Day 3-

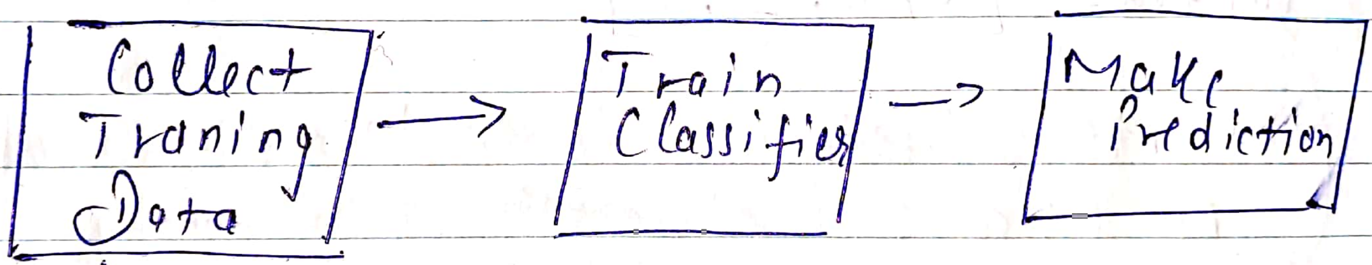
Image Classification

#(Classifier :) Let think classifier as a function, it takes some data as input and assigns a label to it as output.

for eg We have a picture of an apple and want to classify it is an Orange or an apple, the techinique to write a classifier automatically is called Supervised Learning.

To code, this we'll work with scikit-Learn.

Supervised Learning Recipe :)



Collect Traning Data → Train Classifier → Make Prediction

eg ↓ will take description of fruit as input and predict whither it is apple or orange based on features like its weight & texture

Step ①
Traning data!)

(The more data, we have more accurate be the classifier)

Weight	Texture	Label
150g	Bumpy	Orange
170g	Bumpy	Orange
140g	Smooth	Apple
130g	Smooth	Apple.

In Machine Learning these measurements are called features.

~~Step~~

<code>

```
1) import sklearn
2) features = [[140, 1], [130, 1], [150, 0], [170,0]]
3) labels = [0, 0, 1, 1]
```

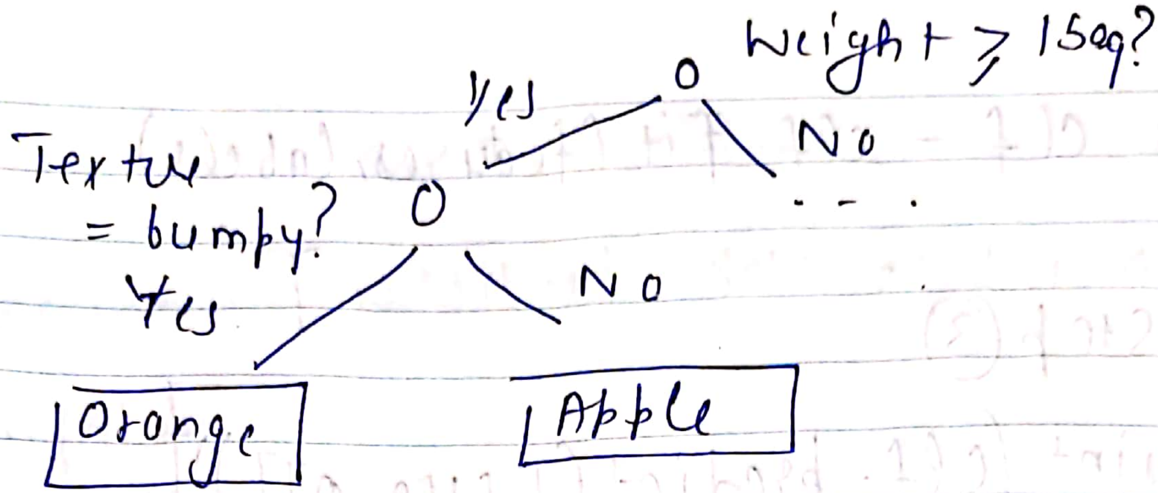
-> contains first two columns [Weight, Texture]
we used 1 for Smooth, 0 for Bumpy

-> Contains Last Column [Label]
(We used 1 -> orange , 0 -> Apple)

Step ② Train the Classifier,

— The type of classifier we start with is "Decision Tree".

Decision Tree



<code>

1) import tree

4) Clf = tree.DecisionTreeClassifier()

(To train, classifier, we need a learning alogrithim } as the procedure that Create them, It does that finding patterns in our training data.)
for eg, it noticed that oranges tend to weigh more, so it create a rule that heavier fruit is more likely to be orange.)

In scikit, the training algorithim is included in the classifier object and it's called $\boxed{\text{fit}}$ * → being a synomym for "find Patterns". in data

5) clf = clf. fit (features, labels)

[Now Lets Make prediction]

Step ③

6) print (clf. predict ([[150, 0]])) } input

[1] } output
(orange)

—x— —x— —x— —x— —

ML pipelines !)

A machine Learning pipe line is used to help automate machine learning workflows They operate by enabling a sequence of data to be transformed and correlated together in a model that can be tested and evaluated to achieve an outcome, whether Positive or negative.

Code a Basic Pipeline for supervised Learning

① Building a spam classifier !)

Step 1 Dataset collection

eg !

Email	Label
Click here to claim	Spam
What's New?	Not Spam
Hangout Later?	Not Spam
You won $100000	Spam
- - -	- - -

→ our approach is to partition our dataset into Two parts :

Train → used to train our model

Test → to see how accurate it is on new data

—x—x—x—x—x

< code >

```
# import a dataset
from sklearn import datasets
iris = datasets.load_iris()

X = iris.data
y = iris.target
```

features   Label

$$\boxed{f(x) = y}$$

(after we, import dataset, first thing we want to do is parti~~ta~~tion it into Train and Test.)

7) from sklearn.cross_validation import
train_test_split

features and Labels for training set

8) X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.5)

→ features and Labels for testing set

test_size = 0.5 means half of data used for testing eg (150 examp in iris 75 will be in Train and 75 will be in Test)

Now we create our classifier :) → (Classifier)

9) from sklearn import tree
10) my_classifier = tree.DecisionTreeClassifier()
11)
12) my_classfier.fit(X_train, Y_train)

train classifier using training data

13) predictions = my_classifier.predict(X_test)
14) print predictions

(used to test the data)

(In order to check how accurate our classifier was on testing set.)

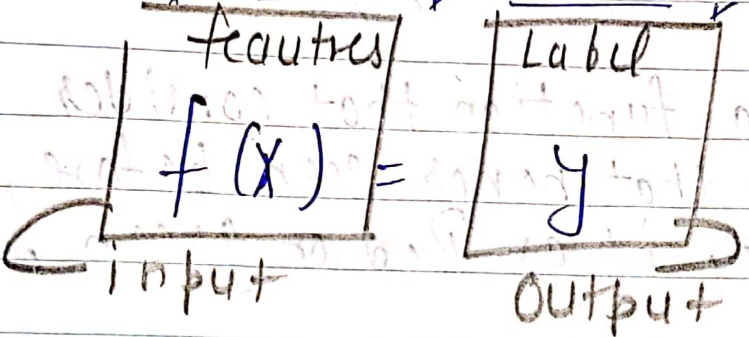
(To calculate, our accuracy, we can compare the predicted labels to true labels)

```
15) from sklearn.metrics import accuracy_score
19) print accuracy_score(y_test, predictions)
```

the line (9, 10) → By Replacing the Decision tree classifier, with another ie. (with kNeighorsClassifier) :

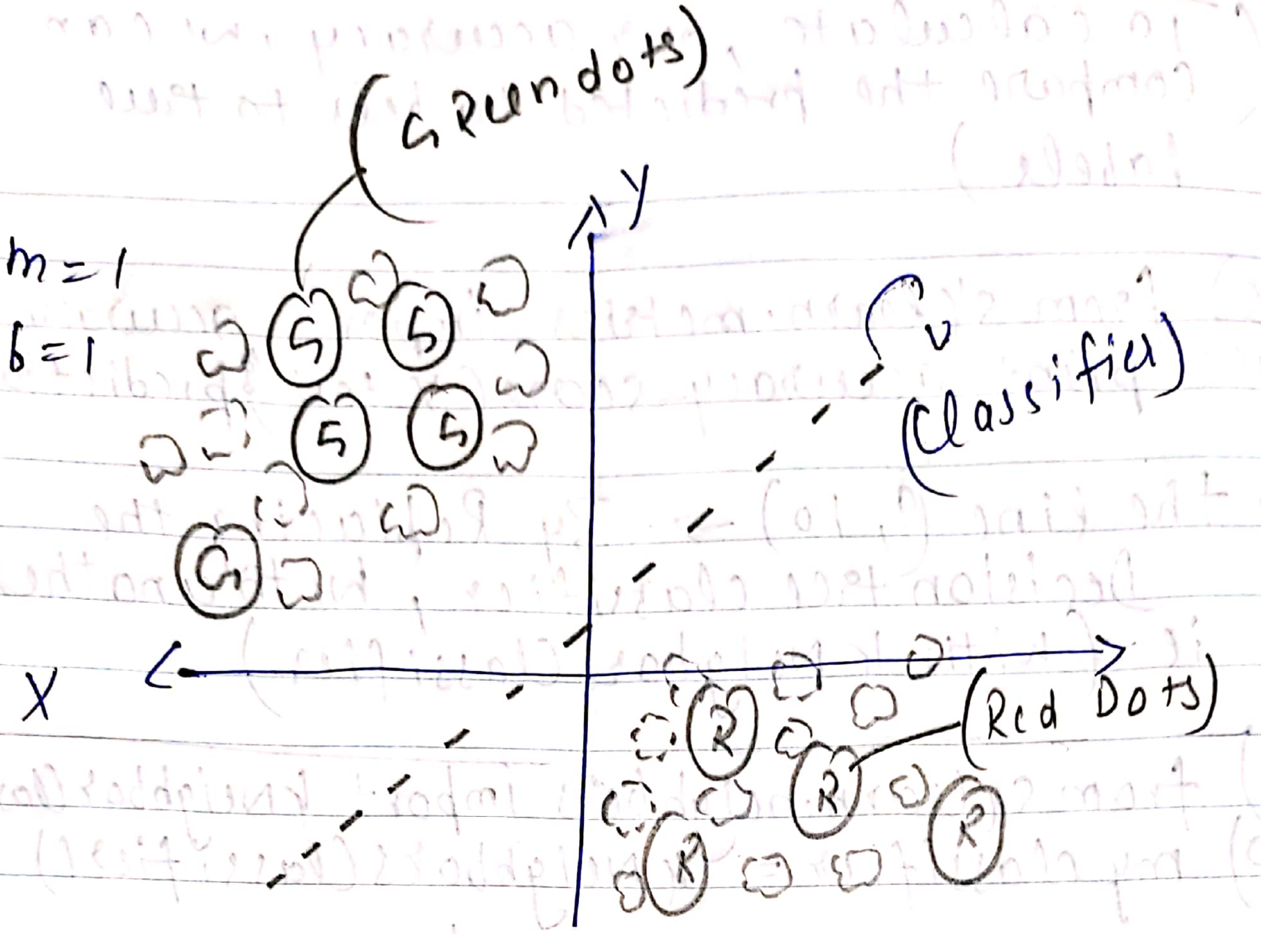
```
9) from sklearn.neighbors import kNeighborClassifier
10) my classifier = kNeighborsClassifier()
```

↳ (Classifier)



```
def classify (features):
    # do some Logic
return label
```

As we know, in supervised Learning, we don't want to write this ourselfes. we want an alogrithim to learn it from training data



To Distinguish b/w Red & Green, we just use two parameter X & Y coordinate of a dot.

We want a function that consider a new dot that never seen before, and classify it as Red or green,

Let, testing example the Dotted dots on Green side represents (Light Green) and on Red side → Light Red.

were not in our traning data, Classifier has never seen them before

Well, imagine if we draw a line, then we can say dots to left of line are green and dots to Right of line are Red and line serves as "Classifier"

(Way to Learn the line)

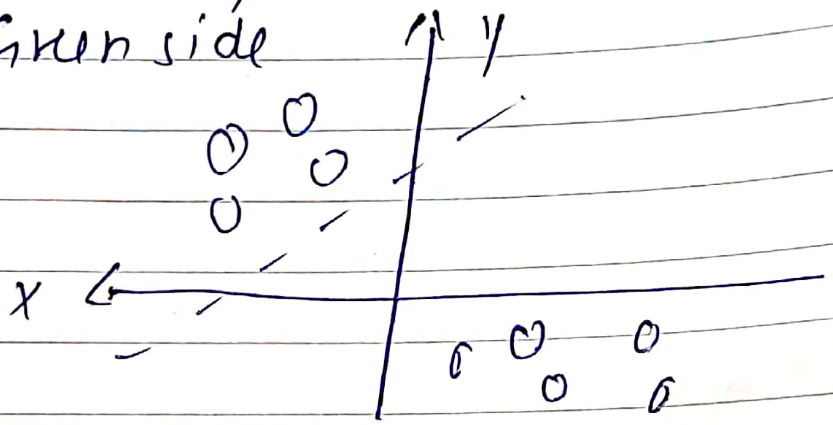
Let assume two parmiter m & b

m=1, b=1

↳ Changes to m=1, b=5 line shift towards Green side

m = 1

b = 5



Let Start with a random line, and use it to classify first example, it it gets right, we don't change our line, so, we move on to next one. but if it gets wrong, we can slighty adjust parameters of our model to make it more accurate.

Q ~~for~~ another way → (Tensor flow/ Play ground) (example of Neural Network)