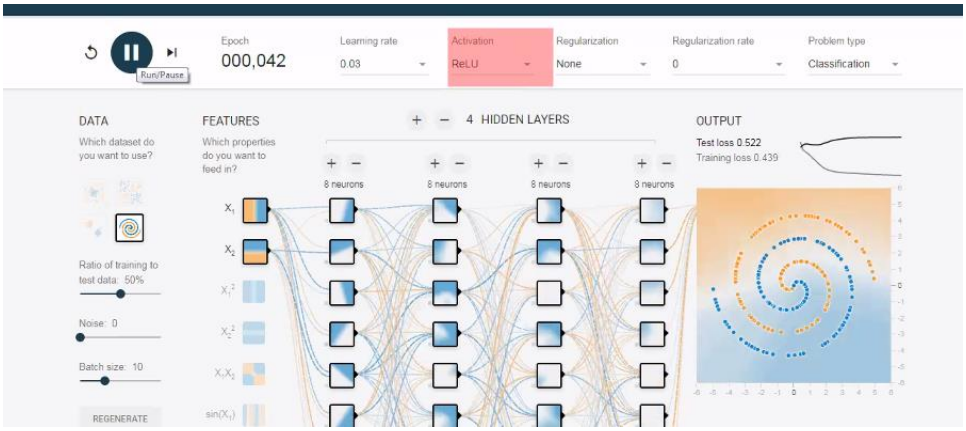
Bias = intercept

Activation function

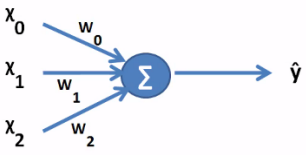
Tensor flow playground - To see how neural network work

To see how activation function work





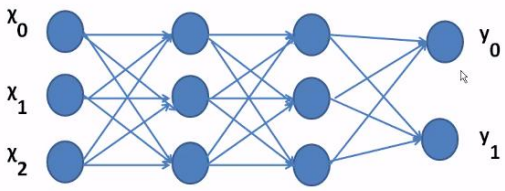
Neural Network with Single Neuron



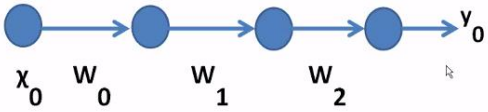
$$\hat{y} = x_0 w_0 + x_1 w_1 + x_2 w_2 = \sum_i x_i w_i$$

$$\begin{pmatrix} w_0 & w_1 & w_2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

Muti-layer Perceptron

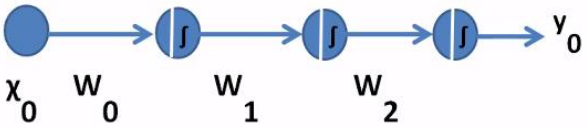


Linear Function



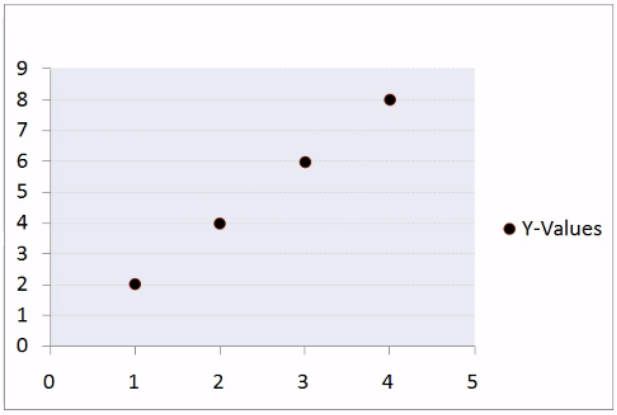
$$y_0 = x_0 w_0 w_1 w_2 = x_0 w_C$$

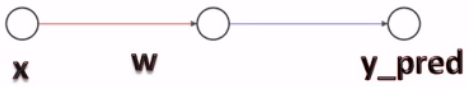
Non-Linear Function



$$y_0 = \sigma (\sigma (\sigma (x_0 w_0) w_1) w_2)$$

Simple Example





y_pred = xw

E = $(\hat{y}-y)^2 = (xw-y)^2$ *sum of squared Error*

$\frac{\partial \mathbf{E}}{\partial \mathbf{w}} = 2x(xw-y)$ Derivative

$\mathbf{w - \alpha \frac{\partial E}{\partial w} = w - \alpha\, 2x(xw-y)}$ Update Rule

- Random weight initialization w=0.5
- learning rating α = 0.1

- $w_{new} = w - 0.1 * 2x(xw-y)$
- (x,y)
- (2,4) w=0.5 ← 0.5 - 0.1*2*2(2*0.5-4) = 1.7
- (1,2) w =1.7 ← 1.7-0.1*2*1(1*1.7-2) = 1.76
- (3,6) w= 1.76 ← 1.76-0.1*2*3(3*1.76-6)=2.192
- w ~ 2

Weight update Rule:

SGD:

Epoch1	x	y	LR	weights	NewWeights	y_pred	y_test	Error / Loss	
	2	4	0.1	0.5	1.7	3.4	4	-0.6	individual Row Error
	1	2	0.1	1.7	1.76	1.76	2	-0.24	individual Row Error
	3	6	0.1	1.76	2.192	6.576	6	0.576	individual Row Error
	4	8	0.1	2.192	1.5776	6.3104	8	-1.6896	individual Row Error
								-0.4884	

Epoch2	x	y	LR	weights	NewWeights	y_pred	y_test	Error / Loss	
	2	4	0.1	1.5776	1.91552	3.83104	4	-0.16896	individual Row Error
	1	2	0.1	1.91552	1.932416	1.932416	2	-0.067584	individual Row Error
	3	6	0.1	1.932416	2.0540672	6.162202	6	0.1622016	individual Row Error
	4	8	0.1	2.054067	1.88105216	7.524209	8	-0.47579136	individual Row Error
								-0.13753344	Epoch Error

Epoch3	x	y	LR	weights	NewWeights	y_pred	y_test	Error / Loss	
	2	4	0.1	1.881052	1.976210432	3.952421	4	-0.047579136	individual Row Error
	1	2	0.1	1.97621	1.980968346	1.980968	2	-0.019031654	individual Row Error
	3	6	0.1	1.980968	2.015225324	6.045676	6	0.045675971	individual Row Error
	4	8	0.1	2.015225	1.966504288	7.866017	8	-0.133982847	individual Row Error
								-0.038729417	Epoch Error

Epoch4	x	y	LR	weights	NewWeights	y_pred	y_test	Error / Loss	
	2	4	0.1	1.966504	1.993300858	3.986602	4	-0.013398285	individual Row Error
	1	2	0.1	1.993301	1.994640686	1.994641	2	-0.005359314	individual Row Error
	3	6	0.1	1.994641	2.004287451	6.012862	6	0.012862353	individual Row Error
	4	8	0.1	2.004287	1.990567608	7.96227	8	-0.03772957	individual Row Error
								-0.010906204	Epoch Error

BGD:

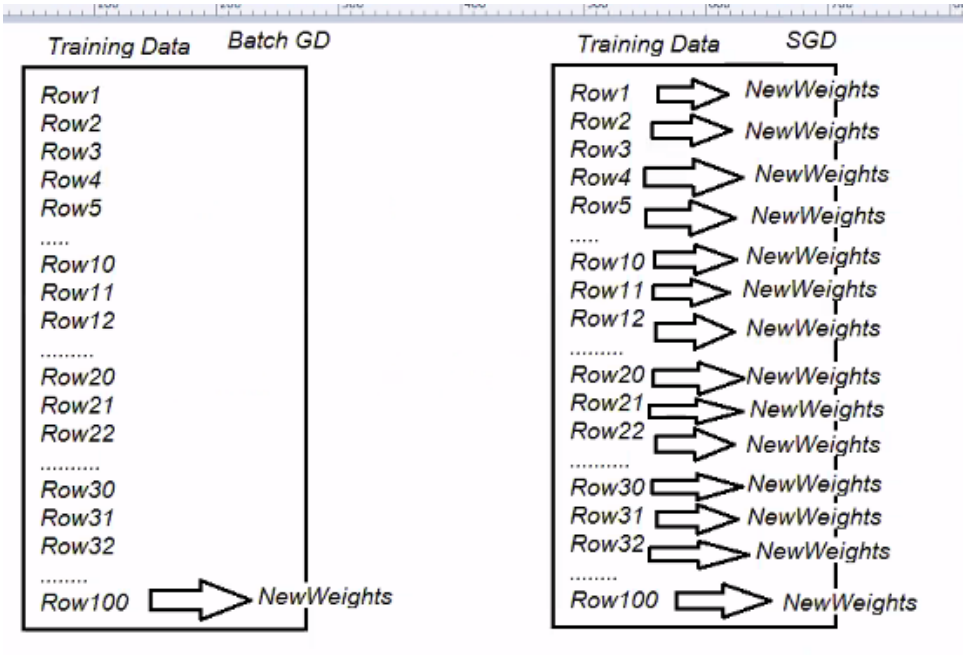
Epoch1	x	y	LR	weights	NewWeights	y_pred	y_test	Error / Loss
	2	4	0.1	0.5	1.7	3.4	4	-0.6
	1	2	0.1	0.5	0.8	0.8	2	-1.2
	3	6	0.1	0.5	3.2	9.6	6	3.6
	4	8	0.1	0.5	5.3	21.2	8	13.2
					2.75	NewWeights		3.75

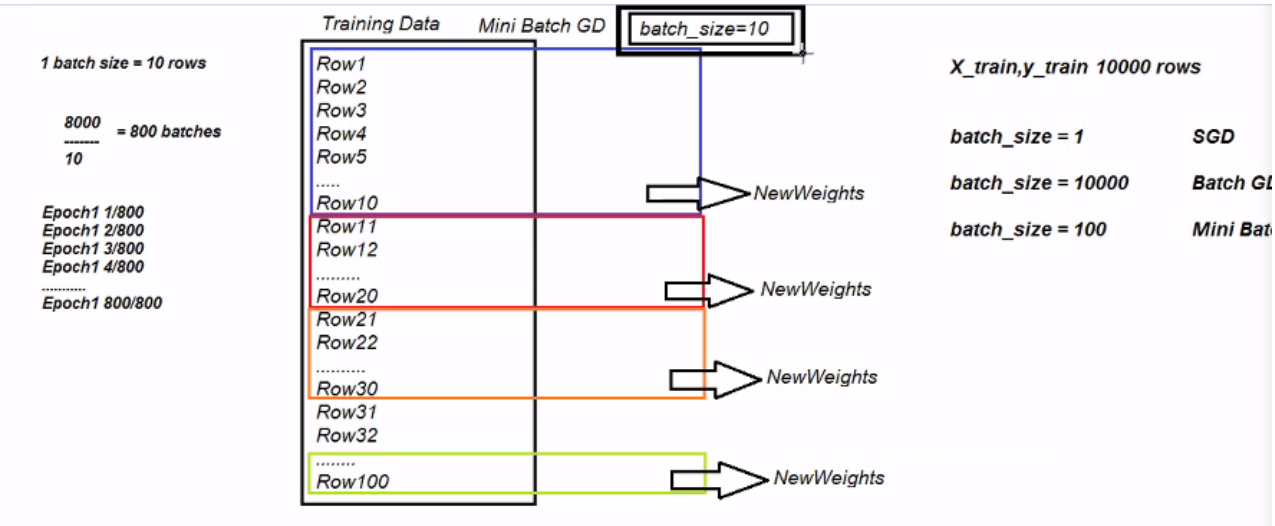
Epoch2	x	y	LR	weights	NewWeights	y_pred	y_test	Error	
	2	4	0.1	2.75	2.15	4.3	4	0.3	individual Row Error
	1	2	0.1	2.75	2.6	2.6	2	0.6	individual Row Error
	3	6	0.1	2.75	1.4	4.2	6	-1.8	individual Row Error
	4	8	0.1	2.75	0.35	1.4	8	-6.6	individual Row Error
					1.625			-1.875	Epoch Error

Epoch3	x	y	LR	weights	NewWeights	y_pred	y_test	Error	
	2	4	0.1	1.625	1.925	3.85	4	-0.15	individual Row Error
	1	2	0.1	1.625	1.7	1.7	2	-0.3	individual Row Error
	3	6	0.1	1.625	2.3	6.9	6	0.9	individual Row Error
	4	8	0.1	1.625	2.825	11.3	8	3.3	individual Row Error
					2.1875			0.9375	Epoch Error

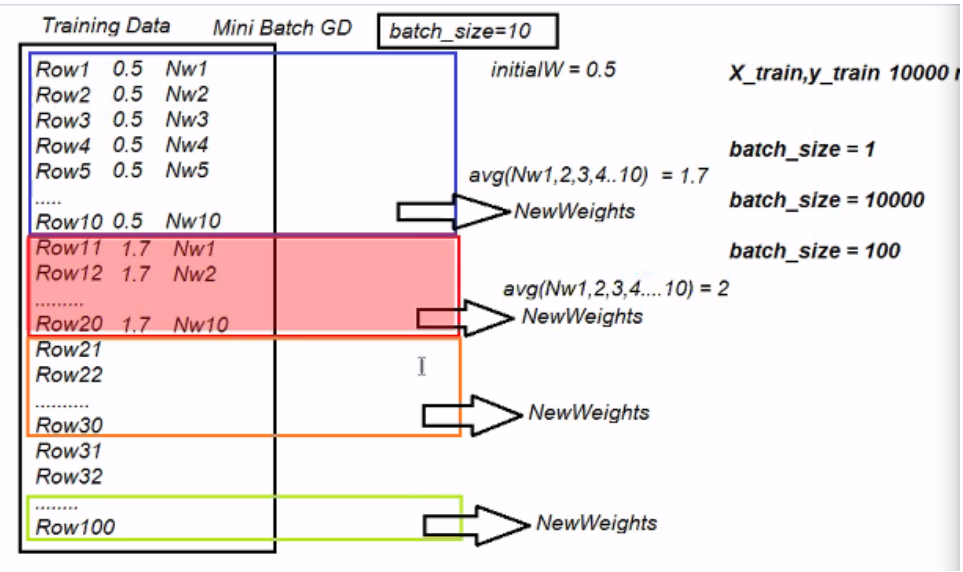
Epoch4	x	y	LR	weights	NewWeights	y_pred	y_test	Error
	2	4	0.1		1.6	3.2	4	-0.8
	1	2	0.1		0.4	0.4	2	-1.6
	3	6	0.1		3.6	10.8	6	4.8
	4	8	0.1		6.4	25.6	8	17.6
					3			5

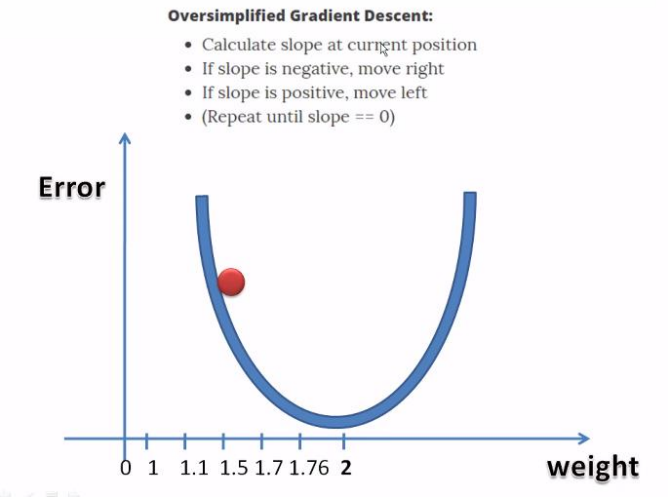
Epoch5	x	y	LR	weights	NewWeights	y_pred	y_test	Error	
	2	4	0.1	1.90625	1.98125	3.9625	4	-0.0375	individual Row Error
	1	2	0.1	1.90625	1.925	1.925	2	-0.075	individual Row Error
	3	6	0.1	1.90625	2.075	6.225	6	0.225	individual Row Error
	4	8	0.1	1.90625	2.20625	8.825	8	0.825	individual Row Error
					2.046875			0.234375	Epoch Error





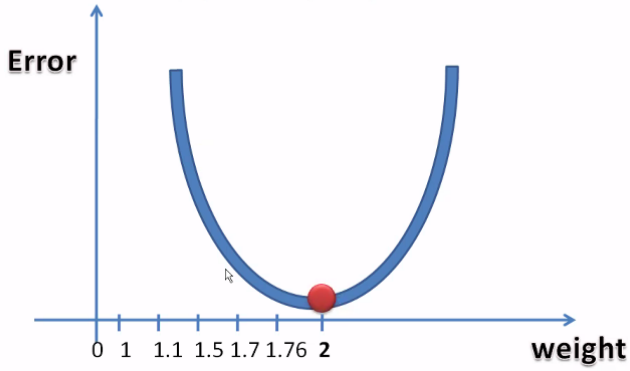
Mini batch is good approach





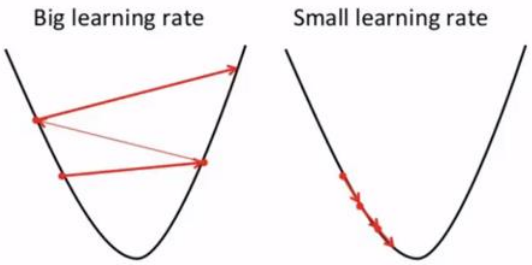
Oversimplified Gradient Descent:

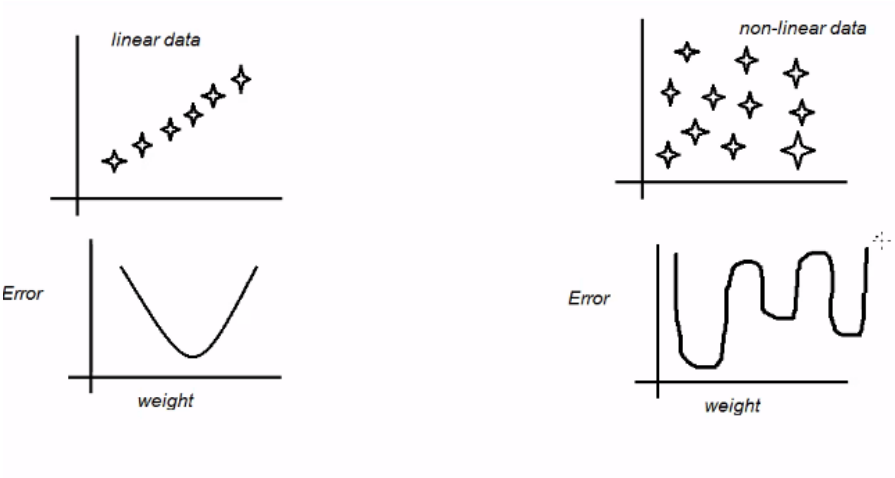
- Calculate slope at current position
- If slope is negative, move right
- If slope is positive, move left
- (Repeat until slope == 0)

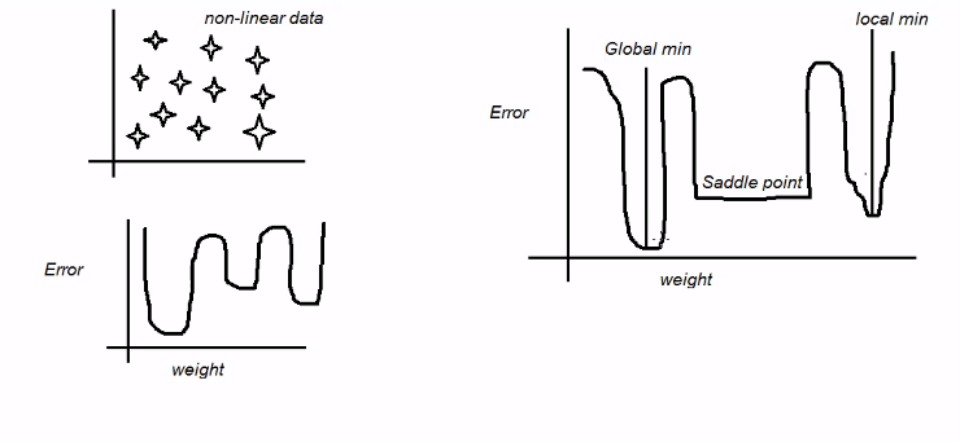


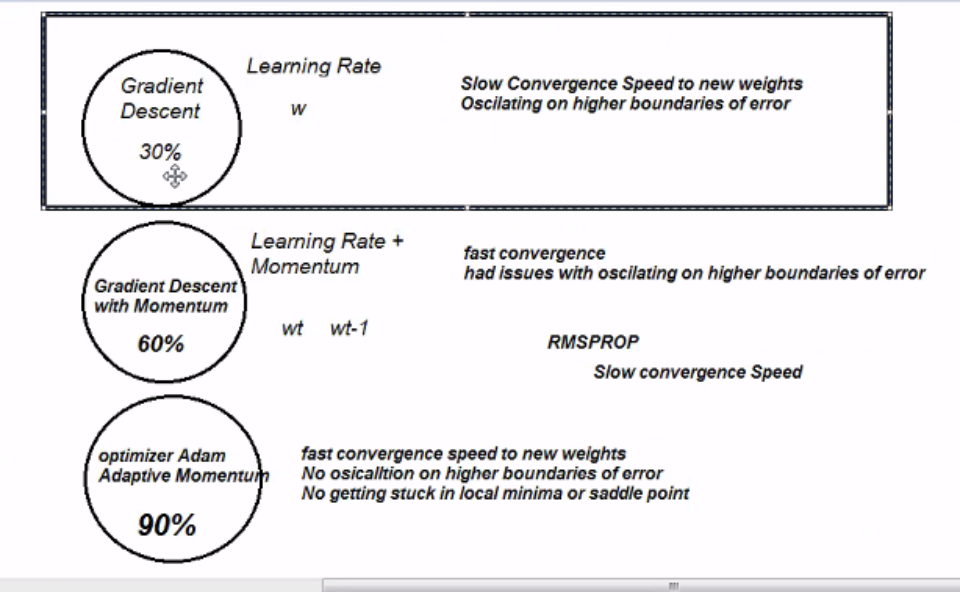
Learning rate

1. **if it is too small, then the model will take some time to learn.**
2. **if it is too large, model will converge as our pointer will shoot and we'll not be able to get to minima.**







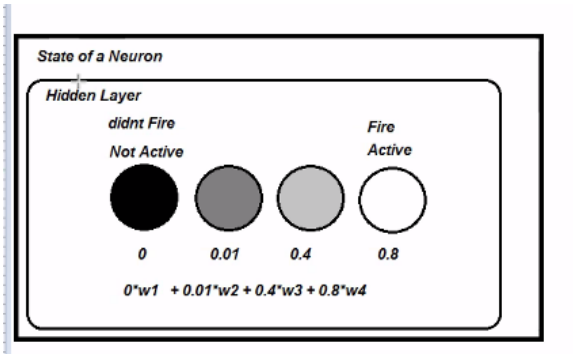


Adaptive Momentum

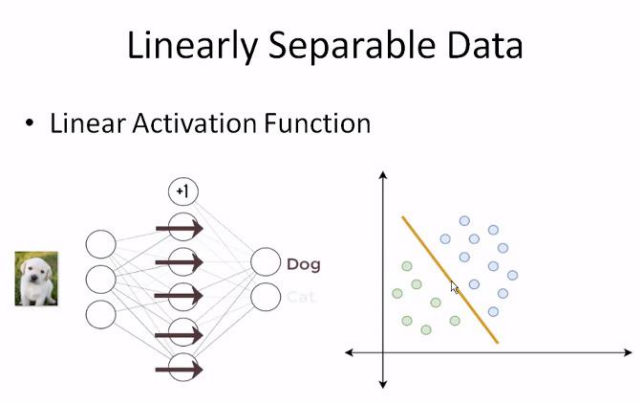
- Adam method is used to accelerate the gradient descent algorithm by taking into consideration the exponentially weighted average of the gradients.

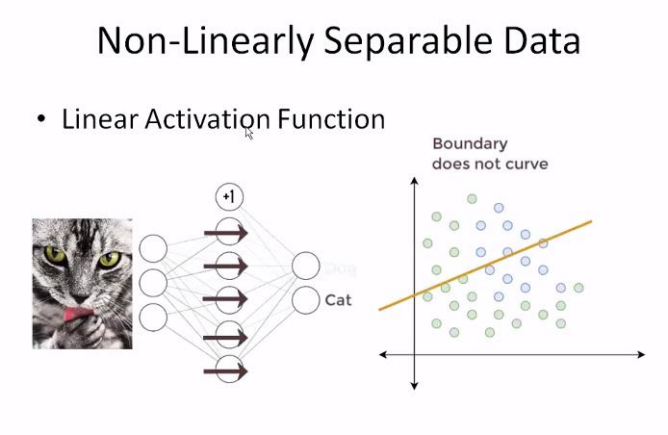
- **new weight ←(old weight)-(learning rate)(gradient)**
- **new weight ←(old weight)-(learning rate)(gradient) + past gradient**
- **(accumulator)←(old accumulator)(momentum)+gradient**
- **momentum → weighted average of past gradients**
- **new weight ←(old weight)-(learning rate)(accumulator)**

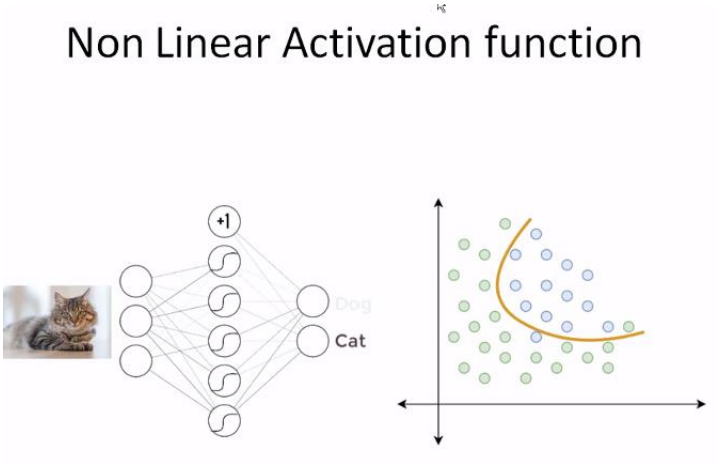
Not fire means not contirbute

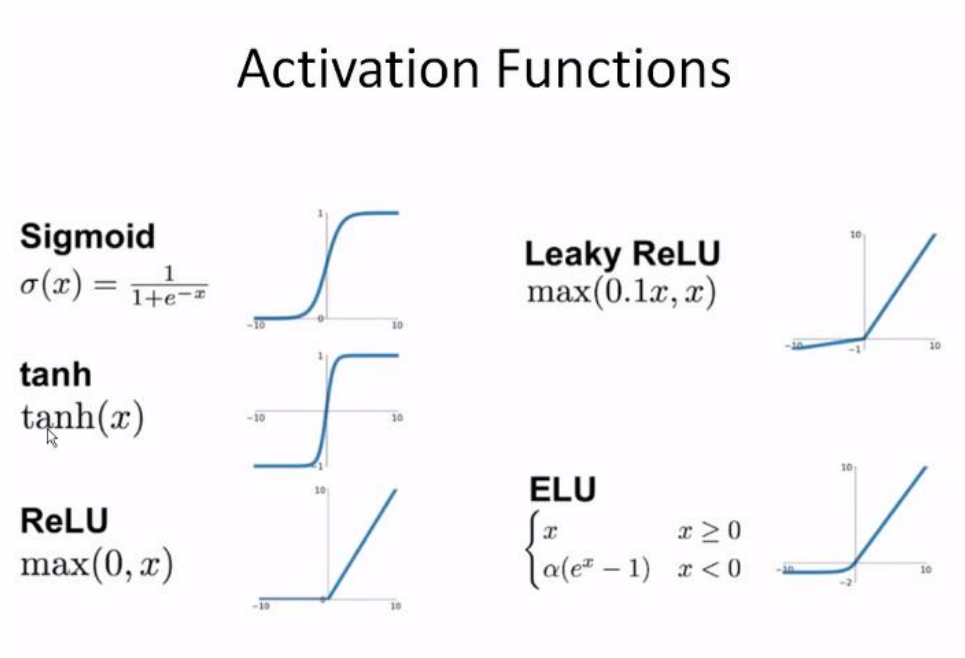


Activation Functions:









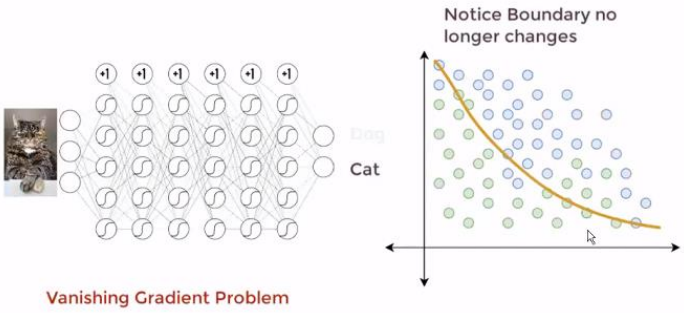
Sigmoid – squeezing in nature – Range – 0 to 1 - Put in problem in case of complex situation

Tanh –Range: -1 to 1 - squeezing in nature - Put in problem in case of complex situation

Vanishing gradient problem due to squeezing nature

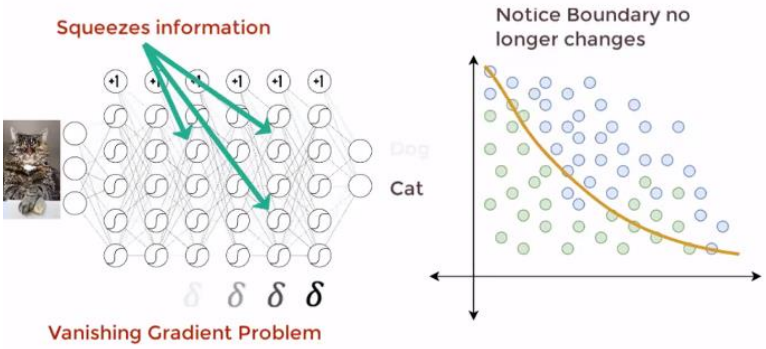
ReLU - (0, x)

Even if the model goes through many examples it doesnt learn much



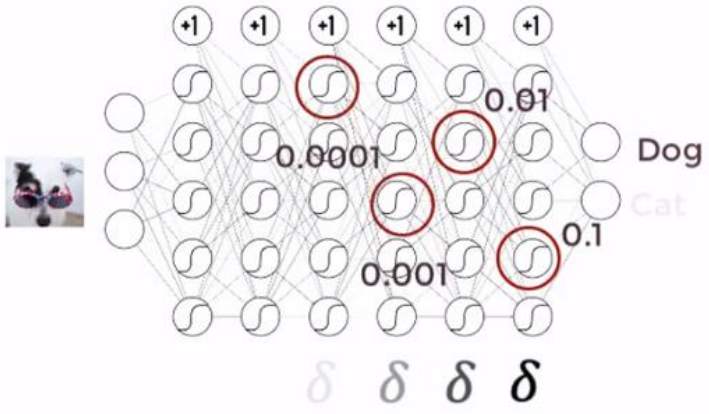
Vanishing Gradient Problem

During backpropogation the gradients become smaller and smaller and smaller until they eventually vanish no gradients means no learning



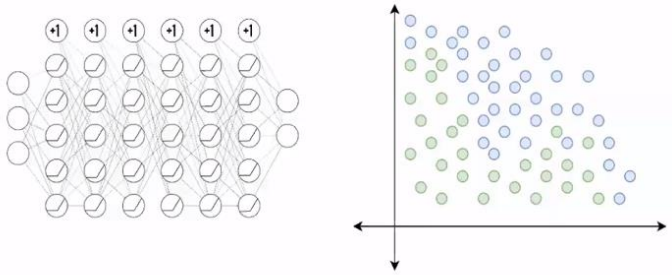
Vanishing Gradient Problem

Vanishing Gradient

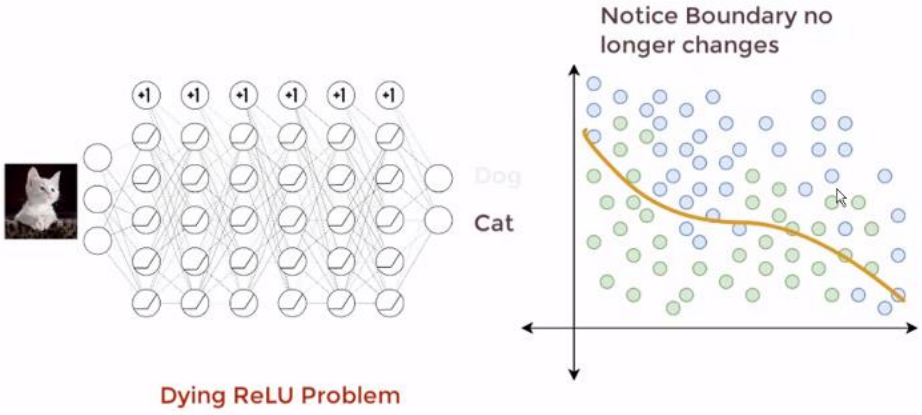


Vanishing Gradient Problem

Remedy to vanishing gradient is to use activation function that doesnot squeeze values, like ReLu

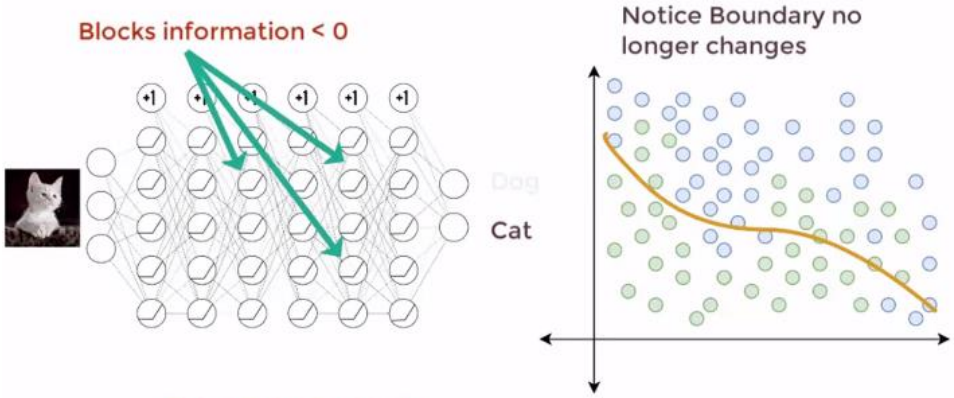


During backpropogation the gradients become smaller and smaller and smaller until they eventually vanish no gradients means no learning. Same problem like vanishing gradient

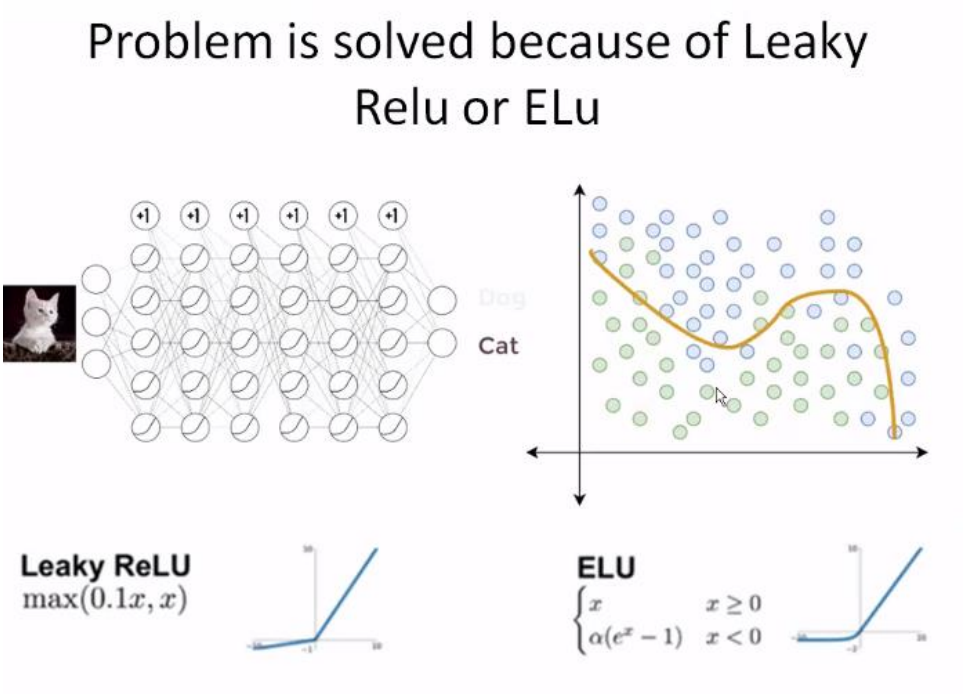


Dying ReLU Problem

Dying ReLu causes as it blocks information less than 0



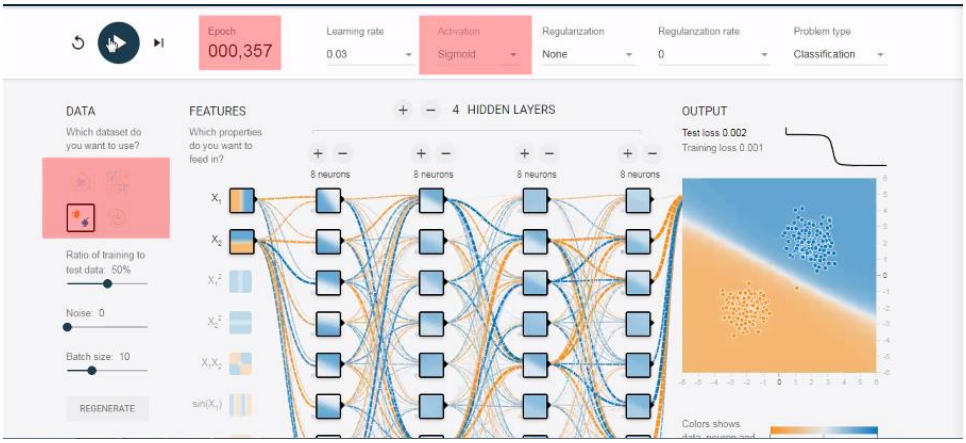
Dying ReLU Problem



Hyper Parameter Tunning (values to be found with trial and error)

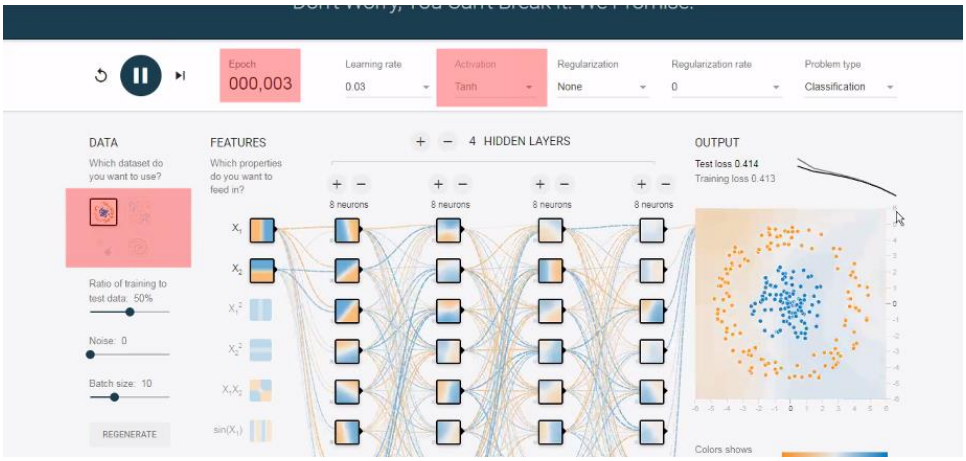
1)Num of Neurons in Hidden Layer
2)Num of Hidden Layers
3)Num of Epochs
4)Activation Functions
5)batch_size

Explain activation function in tensorflow playground:

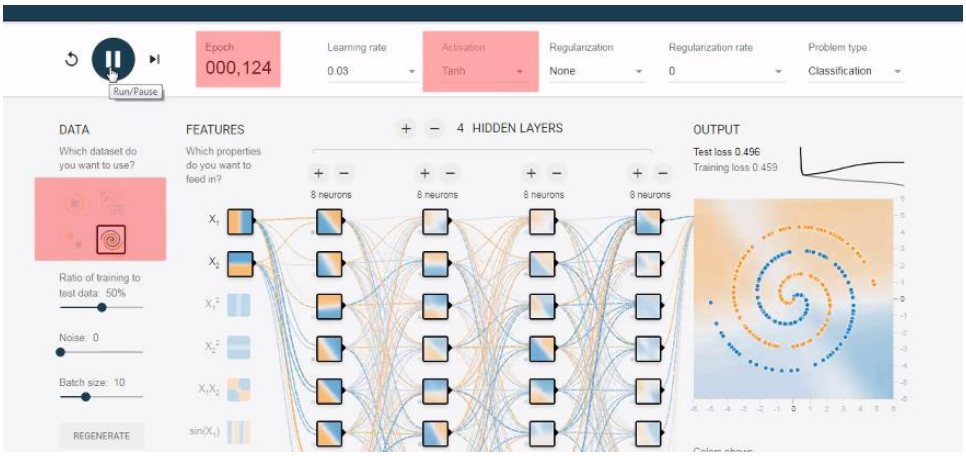


Took too much time to find the boundary

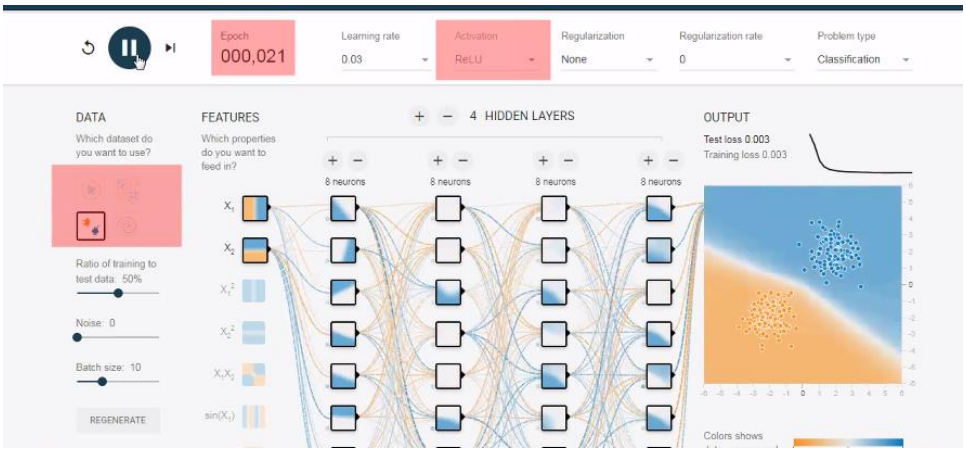
Tanh – advanatge – bigger boundaries

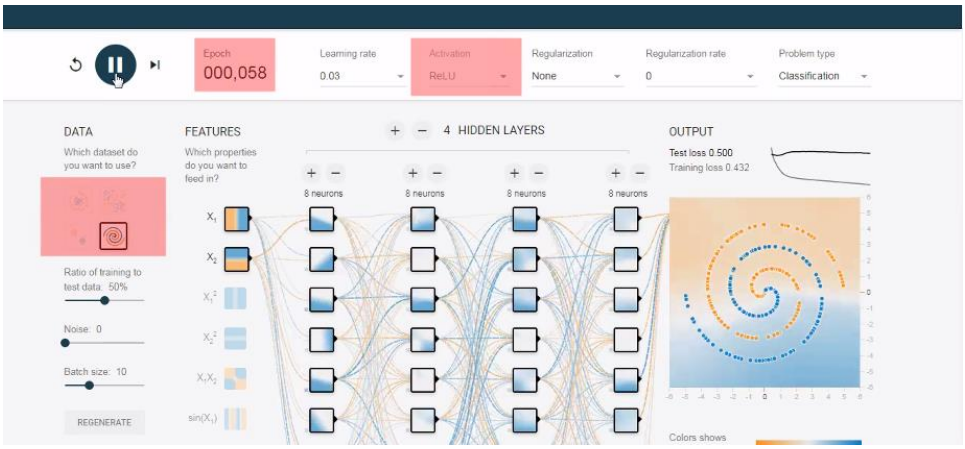


Problem of tanh:



ReLu: Identify the boundary in few numbers of epoch





Guidelines to choose the activation function:

Hyper Parameter Tunning (values to be found with trial and error)

1)Num of Neurons in Hidden Layer
2)Num of Hidden Layers
3)Num of Epochs
4)Activation Functions
5)batch_size

	InputLayer	HL	HL	OutputLayer	LossFunction
Regression		ReLu	ReLu	ReLu	mean_squared_error
Binary-Classification		ReLu	ReLu	Sigmoid	binary_classification
Multi-Class-Classification		ReLu	ReLu	Softmax	categorical_crossentropy

Implementation:

Step 1: Start anaconda prompt

Step 2:

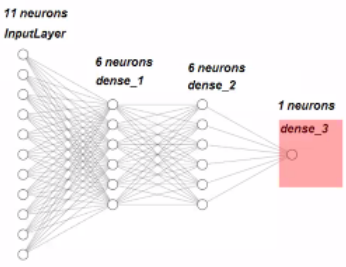
python -m pip install --upgrade pip

pip install keras tensorflow

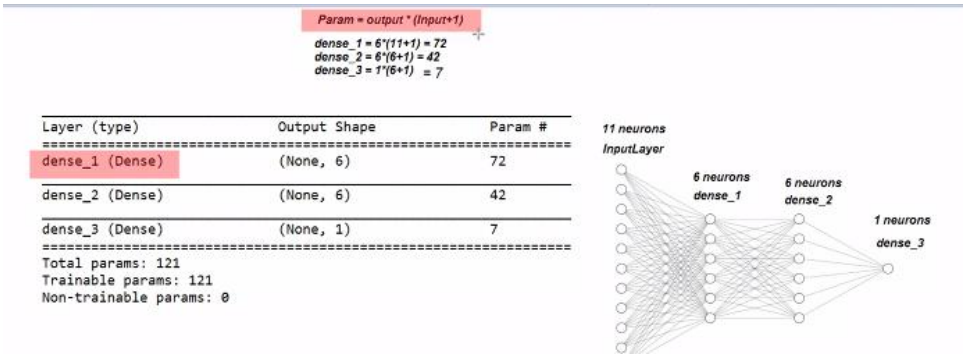
ANN:

Summary:

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 6)                 72
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 42
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 7
=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
```



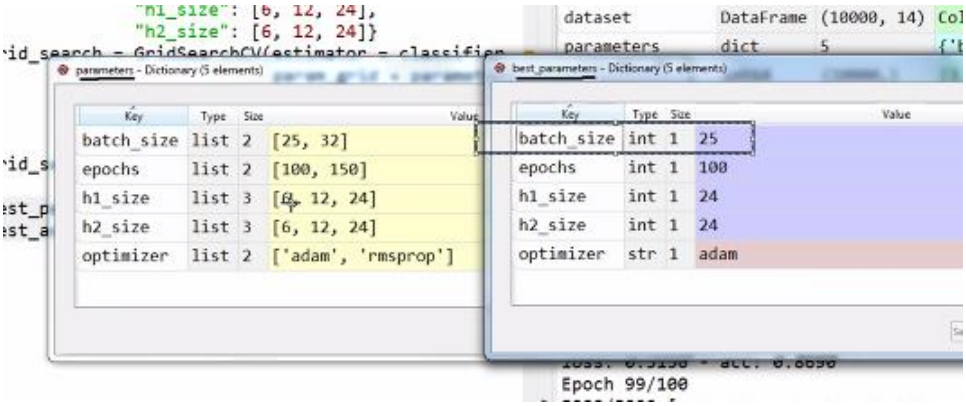
How to get the param count?

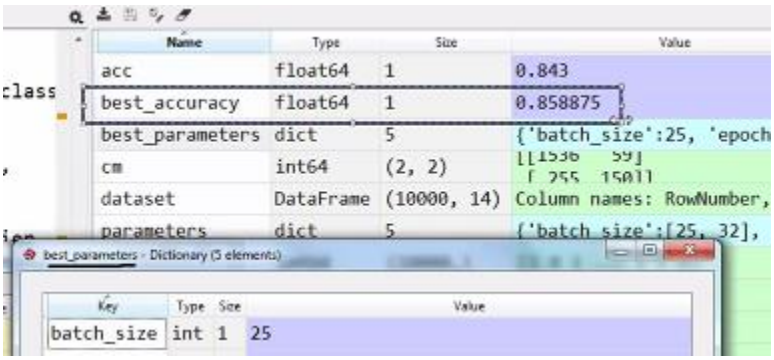


How many times I need to run?

Accuracy not stable – For that use GridSearchCV(sklearn Obj) - Trail and error with k fold cross validation

Neural network Keras object ------typecast --------sklearn obj





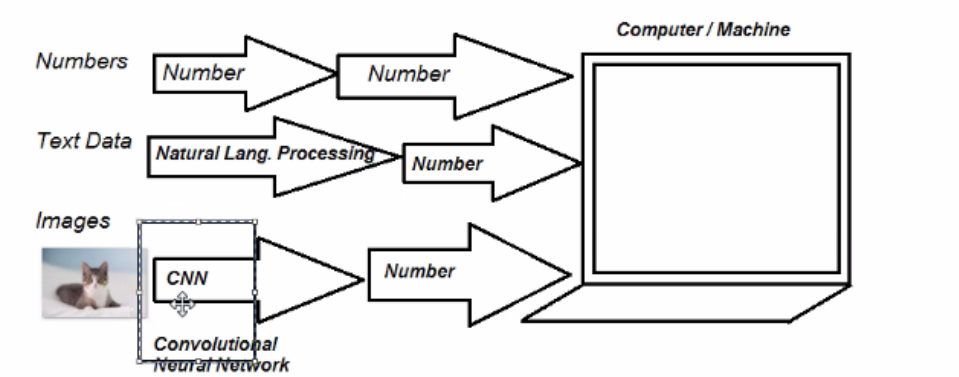
SGD:

Batch size 1 -> Slow learning -> Overfitting

BGD:

Batch size 8000 - > Fast learning -> Accuracy didn't increase – stays at 79% -> Too much of generalization

CNN:


Computer / Machine
Numbers
Number
Number
Text Data
Natural Lang. Processing
Number
Images
CNN
Number
Convolutional
Neural Network

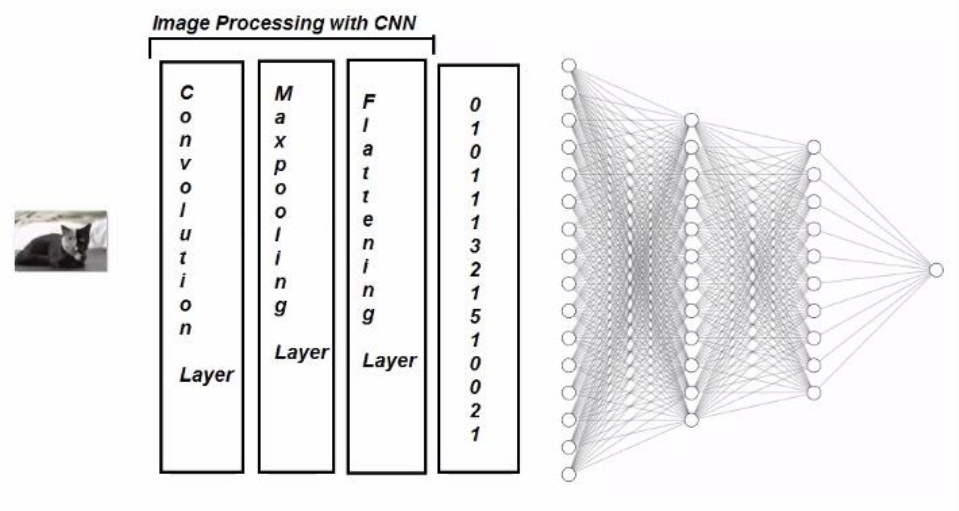


Image Processing with CNN
Convolution Layer
Maxpooling Layer
Flattening Layer
01011132151002 1

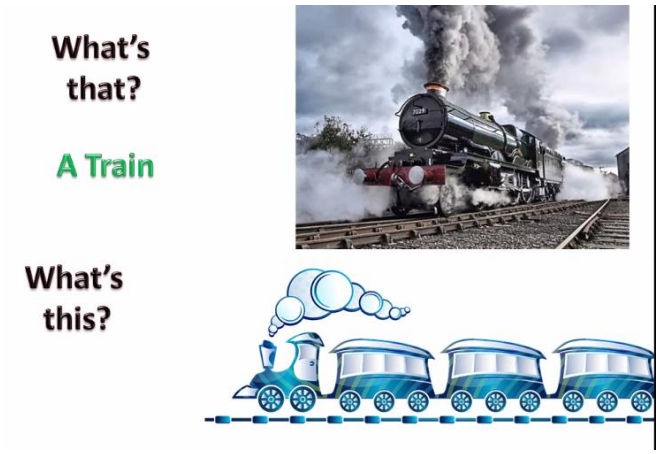

X_train
Input Data
y_train
Output Data
Dog
Cat
Dog
Cat
Cat
X_test
y_pred
????
????

Convolution Neural Network

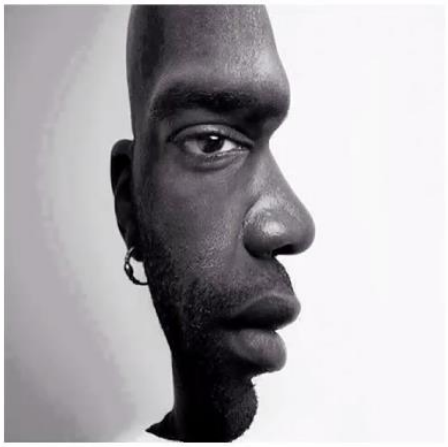
- Facebook uses neural nets for their automatic tagging algorithms
- Google for their photo search
- Amazon for their product recommendations
- Pinterest for their home feed personalization
- Instagram for their search infrastructure.

How Humans identify images??

- For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults.
- Most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing.

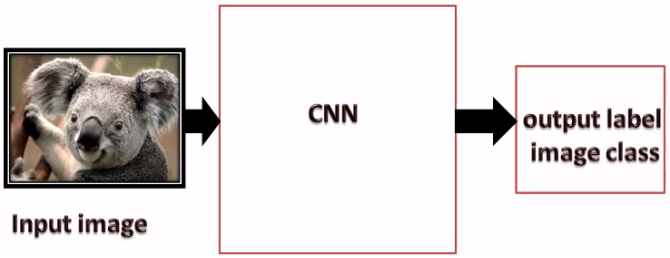


illusions

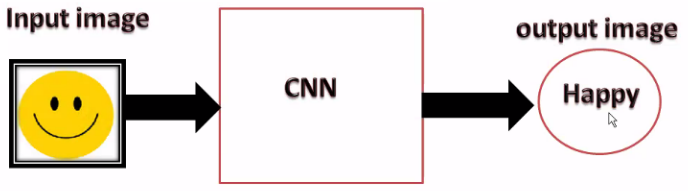


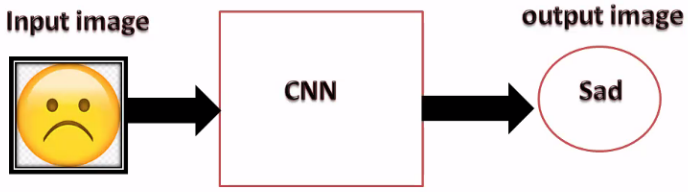
illusions

Convolution Neural Network



Convolution Neural Network





How Does Computer See an image??

When a computer sees an image ,it will see an array of pixel values. Depending on the resolution and size of the image, it will see a 32 x 32 x 3 array of numbers (The 3 refers to RGB values).

What we see

What computers see

How Does Computer See an image??

What we see

What computers see

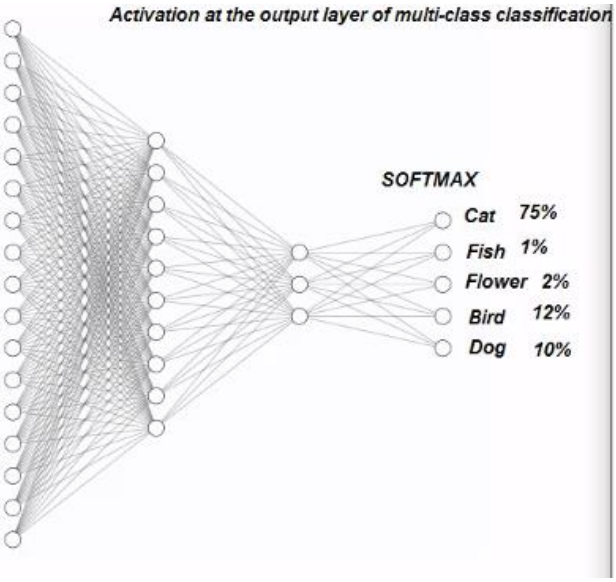
These numbers, when we perform image classification, are the only inputs available to the computer.

The idea is that you give the computer this array of numbers and it will output numbers that describe the probability of the image being a certain class

(80% for flower, 15% for sky etc).

Multi-Class Classification

	y0	y1	y2	y3	y4	
	class0	class1	class2	class3	class4	
	Cat	Fish	Flower	Bird	Dog	
Input Image						
	75%	1%	2%	12%	10%	=100%



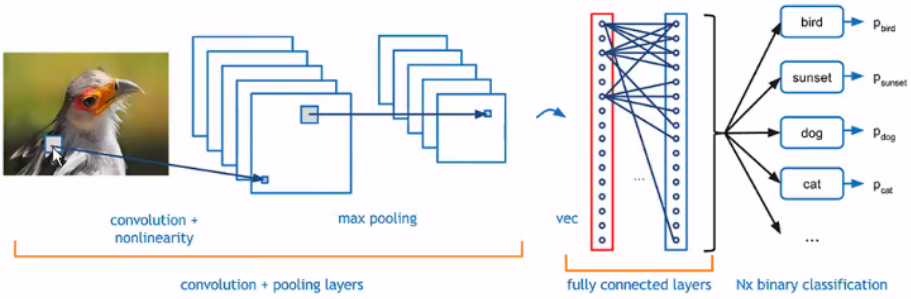
	Cat	Fish	Flower	Bird	Dog
			y_train		
X_train	y0	y1	y2	y3	y4
	1	0	0	0	0
	0	0	0	0	1
	1	0	0	0	0
	0	0	0	0	1

What We Want the Computer to Do

- we want the computer to do is to be able to differentiate between all the images it's given and figure out the **unique features** that make a dog a dog , that make a cat a cat or that make a flower a flower .

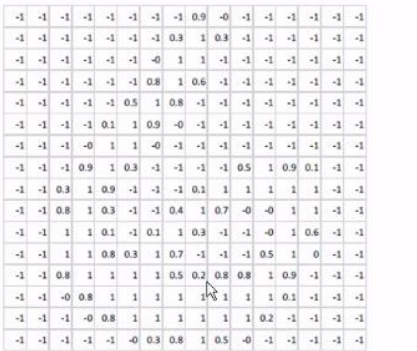
Convolution Neural Network

CNNs take the image, pass it through a series of convolutional, nonlinear, pooling (downsampling), and fully connected layers, and get an output.



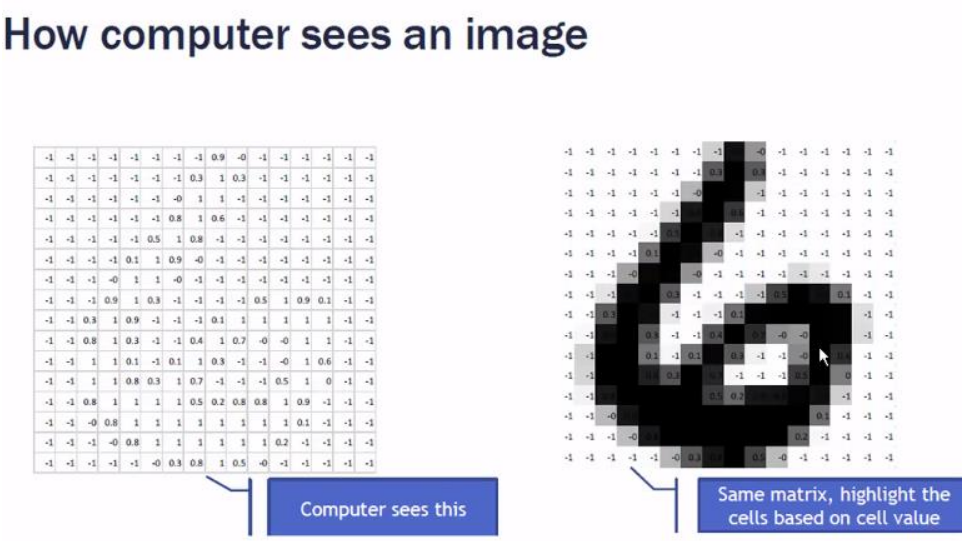
How computer sees an image

Humans see this

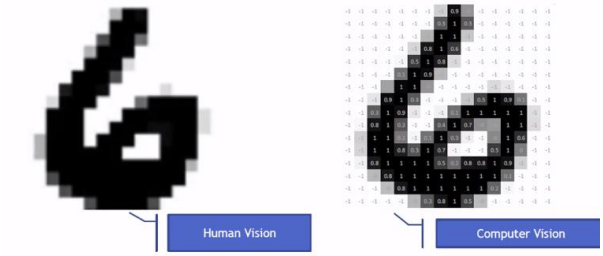


Computer sees this

How computer sees an image

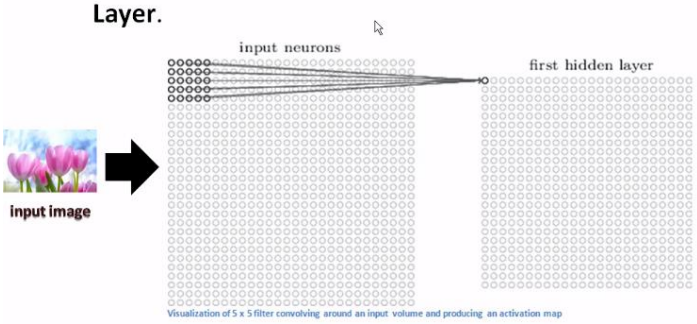


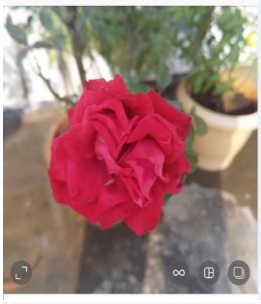
How computer sees an image

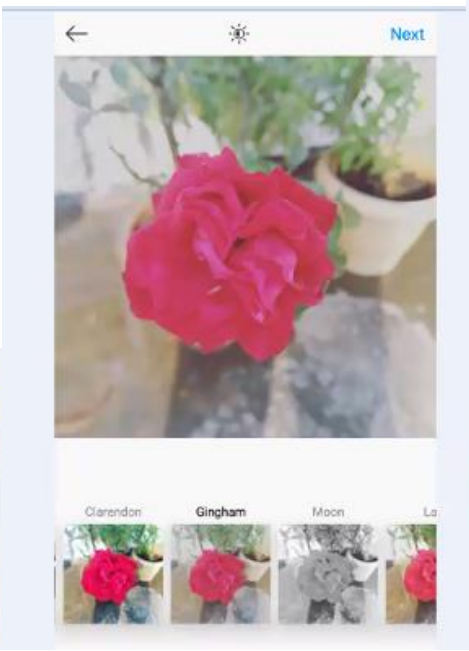


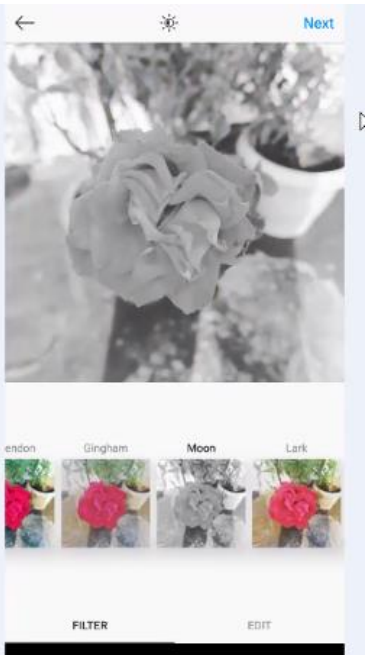
Convolution Neural Network

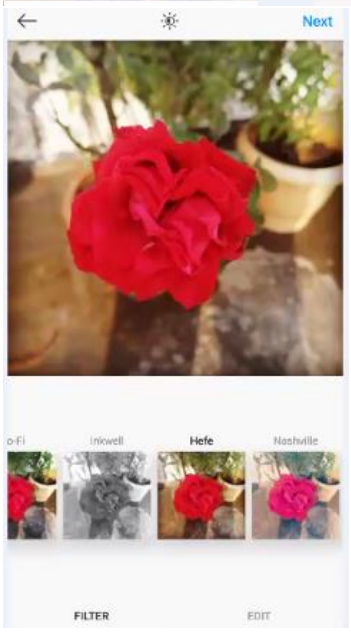
- **First Layer – Math Part**
  - The first layer in a CNN is always a **Convolutional Layer**.

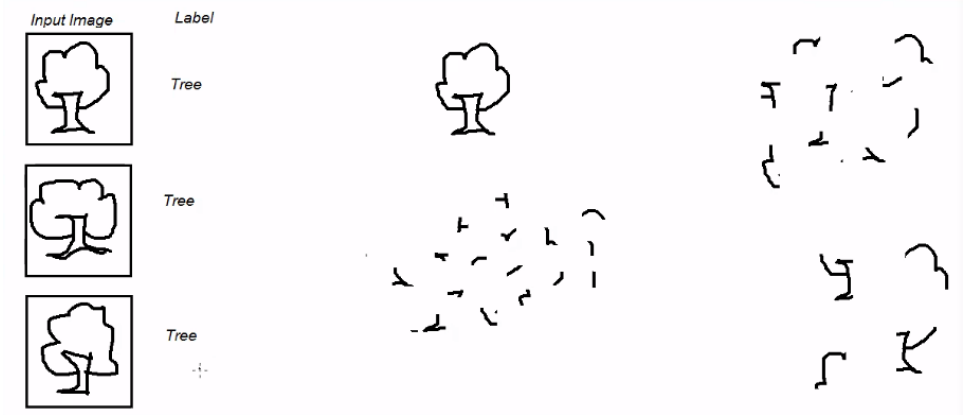


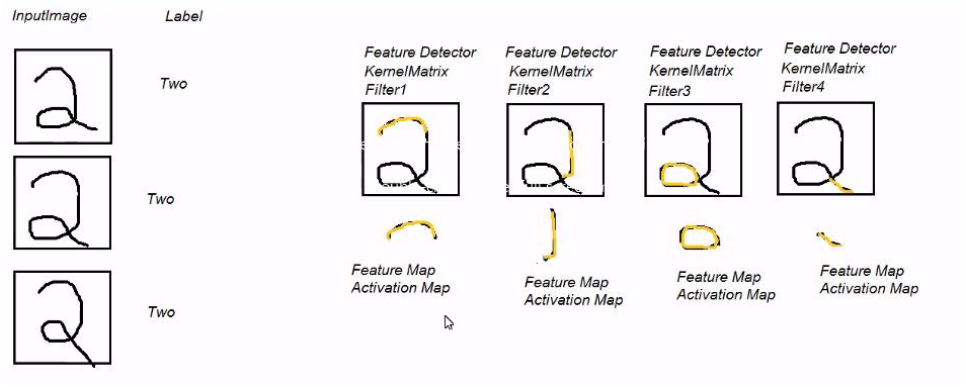



Next
Clarendon
Gingham
Moon


Next
Gingham
Moon
Lark
FILTER
EDIT

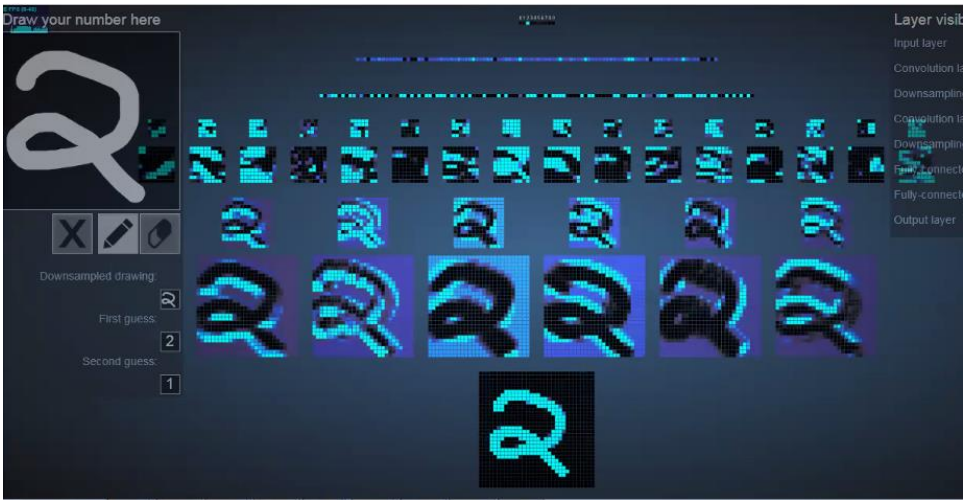

Next
Inkwell
Hefe
Nashville
FILTER
EDIT

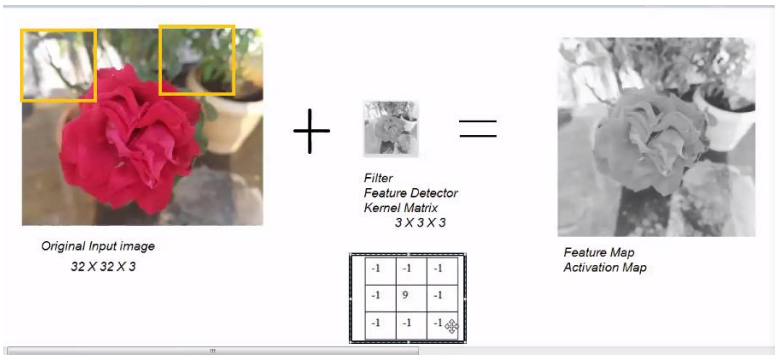

Input Image
Label
Tree
Tree
Tree



http://www.cs.cmu.edu/~aharley/nn_vis/cnn/2d.html

This identify the edges:





Kernel Matrix examples

1	1	1
1	1	1
1	1	1

Unweighted 3x3 smoothing kernel

0	1	0
1	4	1
0	1	0

Weighted 3x3 smoothing kernel with Gaussian blur

0	-1	0
-1	5	-1
0	-1	0

Kernel to make image sharper

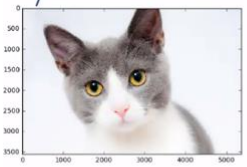
-1	-1	-1
-1	9	-1
-1	-1	-1

Intensified sharper image

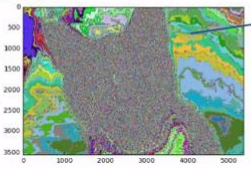
Gaussian Blur

Sharpened image

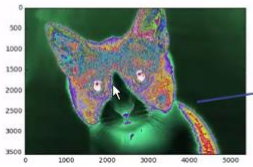
Convoluted features



Different filters capture different features



A filter to capture overall shape and edges



A filter to ignore white spaces on image

The depth in convolution layer

- Every filter gives us a resultant matrix (activation map)

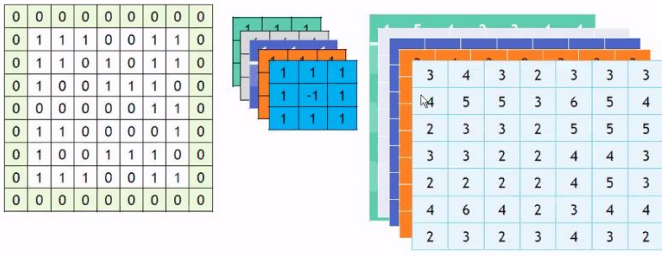
0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	1	1	0
0	1	1	0	1	0	1	1	0
0	1	0	0	1	1	1	0	0
0	0	0	0	0	0	1	1	0
0	1	1	0	0	0	0	1	0
0	1	0	0	1	1	1	0	0
0	1	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0	0

1	1	1
1	1	1
1	1	1

4	5	4	2	3	4	4
5	6	5	4	6	6	5
3	3	3	3	6	6	5
3	3	2	2	4	5	4
3	3	2	2	4	5	4
5	6	4	3	4	5	4
3	4	3	3	4	4	3

The depth in convolution layer

- If we apply 10 different filters then we will get 10 resultant matrices. The depth is 10



Max Pooling

0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Pooling

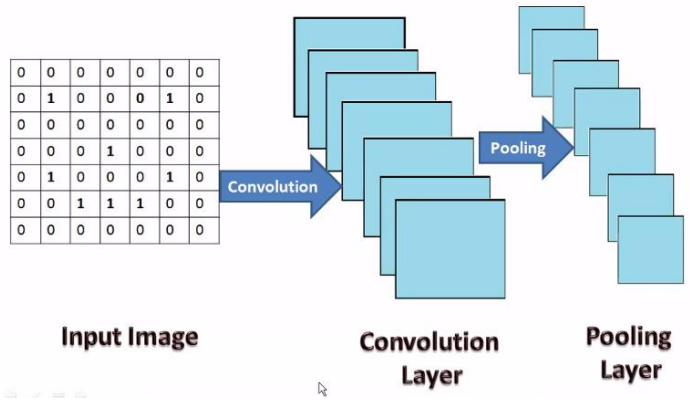
1	1	0
4	2	1
0	2	1

Feature Map

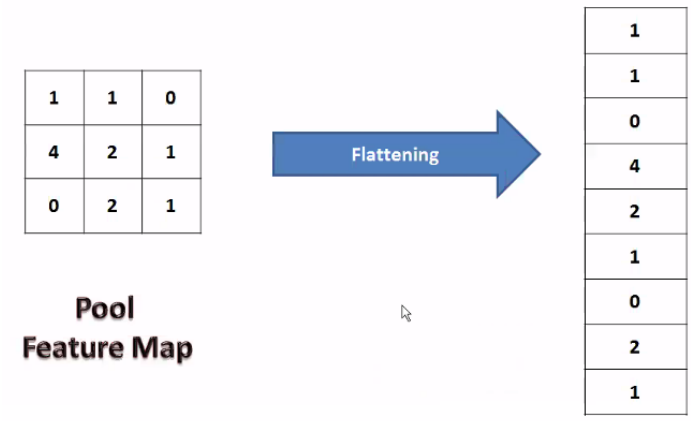
Pool Feature Map

Max pooling is usually done with 2x2 windows and stride 2, so as to downsample the feature maps

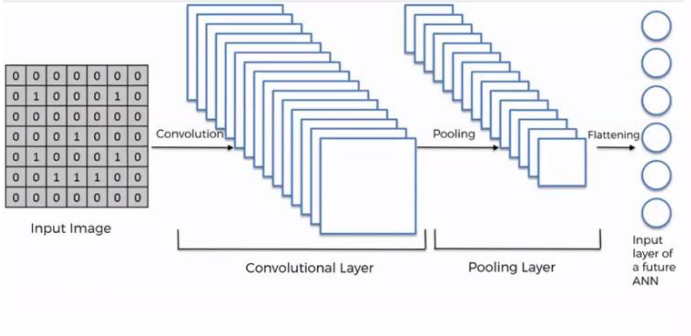
Max Pooling



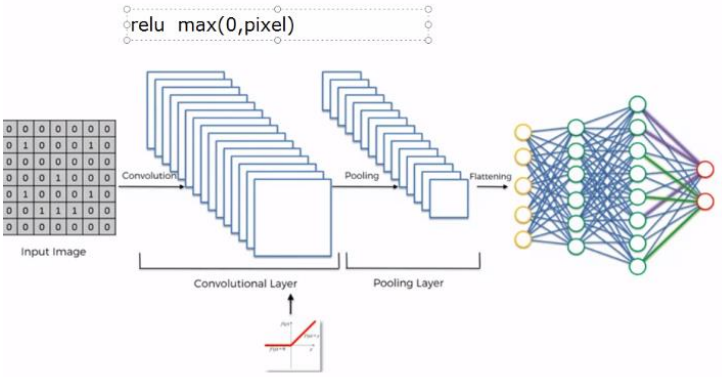
Flattening



Flattening

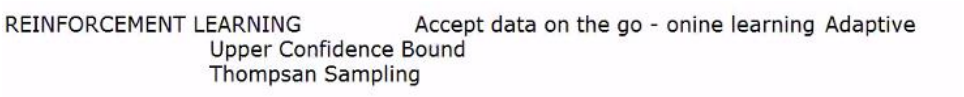


Full Connection



Data Science Training - Day 5

Reinforcement learning



- Adaptive learning!
- Provide data on the go
- Code the nature of behavior of the machine
- Lot of flexibility we require to code
- Not use any ready-made packages
- Use raw model
- Popularly used in robotics

At any given situation, robotics takes right step, then it is provided with positive reward. If robotics doesn't take proper steps to overcome the hurdle, then it is provided with negative reward.

Example 1:

Positive rewards: Provide smile and hug

Negative reward: Giving warning

Growing with good habits

Example 2: It is also used in e-commerce applications

Utilizing the existing information - people searching the product, bought, clicks –Product moves up

At the 66th minute, we declare the product as trending!

Trending product of the hours

Example 3: Crossing the bridge for the first time in the life.

None of them knows how to do that

First person is at risk

First person uses all his common sense and knowledge and takes steps

Second person uses the first person and perform the steps

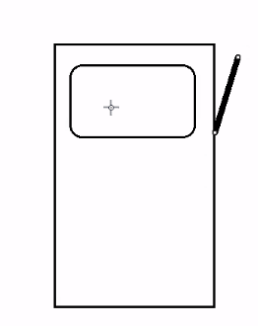
Third person sees the first and second person and executes the better step.

Best possible decision is made by watching out the previous step.

Thompsan Sampling:

Multi arm banded machine

Take a pully example: When we pull the pulley (black bolded one) - A number will be displayed in the LCD display.



These are the casino machines

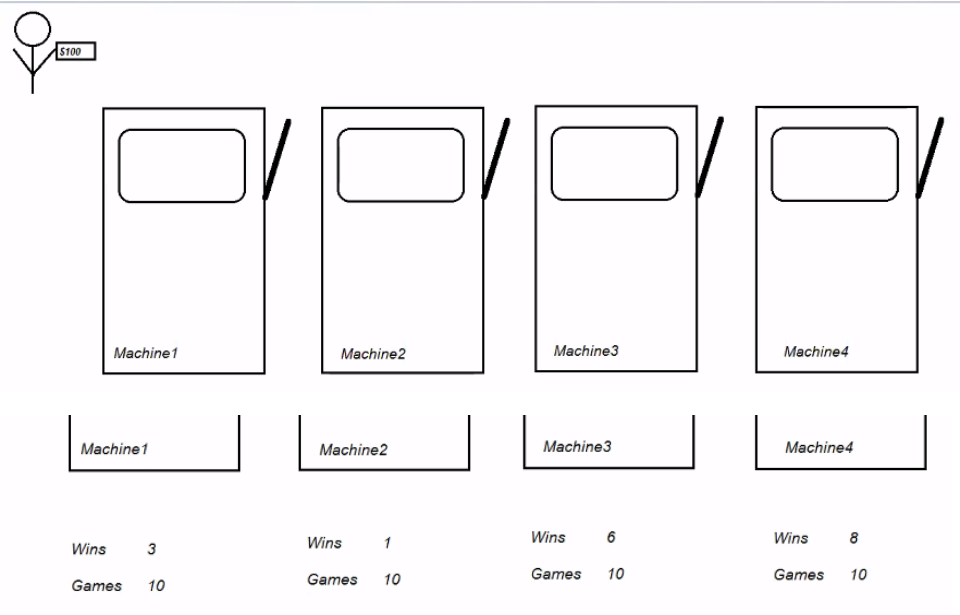
Out of these four what is the best – won't be disclosed

Don’t know in which machine we should spend all these money

Only one machine will be good.

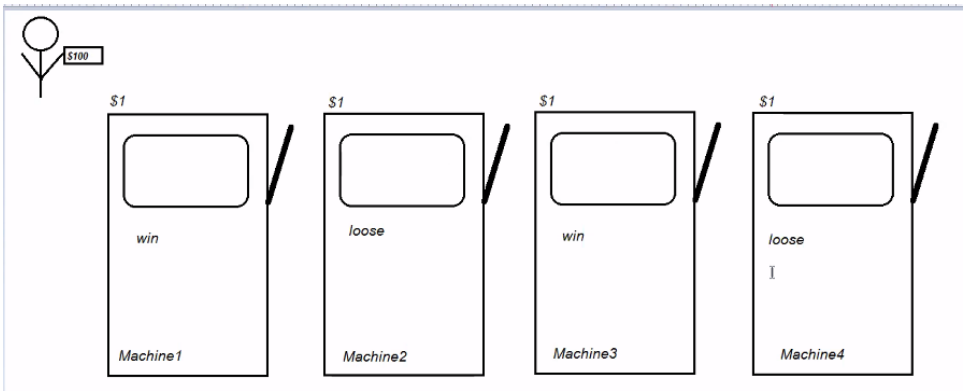
We will be in Delamo – which one is best?

Best is only own by the casino owners



You can see machine 4 is best after playing.

We will take decision after playing the game and based on the win.

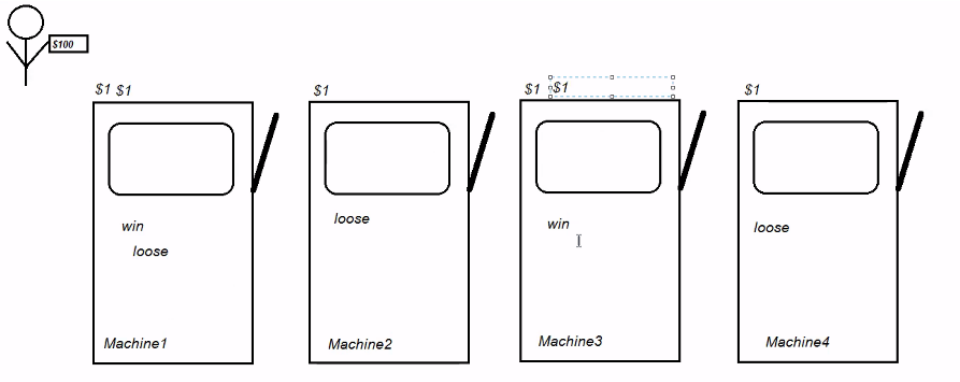


Our agenda is to find in which machine we win.

Now, where we will put the 5th dollar? Where to spend it?

We will go to the machine which we win

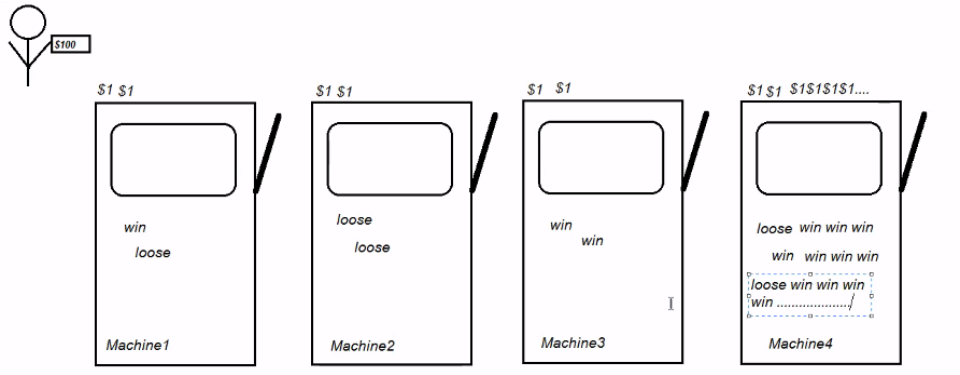
(machine 1 – lucky one consider – put the dollar – but you loose



Eventually play more and decide or find out the best machine

Here we perform **exploration – exploitation**.

Explore all our choices and then exploit our best choices.



What is the use case we are going to solve?

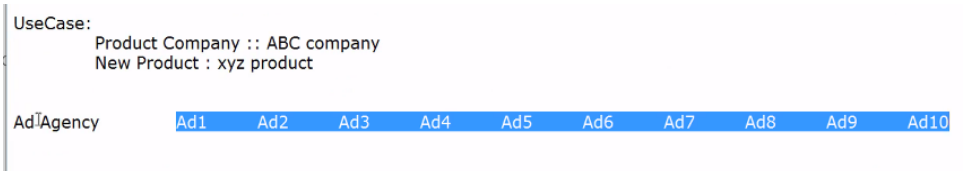
Product company: ABC company

New Product: xyz product

TO promote this, they add “AD agency”.

Start developing very attractive ads. Say they prepare 10 variations.

They call product owners to choose the best variates. They will say customers to choose the best.



Make use of social media platforms. They are giving us 10000 rounds Free

1 Round == 1 user connection the social media account.

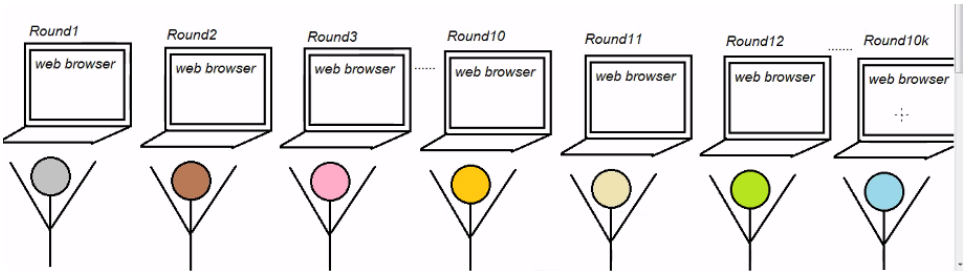
People are Unknowingly targeted for their product to do the survey.

Example:

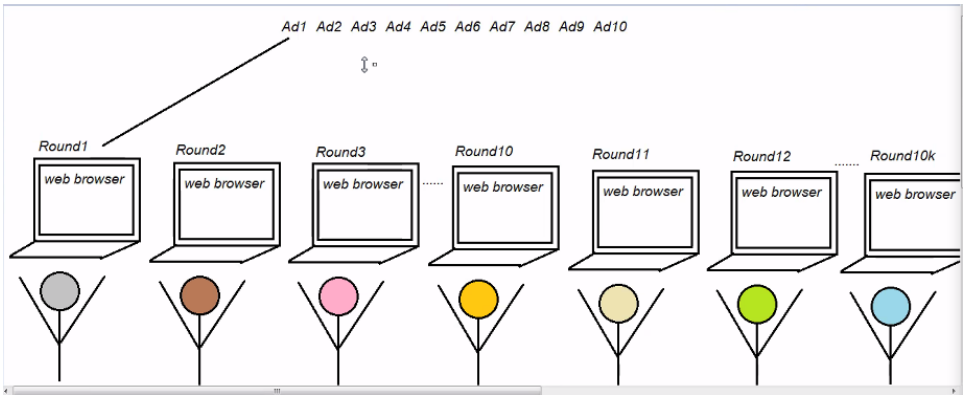
Youtube – ads floating in the starting – Fresh contents – something attracts you – you click – Interest captured with the click.

If you ignore that, considered as not interested.

If you keep on seeing it after 5 seconds, then you are showing the interested.



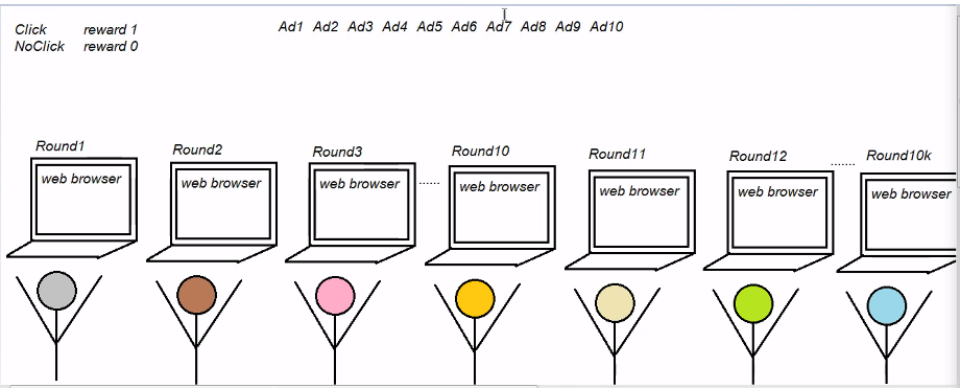
Independently connecting by the interest.



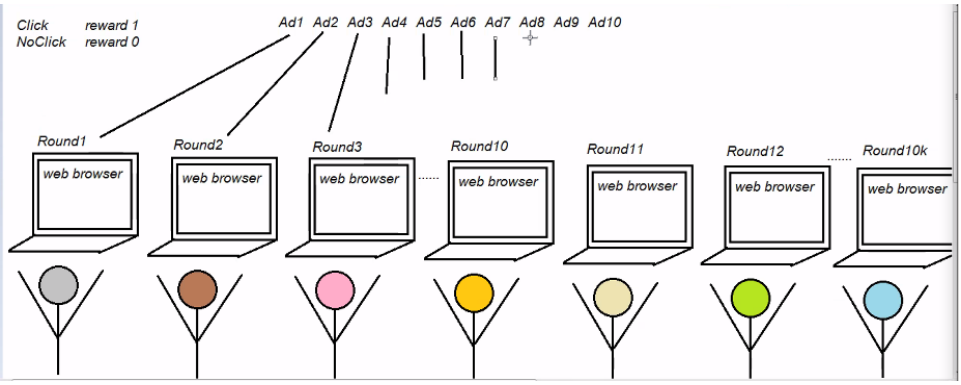
Only one add is displayed on the user

If someone likes it – click – reward 1

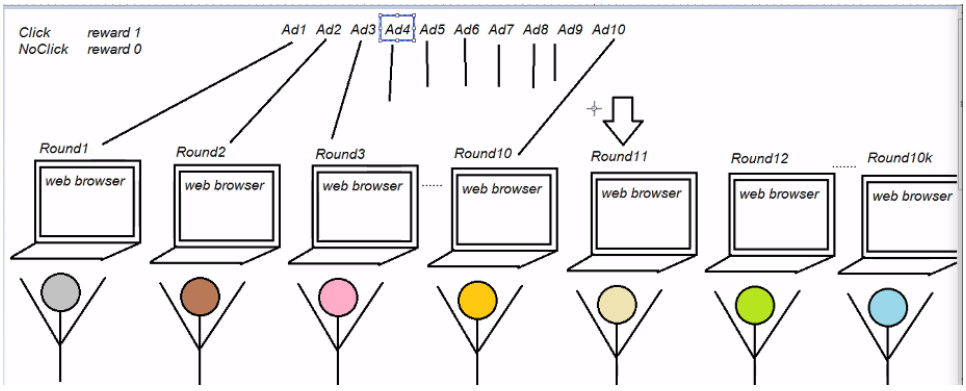
No click – reward 0



Give everyone one chance



Now, the question is what ad should I show him?

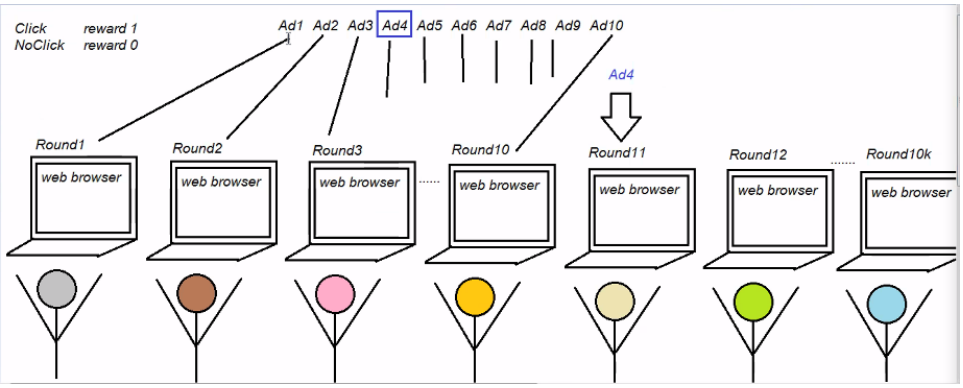


Consider Ad 4 is gained a click!

If none of them gets click, then we will give all ads another chance.

We will provide the Ad 4 (exploit) to other users and gain the click count.

Who has got the max reward? what if only ad4 gets more chance – logic becomes unbiased



Round 15 , Ad4 has got 6 chances it has earned 6 rewards

Average Reward = TotalRewards / NumberOfChancesToDisplay
= 6/6 = 1

Every time Ad 4 is given a chance to appear in the user browser, it gains the reward.

Round 20 , Ad4 has got 11 chances its has earned 6 rewards

Average Reward = 6/11 = 0.54

Rewards didn't increase but rounds got increased!

So, the average reward decreases.

Here, they won't survive the competitions.

How the ads get the chances, if the average reward is 0

When they will be getting chance?

Everyone will be given grace rewards.

Small bias is provided by this grace rewards.

graceRewards	*Number Of Chances a Ad has got* *If a Ad has got more chances to display "**graceRewards**" will be very small value* *If a Ad has got less chances to display "**graceRewards**" will be large value*

Upper Confidence Bound

- There are many algorithms to optimize the decision making behaviour of the agent, some perform better than others.
- A very popular method is the UCB exploration strategy
- This algorithm chooses the arm based on the average reward mean plus an exploration bonus.
- The exploration bonus is dependent on the number of times the action has been tried out before and the total number of action selections.

- We have **d** Ads that we display to users each time they connect to web page.
- Each time a user connects to this web page, that makes a **round**
- At **each round n**, we choose one Ad to display to the user.
- At each **round n**, **Ad i** gets reward
- if the user clicked on Ad $r_i(n) \in \{0,1\}: \; r_i(n) = 1$
- if the user didnt then 0
- The goal is to **maximize the total reward** we get over many rounds

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i(n)$ - the number of times the ad i was selected up to round n,
- $R_i(n)$ - the sum of rewards of the ad i up to round n.

Step 2. From these two numbers we compute:

- the average reward of ad i up to round n

$$\bar{r}_i(n) = \frac{R_i(n)}{N_i(n)}$$

- UCB $\bar{r}_i(n) + \Delta_i(n)$

$$\Delta_i(n) = \sqrt{\frac{3}{2}\frac{\log(n)}{N_i(n)}}$$

Grace reward is the delta i.

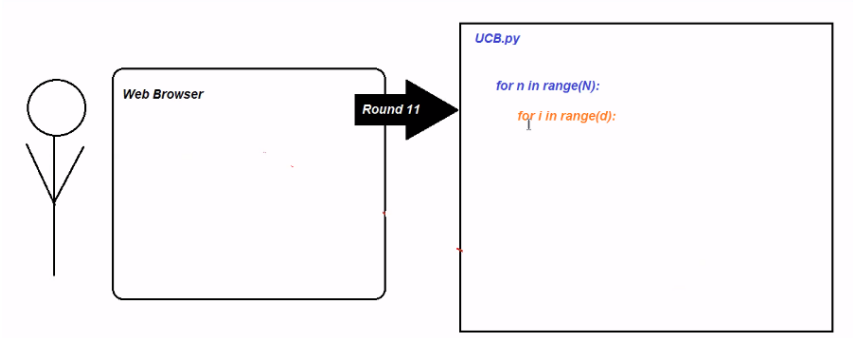
Less chance – more grace reward

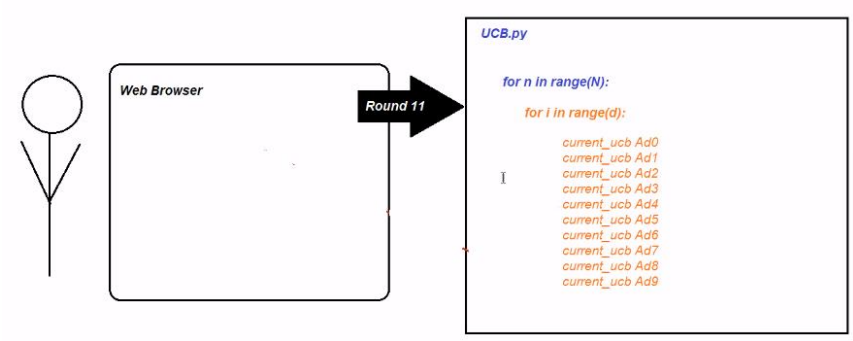
More chance – less grace reward

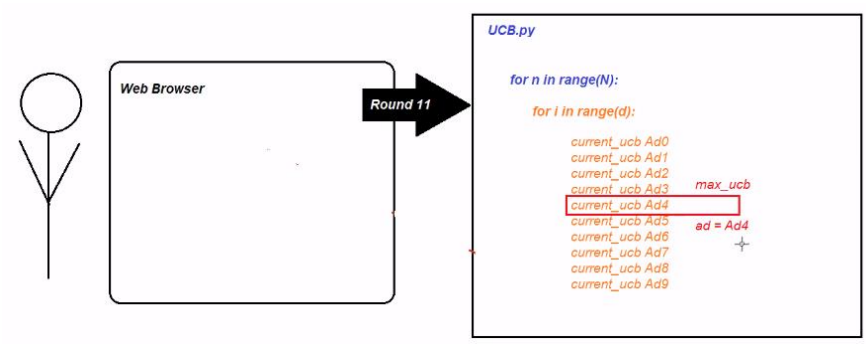
There are lots of things happen for not click. Though the Ads is potential, this grace marks will help to get the ads more clicks.

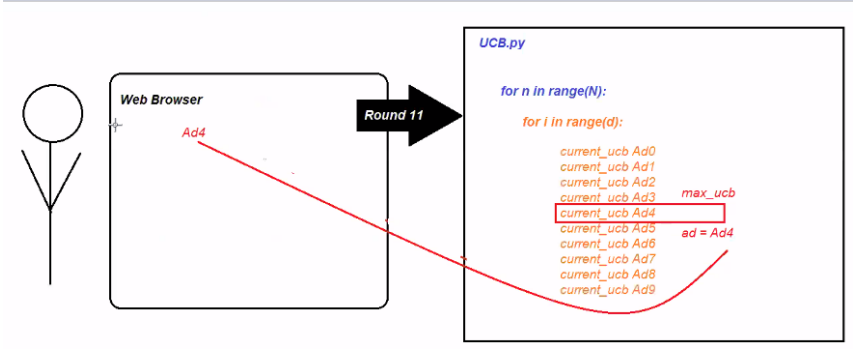
Situation – power cut, accidently click somewhere or another web browser opens.

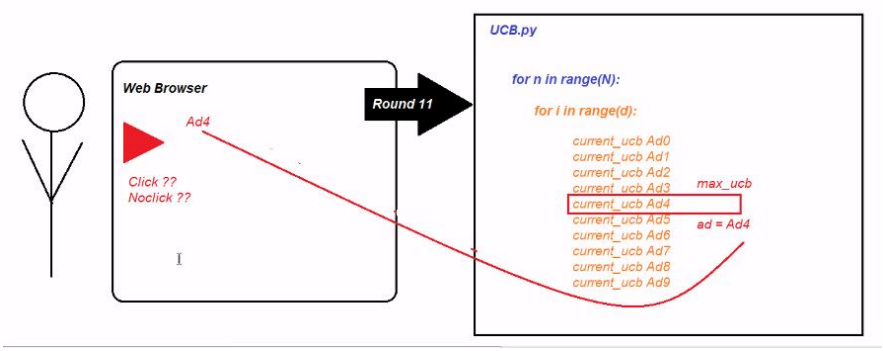
Step 3. We select the ad i that has the maximum UCB $\bar{r}_i(n) + \Delta_i(n)$.








Web Browser
Ad4
Round 11
UCB.py
for n in range(N):
for i in range(d):
current_ucb Ad0
current_ucb Ad1
current_ucb Ad2
current_ucb Ad3
current_ucb Ad4
current_ucb Ad5
current_ucb Ad6
current_ucb Ad7
current_ucb Ad8
current_ucb Ad9
max_ucb
ad = Ad4


Web Browser
Ad4
Click ??
Noclick ??
Round 11
UCB.py
for n in range(N):
for i in range(d):
current_ucb Ad0
current_ucb Ad1
current_ucb Ad2
current_ucb Ad3
current_ucb Ad4
current_ucb Ad5
current_ucb Ad6
current_ucb Ad7
current_ucb Ad8
current_ucb Ad9
max_ucb
ad = Ad4

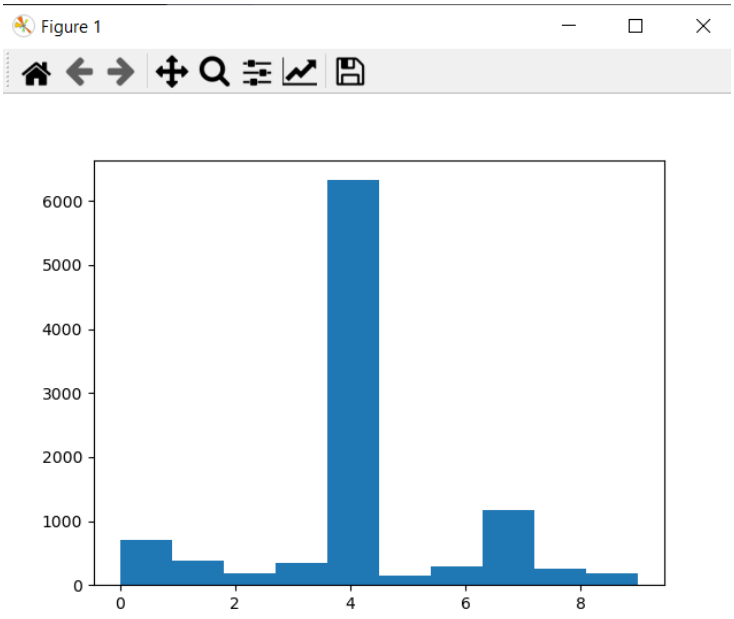
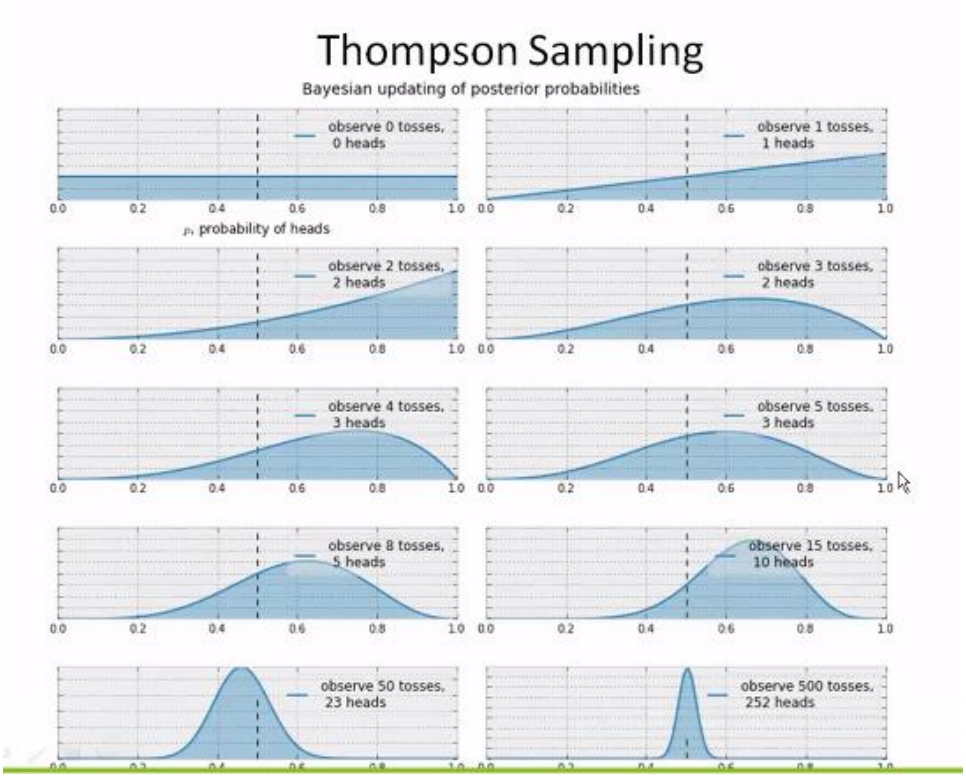


Figure 1
6000
5000
4000
3000
2000
1000
0
0
2
4
6
8



Probabilistic behavior

Posterior Probability

P(A|B) = P(A) * P(B|A) / P(B)

P(A|B) probability of Event A given Event B has occured

Probability of the Ad to get a click given it has got 10 noclicks

For each Ad we need to Capture

number of clicks
number of no-clicks

Thompson Sampling:

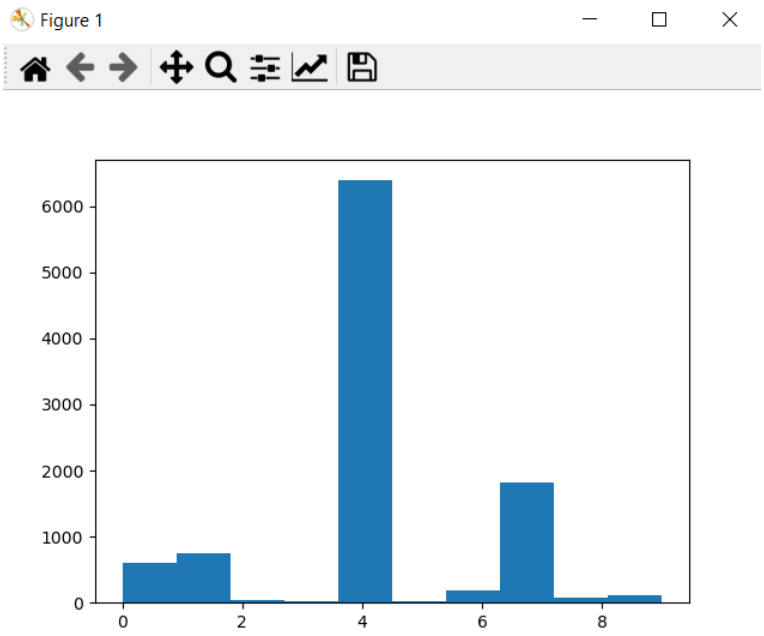
Thompson Sampling

Step 1. At each round n, we consider two numbers for each ad i:

- $N_i^1(n)$ - the number of times the ad i got reward 1 up to round n,
- $N_i^0(n)$ - the number of times the ad i got reward 0 up to round n.

Step 2. For each ad i, we take a random draw from the distribution below:

$$\theta_i(n) = \beta(N_i^1(n) + 1, N_i^0(n) + 1)$$



NLP:

NLTK – NL tool kit

Natural Lang Processing (nltk) Natural Lang. Tool-Kit

Review	Liked
Wow....Loved this place.	1
Crust is not that good :(	0
Great place will come back :) !!	1

Step1: Substitue all non-alphabets with a space , using python re package

Wow....Loved this place
Crust is not that good
Great place will come back

Wow Loved this place
Crust is not that good
Great place will come back

Step2: Convert the review to lower case , python lower()

wow loved this place
crust is not that good
great place will come back

Step3: Convert the stmt into tokens of words , using split() method in python
Tokenization

[wow, loved, this, place]
[crust ,is ,not ,that ,good]
[great ,place, will, come, back]

Step4:Eliminate Stopwords , using nltk stopwords

[wow, loved, place]
[crust ,not ,good]
[great ,place, will, come, back]

Step5:Stemming of words, using nltk stemmer

love	loved	lovable	lovely
love	love	love	love

[wow, love, place]
[crust ,not ,good]
[great ,place]

Step6: Join the words back to stmt, using join() method in python

wow love place
crust not good
great place

Step7: Convert Text to number , using CountVectorizer & Calculation of TFIDF-(Term Frequency and inverse document Freq)

	crust	good	great	love	not	place	wow
wow love place	0	0	0	1	0	1	1
crust not good	1	1	0	0	1	0	0
great place	0	0	1	0	0	1	0

Step7: Convert Text to number , using CountVectorizer & Calculation of TFIDF-(Term Frequency and inverse document Freq)

	crust	good	great	love	not	place	wow	Liked
wow love place	0	0	0	1	0	1	1	1
crust not good	1	1	0	0	1	0	0	0
great place	0	0	1	0	0	1	0	1
			X					Y

NLP

- TF-IDF
- TF => Trem Frequency
- IDF => Inverse document frequency

TF	=	Term Frequency
IDF	=	Inverse Document Frequency
TF-TDF	=	TF * IDF

TF

$$\frac{(Number\ of\ occurrences\ of\ a\ word\ in\ a\ document)}{(Number\ of\ words\ in\ that\ document)}$$

"to be or not to be"

to = $\frac{1+1}{6}$

to = 0.33

be = 0.33

or = 0.16

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

IDF

Formula

$$log(\frac{(Number\ of\ documents)}{(Number\ of\ documents\ containing\ word)})$$

$$log(\frac{(Number\ of\ documents)}{(Number\ of\ documents\ containing\ word)})$$

"to be or not to be"
"i have to be"
"you got to be"

to = $\log\left(\frac{3}{3}\right)$

to = 0

be = $\log(\frac{3}{3})$

be = 0

have = $\log(\frac{3}{1})$

Words	IDF Value
going	0
to	0.41
today	0.41
i	0.41
am	0.41
It	1.09
is	1.09
rain	1.09

Words/ Documents	Document 1	Document 2	Document 3
going	0.16	0.16	0.12
to	0.16	0	0.12
today	0.16	0.16	0
i	0	0.16	0.12
am	0	0.16	0.12
it	0.16	0	0
is	0.16	0	0
rain	0.16	0	0

Words/ Documents	going	to	today	i	am	it	is	rain
Document 1	0	0.07	0.07	0	0	0.17	0.17	0.17
Document 2	0	0	0.07	0.07	0.07	0	0	0
Document 3	0	0.05	0	0.05	0.05	0	0	0

$$TFIDF(Word) = TF(Document, Word) * IDF(Word)$$

Step8: Split the data into Train-Test, choose a algo, check accuracy

Implementation:

dataset - DataFrame

Index	label	review
0	neg	whatever promise the film starts with soon det… the script becomes so unspeakably bad that the…
1	neg	nobody . "
2	pos	he is part churchman and part politician . however the tension , like the heat , flies an…
3	pos	because of that , almost everything in this fi…
4	neg	the chopper with the fire-retardant chemicals …
5	neg	you'll get the hang of this . they open the stargate , a bunch of them go th…
6	neg	and , of course , she stalks , so we see a num… i won't reveal the ludicrous ending to this tu…
7	neg	in nothing to lose , there is hardly chemistry…
8	neg	these early scenes were interesting . we're shown the fashion world through a newcom…
9	neg	i could go into more of the plot specifics (s…
10	neg	the whole film seems to be in a race with itse…

dataset - DataFrame

Index	label	review
0	0	whatever promise the film starts with soon det… the script becomes so unspeakably bad that the…
1	0	nobody . "
2	1	he is part churchman and part politician . however the tension , like the heat , flies an…
3	1	because of that , almost everything in this fi…
4	0	the chopper with the fire-retardant chemicals …
5	0	you'll get the hang of this . they open the stargate , a bunch of them go th…
6	0	and , of course , she stalks , so we see a num… i won't reveal the ludicrous ending to this tu…
7	0	in nothing to lose , there is hardly chemistry…
8	0	these early scenes were interesting . we're shown the fashion world through a newcom…
9	0	i could go into more of the plot specifics (s…
10	0	the whole film seems to be in a race with itse… as was usual in the zucker brothers films and …
11	1	much of the time wasted could've been used to … even so , weaver and hunter act very well in t…

```
In [4]: dataset.fi
        dataset.fillna
        dataset.filter
        dataset.first
        dataset.first_valid_index
```

After dropna:

dataset	DataFrame	(1965, 2)

```
t LabelEncoder
transform(dataset
s():
e)
```

dataset - DataFrame

Index	label	review
53	1	at first glance , daylight …
54	1	the coen brothers are back …
55	1	some of my friends who went…
56	0	my son and i share a perver…
57	0	
58	0	how could a g-rated disney …
59	1	wow ! what a movie . it's e…

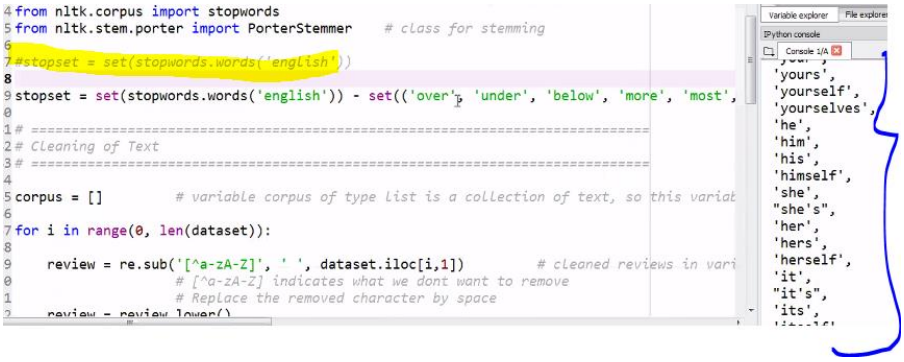
Identify the blank spaces:

blanks - List (27 elements)

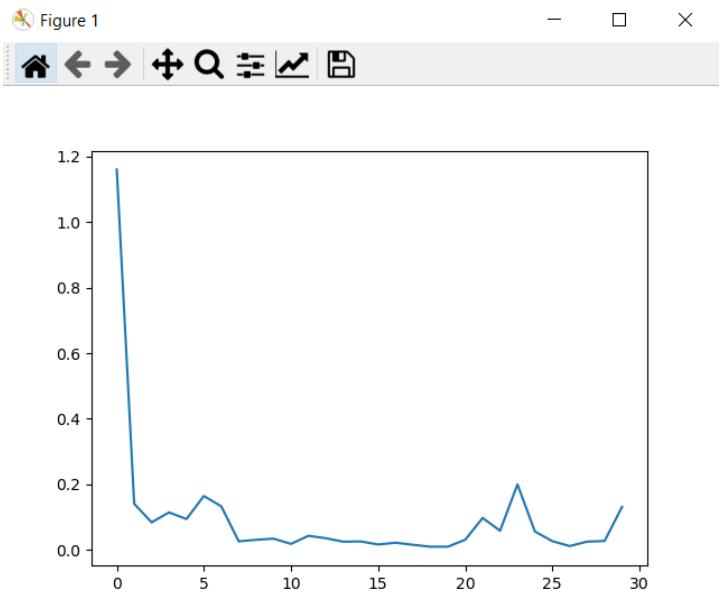
Index	Type	Size	Value
0	int	1	57
1	int	1	71
2	int	1	147
3	int	1	151
4	int	1	283
5	int	1	307
6	int	1	313
7	int	1	323
8	int	1	343
9	int	1	351
10	int	1	427
11	int	1	501

After dropping blanks:

dataset	DataFrame	(1938, 2)

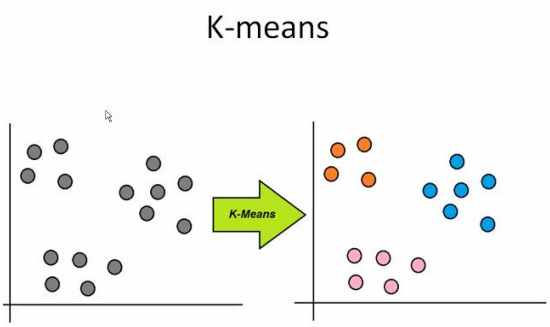


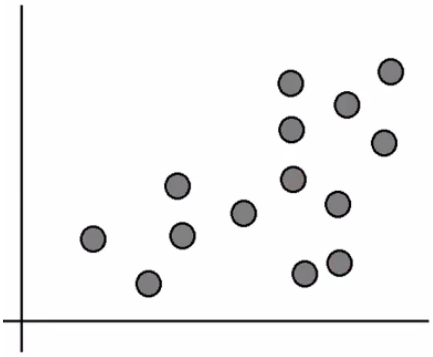
NLP using Deep learning:



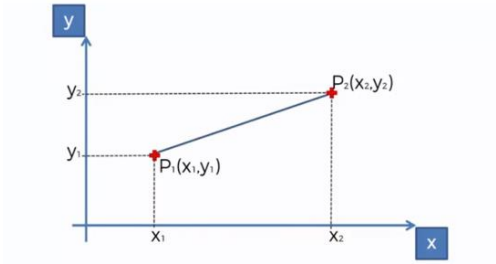
Unsupervised learning:

K means clustering

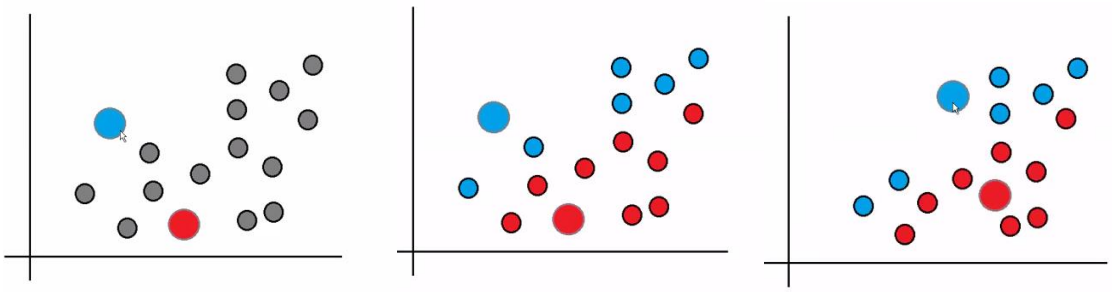


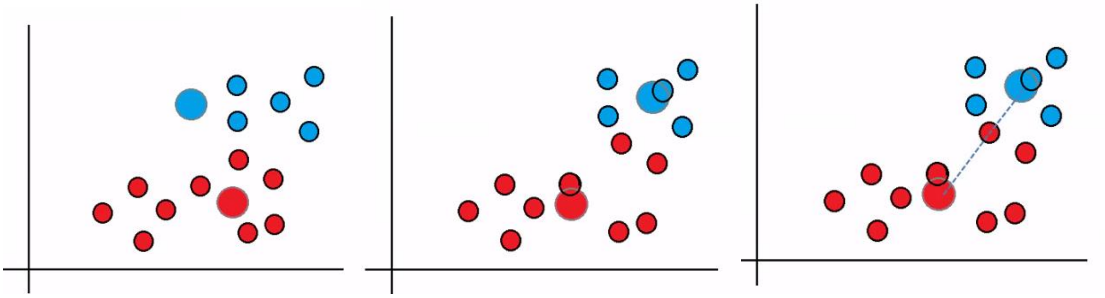


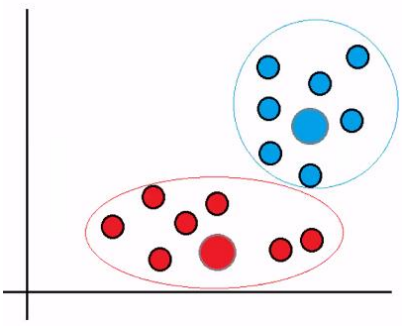
Euclidean Distance



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$







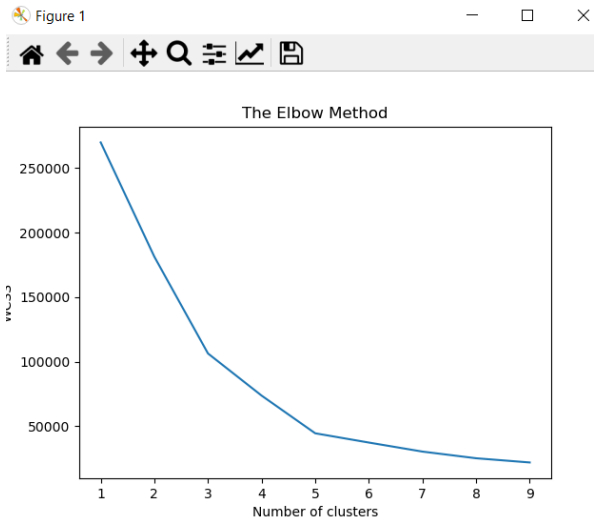
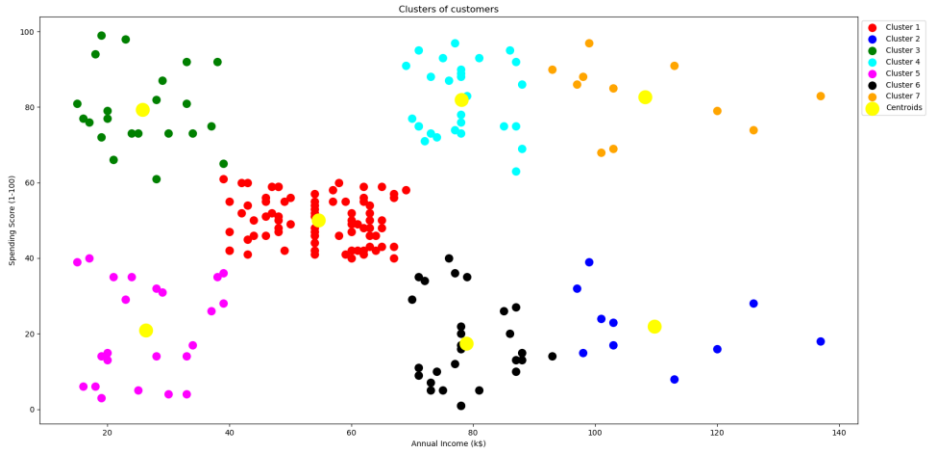
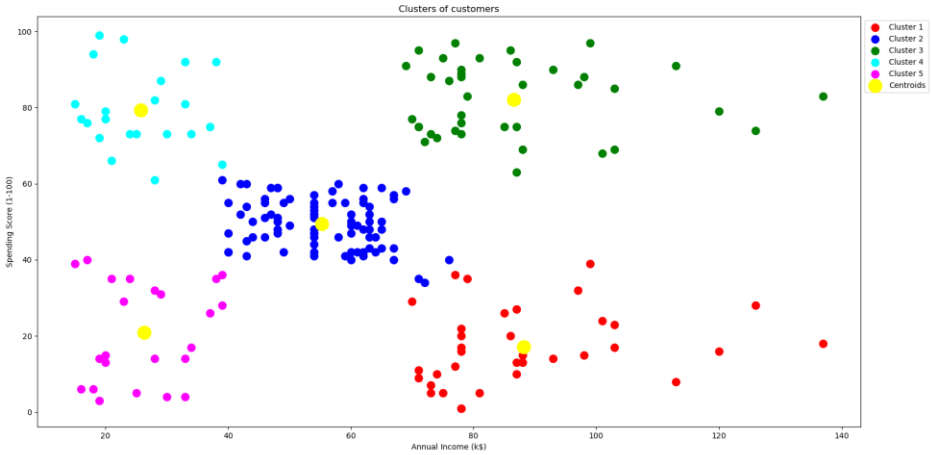
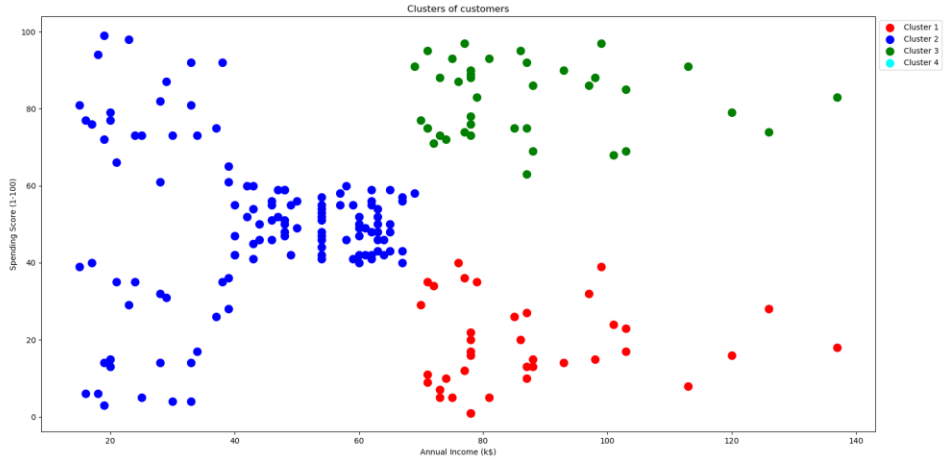


Figure 1
The Elbow Method
250000
200000
150000
100000
50000
1
2
3
4
5
6
7
8
9
Number of clusters


Clusters of customers
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Centroids
Spending Score (1-100)
Annual Income (k$)


Clusters of customers
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroids
Spending Score (1-100)
Annual Income (k$)



Trend stock prices – RNN:

No limitation in the type of data in neural network

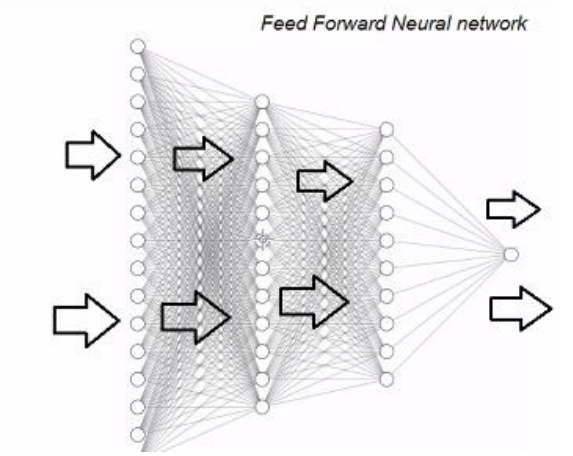
LSTM have different architecture. They have recursion inside them

Recursive functions – call function itself. Feed the input to it and process it.

Let's see what we have in recursion part:

Good when they don't depend on previous output

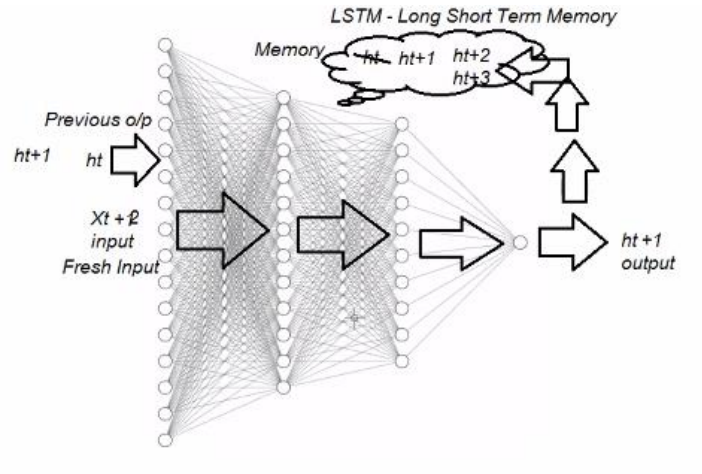
They don't have memory



Recursive NN– have reference of previous output

They will have memory to preserve it.

Preserve as long as you want!



TimeStep = 3 , lookback at 3 previous words

Predict next word by looking at previous 3 words

This
This is
This is my
This is my class
This is my class of
This is my clss of Data
This is my clss of Data science
This is my clss of Data science and
This is my clss of Data science and Machine
This is my clss of Data science and Machine Learning

Help you to preserve the previous output

Sequential processing

They have specialized architecture – Gateway architecture

Long time memory is the cell state.

What remains in the memory is decided by cell state

Cell state – consider it as conveyor belt. Put what you want in that belt

Forget gate – 0

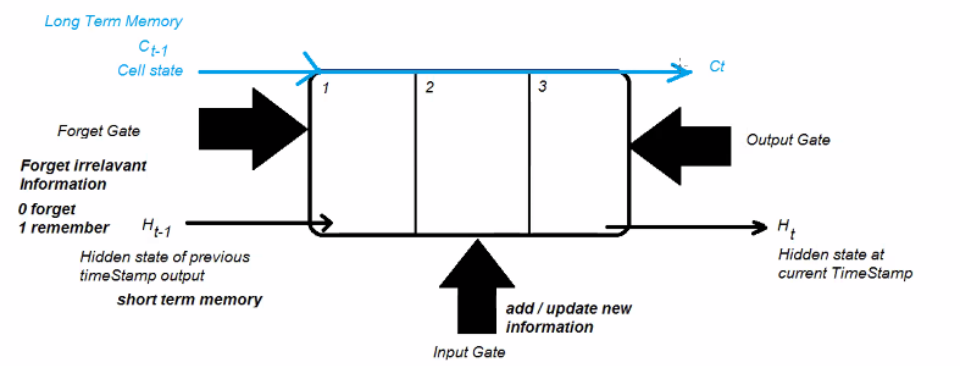
Remember gate – 1

Input gate – add the input data

Output gate – Put the output info in the belt

Long term memory and short-term memory is the immediate output.

To take references and make predictions



What is the use case of using the LSTM?

Stock price analysis:

	symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
0	AAPL	2015-05-2	132.045	132.26	130.05	130.34	4.6E+07	121.683	121.881	119.844	120.111	4.6E+07	0	1
1	AAPL	2015-05-2	131.78	131.95	131.1	131.86	3.1E+07	121.438	121.595	120.812	121.512	3.1E+07	0	1
2	AAPL	2015-05-2	130.28	131.45	129.9	131.23	5.1E+07	120.056	121.134	119.706	120.932	5.1E+07	0	1
3	AAPL	2015-06-0	130.535	131.39	130.05	131.2	3.2E+07	120.291	121.079	119.844	120.904	3.2E+07	0	1
4	AAPL	2015-06-0	129.96	130.655	129.32	129.86	3.4E+07	119.761	120.402	119.171	119.669	3.4E+07	0	1
5	AAPL	2015-06-0	130.12	130.94	129.9	130.66	3.1E+07	119.909	120.664	119.706	120.406	3.1E+07	0	1
6	AAPL	2015-06-0	129.36	130.58	128.91	129.58	3.8E+07	119.208	120.333	118.794	119.411	3.8E+07	0	1
7	AAPL	2015-06-0	128.65	129.69	128.36	129.5	3.6E+07	118.554	119.512	118.287	119.337	3.6E+07	0	1
8	AAPL	2015-06-0	127.8	129.21	126.83	128.9	5.3E+07	117.771	119.07	116.877	118.784	5.3E+07	0	1
9	AAPL	2015-06-0	127.42	128.08	125.62	126.7	5.6E+07	117.421	118.029	115.762	116.757	5.6E+07	0	1
10	AAPL	2015-06-1	128.88	129.34	127.85	127.92	3.9E+07	118.766	119.19	117.817	117.881	3.9E+07	0	1
11	AAPL	2015-06-1	128.59	130.18	128.475	129.18	3.5E+07	118.499	119.964	118.393	119.042	3.5E+07	0	1
12	AAPL	2015-06-1	127.17	128.33	127.11	128.185	3.7E+07	117.19	118.259	117.135	118.125	3.7E+07	0	1
13	AAPL	2015-06-1	126.92	127.24	125.71	126.1	4.4E+07	116.96	117.255	115.845	116.204	4.4E+07	0	1
14	AAPL	2015-06-1	127.6	127.85	126.37	127.03	3.1E+07	117.586	117.817	116.453	117.061	3.1E+07	0	1
15	AAPL	2015-06-1	127.3	127.88	126.74	127.72	3.3E+07	117.31	117.844	116.794	117.697	3.3E+07	0	1
16	AAPL	2015-06-1	127.88	128.31	127.22	127.23	3.5E+07	117.844	118.241	117.236	117.245	3.5E+07	0	1
17	AAPL	2015-06-1	126.6	127.82	126.4	127.71	5.5E+07	116.665	117.789	116.481	117.688	5.5E+07	0	1

Time step – consider

This will help to capture pattern from your data

Time step => Look back sequence

Predict next word looking at previous 3 words.

TimeStep => Look back sequence

Predict next word looking at previous 3 words TimeStep = previous 3 words == 3

Predict next word looking at previous 50 words TimeStep = previous 50 words == 50

Predict next days close price looking at previous 100 days close price of the stock
TimeStep = previous 100 days stock price == 100

If the t is the input data, then t+1 time will be the prediction.

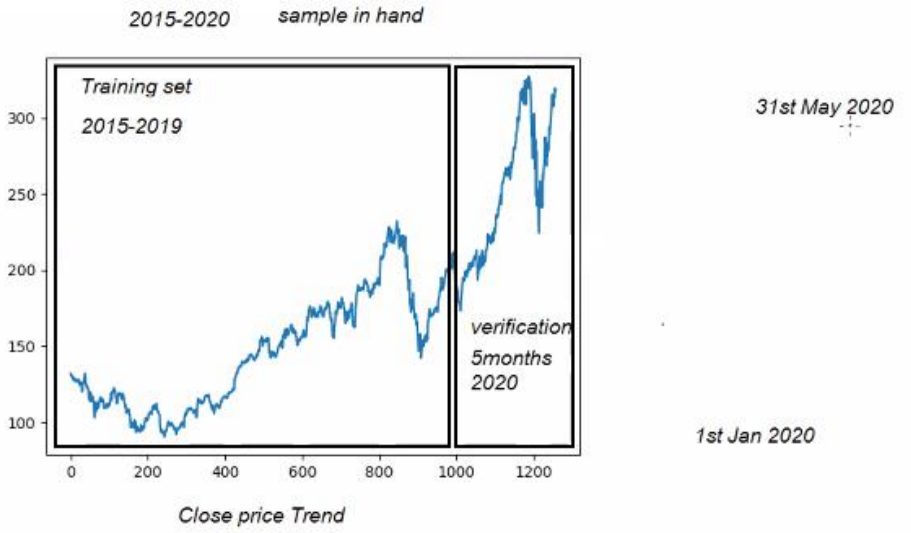
This is the training data.

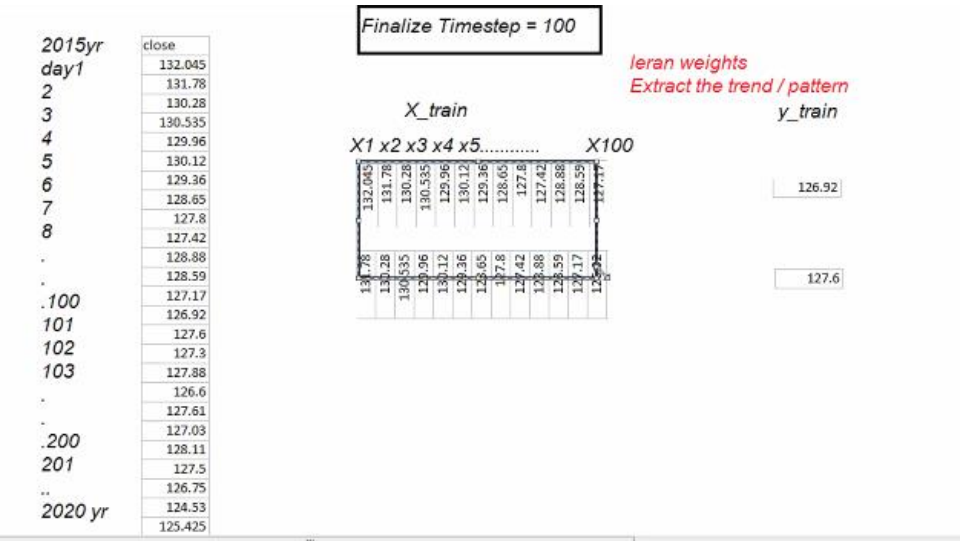
X matrix col = TimeStep

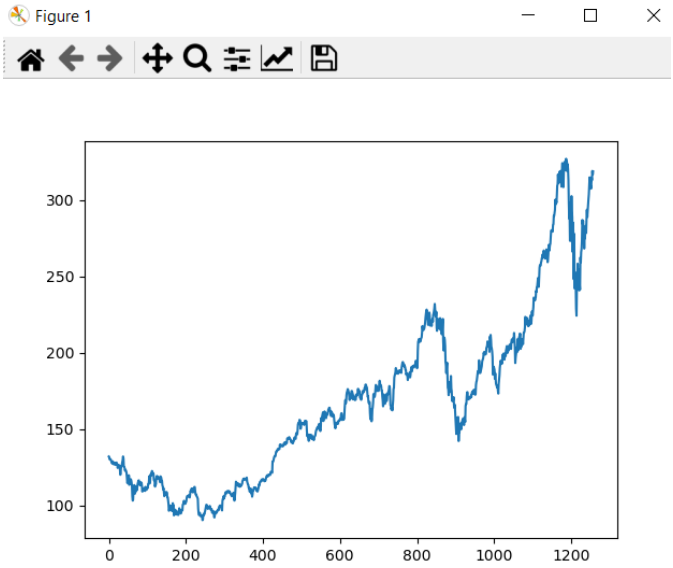
Input Data X_train	Output Data y_train
0 1 2 3 4 5 6 7 8.....99	100
1 2 3 4 5 6 7 8 9.....99 100	101
2 3 4 5 6 7 8 9.... 99 100 101	102

1.....100 12th March ✓ 13thMarch 14th March

1.....100 ✓ 12th March ✗ 13thMarch 14th March ✗ ✗







!pip list

import pandas

pandas.__version__

https://github.com/meghakarale/DataScience-Reference-Repository

https://drive.google.com/drive/folders/1XazjLsUmnmnhw5tbOnPjlMpcADmtIPRV?usp=sharing

https://github.com/meghakarale/DataScience-Reference-Repository