Machine learning Fundamentals

12 January 2023 20:53

Making the machine (computer) do those kinds of tasks in which everytime a new situation (new set of data) arises. eg. Identifying object in an image.

What is Machine Learning ?

Machine learning is a field of science associated with algorithms which allow software applications to continuously learn from data and improve in predicting specific outcomes, without being explicitly programmed to do so. Hence, the main difference between an ML application and a software application is that the former one's self-learning in a sense which a normal software is not. Rather, a normal software is a simple set of logical instructions involved to follow based on specific conditions, and there is no self-learning involved.

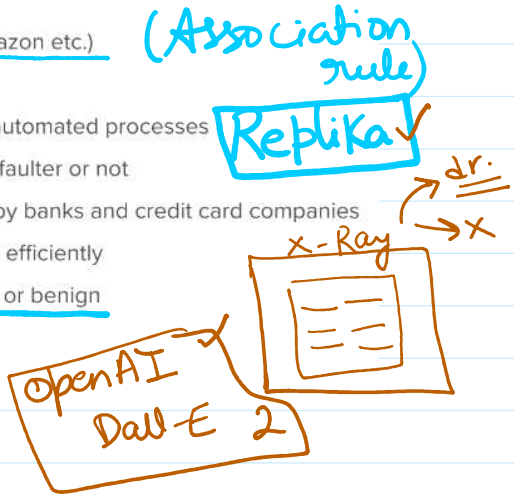
For example –
A bank application which takes a user's Id and Password as input and shows the user his bank account balance, loans, offers etc is an example of a software application. This is because in this case application has a fixed set of rules or logic such that if the credentials are correct, redirect the user to his details screen.

On the contrary, a credit risk model used by bank to determine whether a loan application should be processed or not, is an example of Machine Learning application. The reason being the rules are not defined manually by a developer, instead the ML model develops its own rules based on the historical or sample data.

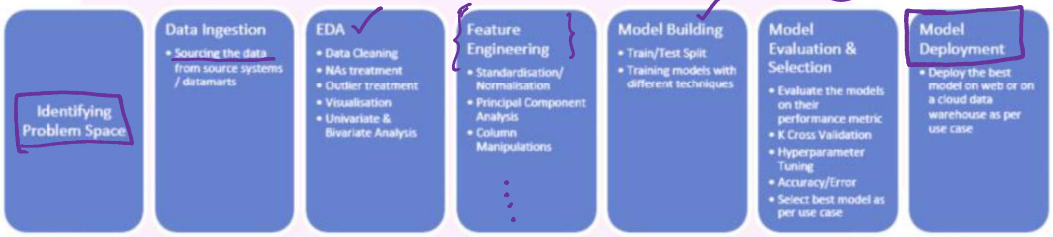
Seeing our EDA case study 1 - remember the target variable which took the values 0 or 1 for non-defaulters or defaulters. If we were able to predict this using the other data present in the dataset we would have built a ML model.

ML & AI applications:

- Product recommendation systems for E-commerce websites (Flipkart, Amazon etc.)
- Media Recommendation systems (Netflix, Hotstar)
- Chatbots (Voice/Text) – Responding or resolving customer problems via automated processes
- Credit Risk Applications – Predicting whether a loan applicant can be a defaulter or not
- Fraud detection – To predict whether a transaction is a fraud or not, used by banks and credit card companies
- Inventory Optimization – Predicting sales of products to manage inventory efficiently
- Healthcare Applications – Predicting whether a Cancer tumor is malignant or benign
- Self-driving cars



General pipeline of a ML project:



Dall-E 2

published for mass use.

1. Identifying Problem Space –

Identifying and understanding a problem is the first and very crucial step of an ML project. Without a well defined and proven problem, any project is bound to fail or deliver bad results. But often, this part is not overlooked upon and teams start working on a half baked problem.

For example :

vague

Problem : Prepare a model for better personalized marketing strategy for a shoe retail company.

Above is a very general problem statement and needs more work to get to more precise on outcome.

to the point

Better Problem Statement : Prepare a model to predict sales of customers in next 1 month so that marketing spend is allocated efficiently.

This problem clearly states what the stakeholder wants, and why does he want it, hence it can be a good starting point for the workflow.

2. Data Ingestion :

Once the problem is well defined and crafted, the next step is to analyze what kind of data can be procured, and from where. In many cases, it can be in house company data that can be used to solve a company's problem, while in other instances there might be third party data providers which may provide the data on chargeable contract. Corelogic, Epsilon , IDM are some of the known data providers in US region.

Getting relevant data for the model training.

Sourcing your data → In house company data

→ External data. (Secondary Data)

3. EDA :

EDA (Exploratory data analysis) has already been explained as practiced as part of previous week. While EDA plays a huge role for data analysis and finding insights from the data, it is also a very important step in a ML workflow.

Involves data cleaning, understanding the data, making

visualisations and taking a general overview to get familiar with what we will be working with.

4. Feature Engineering

It involves concepts like Standardisation/Normalisation of data, Principal Component Analysis etc. These concepts will be covered later in this or subsequent sessions in detail. On a high level, feature engineering is required to make the job easier for algorithm and to reduce computational power required.

It also involves transformation of variables as per use case so that variables can be made usable for a model. For example, for a real-estate data, the year built of a house (1956, 1978 etc.) can hardly be used in any model. But the same can be converted to house age by formula : (Current Year – Year Built), which can be a very useful factor for a given problem.

eg.
birth_days → # days customer has lived.

eg. When was house built → 1976 not very useful to me

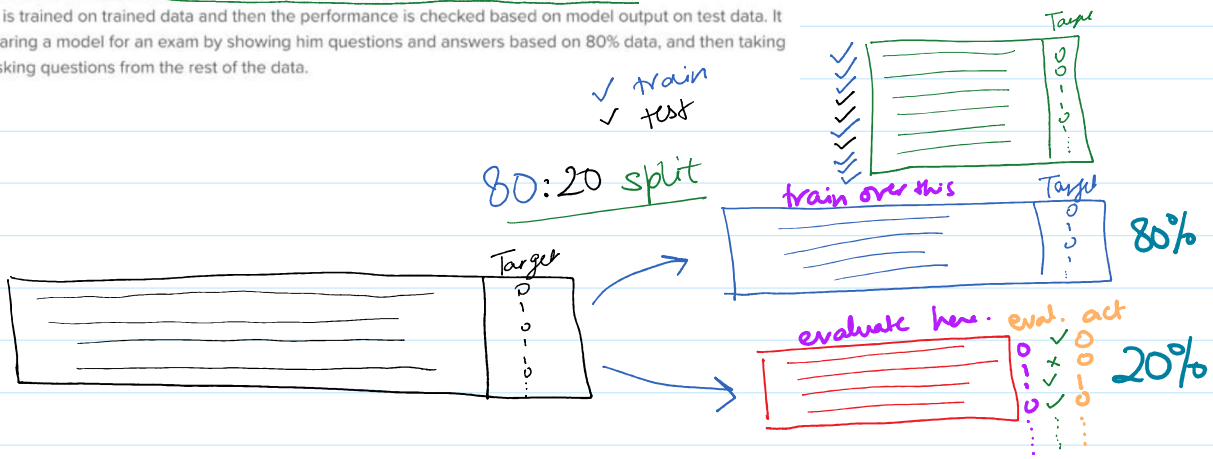
transformed variable

but 2023 - 1976 = 47 tells me the age of the property which can be used in it's evaluation

5. Model Building

Once the data is in a polished format for a ML algorithm, multiple models are trained based on relevant algorithms for that problem. ⇉ multiple possible working models which we will test & select the best one.

The data is divided into 2 sections : Train data and Test data. (Let's say 80%/20% split is done)
The model is trained on trained data and then the performance is checked based on model output on test data. It is like preparing a model for an exam by showing him questions and answers based on 80% data, and then taking a test by asking questions from the rest of the data.



6. Model Evaluation & Selection

Finally, out of all the trained models, 1 model is selected based on the best performance.

Simply we look at the accuracy of prediction provided by each of the models to see which one performed the best.

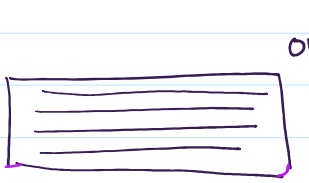
7. Model Deployment –

The selected model is deployed either on a web based UI or at backend on a cloud data warehouse.

For example, in healthcare apps, chatbots etc., the model may be deployed as a UI application to be used by doctors directly.

Applying / Publishing model for mass use / Production

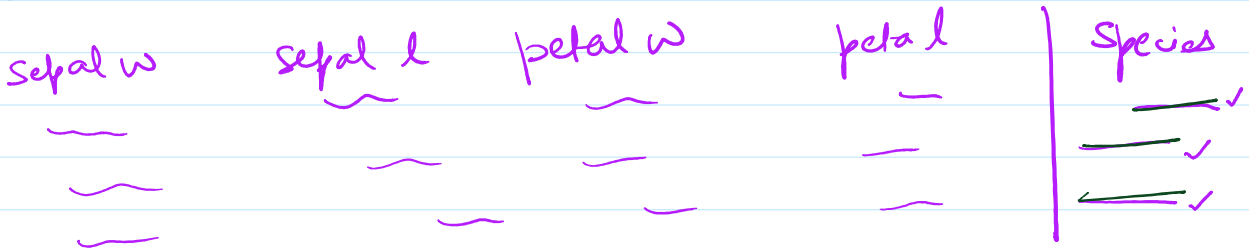
Types of ML models:
1. Supervised Learning models
2. Unsupervised learning models

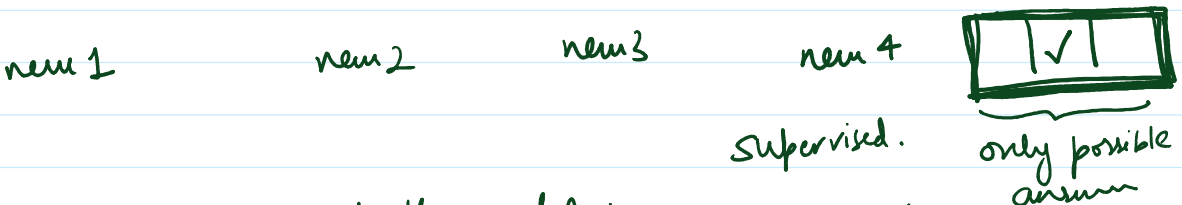


clearly defined

Supervised Learning:
Where the data provided to the model is well labeled i.e. the outputs are clearly defined is called supervised learning.
E.g. Regression/Classification models (most popular and most used)

Regression/Classification



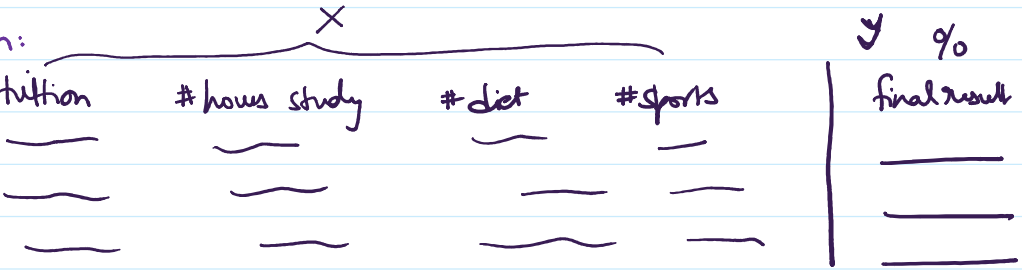


Supervised. only possible answers

which were shown to the model in the training phase

eg. of classification →

eg. of regression:



When predicting specific values we use a regression.

Given a set of X variables we need to predict y

Popular models:
1. Linear regression
2. SVR — support vector regression.
3. Decision trees

Regression

Popular models:

1. Linear regression
2. SVR
3. Decision trees
4. Random forests

support vector regression.

Regression

classification

1. Classification

The target variable is Categorical.

Examples –

- Predicting pet animal type based on animal attributes
- Predicting whether an applicant is a loan defaulter or not
- Predicting whether a customer is loyal customer or not
- Spam detection applications
- Image Classification
- Fraud Detection

2. Regression

The target variable is Quantitative.

Examples-

- House Price Prediction
- Employee bonus recommendation
- Sales forecast

Like cat vs dog image classifier

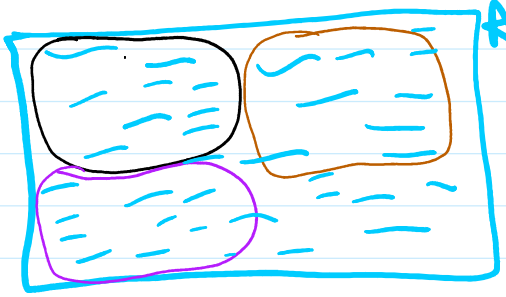
cat dog

Classification Problem

Unsupervised Learning:

Where the data provided to the model not labeled i.e. the model needs to identify patterns in the data and form groups on its own.

E.g. Clustering/Association models



Amazon Product List

millions of products.

If model forms these clusters on it's own by seeing some common features/properties among these bundels it's called unsupervised learning.

Popular models:

1. K means clustering
2. Association rule learnings

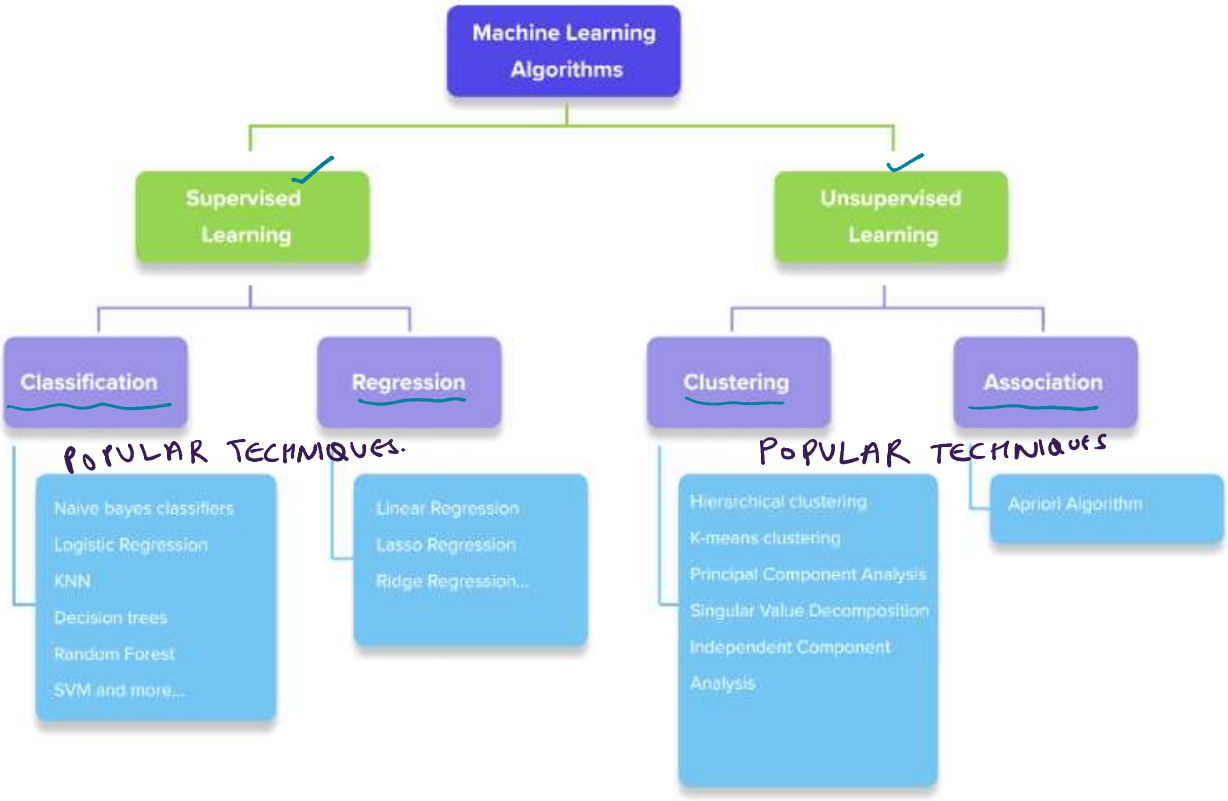
Unsupervised Learning techniques broadly fall into 2 categories –

1. Clustering Techniques – Where the objective is to group the data points into N number of clusters based on similarities and patterns among data

Example – Customer Segmentation into groups based on their spend pattern and demographic factors

2. Association Techniques – Where the task is to find specific rules in the data, for example – Those who buy X also tend to buy Y

Like customers who bought this also bought → association rules



Summary

Supervised	Unsupervised
Train data contains labelled target variable for the model as a reference to train upon	No labelled field
Complexity is usually lower than Unsupervised techniques	High complexity
Main goal is to predict an output	Main goal is to find hidden patterns and associations in data
The results are more accurate in Supervised techniques since there are well defined metrics to measure performance and to improve upon.	The results are not as impactful or accurate as in Supervised models.
It accepts feedback and improves the model accordingly. In a feedback mechanism, the predicted output can be reused to train the model again for newer versions of the model	There is no feedback involved

Splitting your data into Train/Test datasets
(Ratios)

80-20 split. general value

After sourcing your data you split the data (supervised) into training and testing data set. You use the

training (75-95%) dataset to build/train the model and use the testing (5% - 25%) dataset as new values to see how well model has been trained.

What is Feature Engineering

Feature engineering is a machine learning technique that leverages data to create new variables that aren't in the training set. It can produce new features for both supervised and unsupervised learning, with the goal of **simplifying and speeding up data transformations** while also **enhancing model accuracy**. Feature engineering is required when working with machine learning models. Regardless of the data or architecture, a terrible feature will have a direct impact on your model.

- **Feature Creation:** Creating features involves creating new variables which will be most helpful for our model. This can be adding or removing some features. As we saw above, the cost per sq. ft column was a feature creation.
- **Transformations:** Feature transformation is simply a function that transforms features from one representation to another. The goal here is to plot and visualise data, if something is not adding up with the new features we can reduce the number of features used, speed up training, or increase the accuracy of a certain model.
- **Feature Extraction:** Feature extraction is the process of extracting features from a data set to identify useful information. Without distorting the original relationships or significant information, this compresses the amount of data into manageable quantities for algorithms to process.
- **Exploratory Data Analysis :** Exploratory data analysis (EDA) is a powerful and simple tool that can be used to improve your understanding of your data, by exploring its properties. The technique is often applied when the goal is to create new hypotheses or find patterns in the data. It's often used on large amounts of qualitative or quantitative data that haven't been analyzed before.

Techniques:

Now that we have discussed the goals of Feature engineering, let's see what all techniques it actually entails –

1. Data Cleaning

a. Null Value Treatment : Replacing Nulls values with appropriate imputations or discarding data with Nulls)
b. Outlier Treatment : Replacing Outliers (Quantitative feature data points which are too far away from mean) with appropriate imputations. Such values can be erroneous and hence should be treated before being given as input to a model. For example, a data point of 150 for "Age" must be an error and hence should be addressed.
c. Treating incorrect labels (Merge "Male", "M", "Men" all to one category -> "M")
d. Date Values Treatment (Extracting Week, month, day from complete date)
e. Feature Extraction (Extracting House Age from Year Built)

2. EDA

getting to know the data to be able to make good judgement calls.

3. One-hot encoding

Computerised algorithms don't understand categorical data. Hence, it is a common practice to encode Categorical variable to binary fields for the model.

A categorical variable having N labels (distinct values) can be represented by N-1 binary columns (having 0s and 1s). This is because if all the N-1 fields are 0, then that is enough information for the model to deduce that leftover field must be 1.

Since Models better understand 0 & 1 as absence and presence instead of categories like low, medium, high, we use a concept called One Hot Encoding.

Starbucks sizes → Short, Tall, Grande, Venti

Size	Short	Tall	Grande
Tall	0	1	0
Grande	0	0	1
Short	1	0	0
Short	1	0	0
Venti	0	0	0
Short	1	0	0
Tall	0	1	0

One Hot Encoded

3 columns to show 4 varieties.

PROS:
- Gives it in a absence Presence format which can be directly used in model.
- Don't need to decide weightages (1234 or 1568)

CONS:
- creates too many extra columns
- if weights are needed you can't specify.

4. Feature scaling

Scaling refers to transforming the numerical fields onto a common scale.

A dataset might have many different type of fields which might have different units, measured in different ways, have very different ranges. It might be unfair to train a model on such data since there is no benchmark for the model to prioritise the fields for training. There may be a scenario that a field gets high weightage or importance in predicting an outcome just because it had extremely high values as compared to other fields, but in reality that field might not have impact on the target, or that the high values were resulting due to choice of units.

Scaling is required so that we can do an apple to apple comparison of various fields.

Every data has a mean & std. deviation.

Discuss ranges and uses

$x - \bar{X}$ → centre around 0

Standardisation

The formula for Standardisation is given by –

$$y = \frac{x(i) - \bar{X})}{\sigma}$$

Where
y = Scaled value
x(i) = Unscaled value
$\bar{X}$ = mean value of column
σ = Standard deviation of the column

resulting y has mean = 0 & std. dev = 1

range = $-\infty$ to ∞ (sqt bound at $(-3, +3)$)

Normalisation

It is also known as MinMax scaling.
This type of scaling will transform a numerical value to a range between 0 to 1.

The formula for this type of scaling is given by –

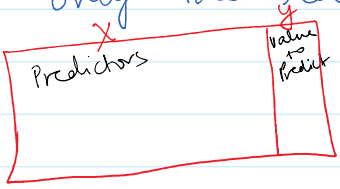
$$y = \frac{x(i) - min}{max - min}$$

y lies b/w 0 & 1 locked bound.

Where
y = Scaled value
x(i) = Unscaled value
min = minimum value of the column
max = maximum value of the column

5. Feature Selection - dimension reduction

Since there can be multiple features (columns in X dataset) but not all of them are as important in the prediction of y variable. Thus Feature selection techniques are used to find only the relevant features.



housing dataset

y → variable to be predicted (sale price) (dependent / target)

X → predictors (features used for prediction) (independent / predictor)
(location, nearby, sq. feet ....)

Recursive Feature Elimination (RFE)

Discuss RFE

Suppose a dataset has 101 columns, 100 → X
1 → y

Start w/ all 100 variables to predict 'y' my model will be very complex and it'll take more time to train

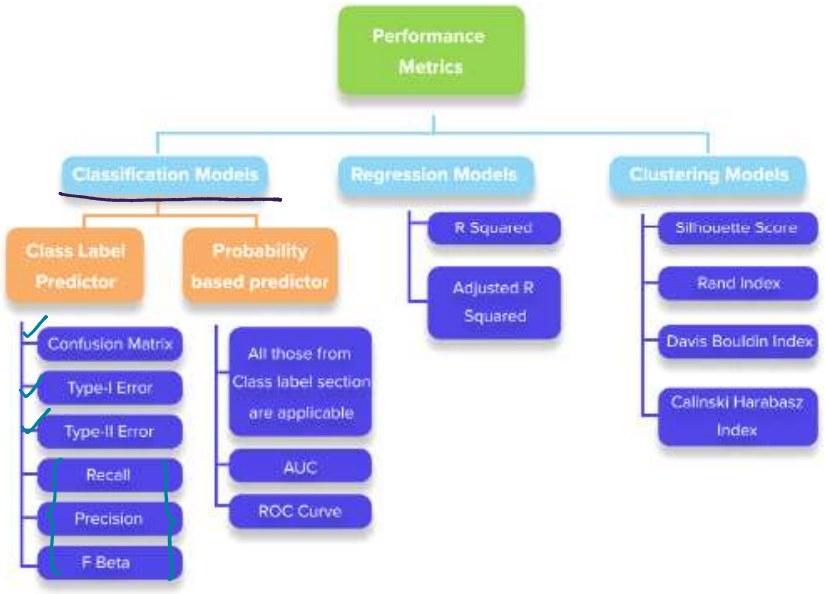
On the other hand If I only take 5 variables time consumed in model training is less but accuracy is hampered.

So RFE is a sweet spot b/w 2. It starts w/ all variables & keeps removing those variables which are not as powerful in predicting 'y' and finally delivers those variables which are most essential.

Performance Metrics

When we've made multiple models, we need to select the one which predicts the best.

So we compare these models on various metrics of Performance.



Confusion matrix

actual results

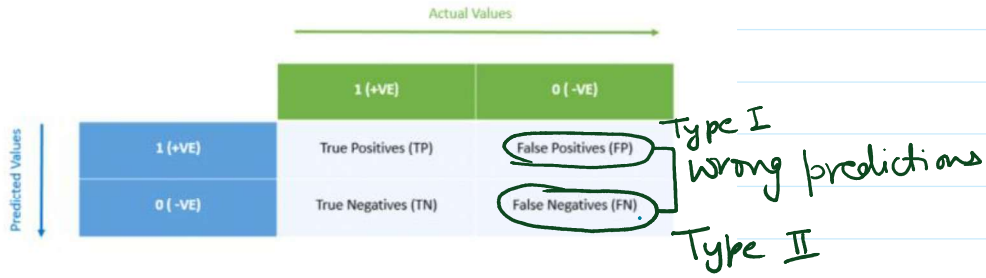
+	−
910	90

test results

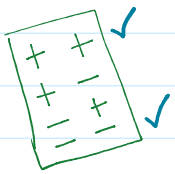
1000 RT-PCR tests were conducted

+	900
−	100

there was some error.



In Python output of a confusion matrix we get in this format



TP (a)	FP (b) Type I
Type I FN (c)	TN (d)

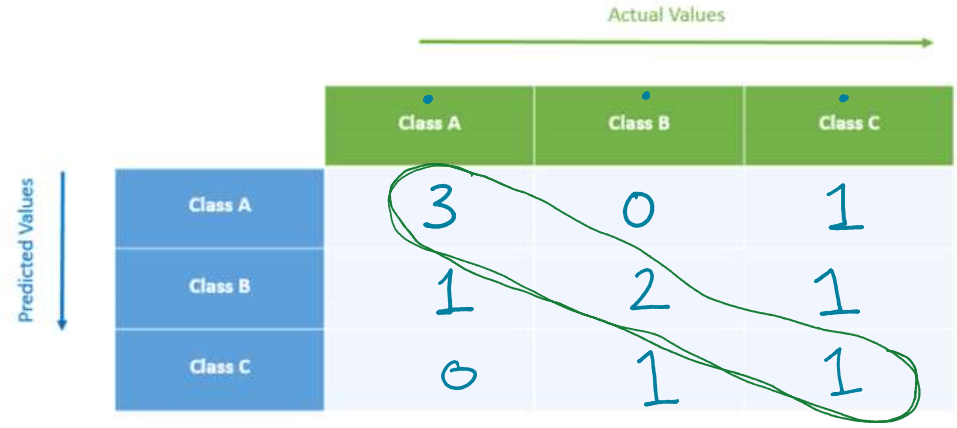
Accuracy = $\frac{a+d}{a+b+c+d}$

eg. 3 classes → A, B, C

CM for multi class predictions

eg. 3 species → A, B, C

actual	Predicted
A	A
B	C
A	A
C	B
C	C
A	B
B	B
B	B
C	A
A	A



Diagonal Always represents TRUE PREDICTIONS and all the rest are errors.

- **Accuracy**

Accuracy is simply the % of correct predictions. Mathematically, for a binary classification model, it can be given as -

Accuracy = (TP+TN) / (TP+TN+FP+FN)

In simpler words –

Accuracy = (Number of correct predictions) / (Number of total predictions)

Disadvantage :

Accuracy can only be a good indicator for a balanced dataset. Balanced datasets have fairly equal distribution of 0s and 1s as Targets in training data for the model to be unbiased.
For example if training data had 1000 rows, and out of 1000, 900 belonged were labelled as 0s, then the trained model will get biased because it was better trained to predict 0s, as compared to 1s.

Discuss over fitting

- **Type 1 & Type 2 Errors**

- **Type 1 Error**

When the Actual Negative class (0) is incorrectly predicted , this is known as Type-I error.
Essentially, **False Positives** are Type-I errors.

Examples :

- Predicting a Student to be eligible (1) for admission in Harvard Business School when in reality he was not eligible (0) due to poor grades.
- Predicting a mail as SPAM (1) when it is not a SPAM (0).

- **Type 2 Error**

When the Actual Positive class (1) is incorrectly predicted , this is known as Type-II error.
Essentially, **False Negatives** are Type-II errors.

Examples :

- Predicting a Student to be NOT eligible (0) for admission in Harvard Business School when in reality he was the topper of the country and was eligible (1)
- Predicting a cancer tumor as benign (0) when in real it was deadly (1).

Move to python nb for seeing commands