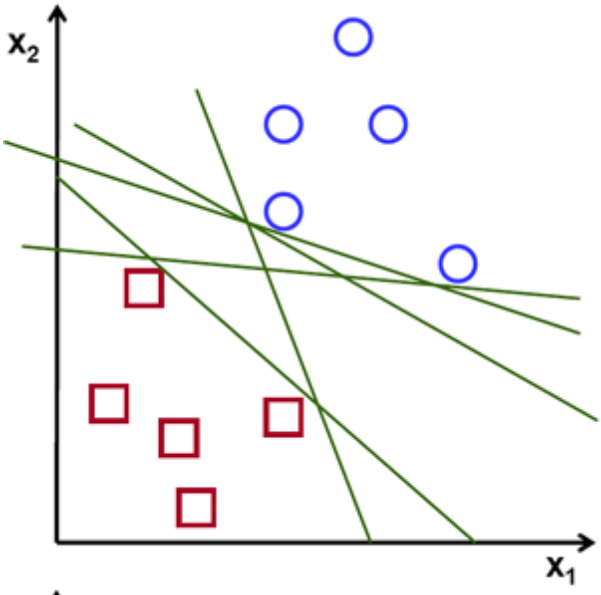
Supervised Learning: Classification

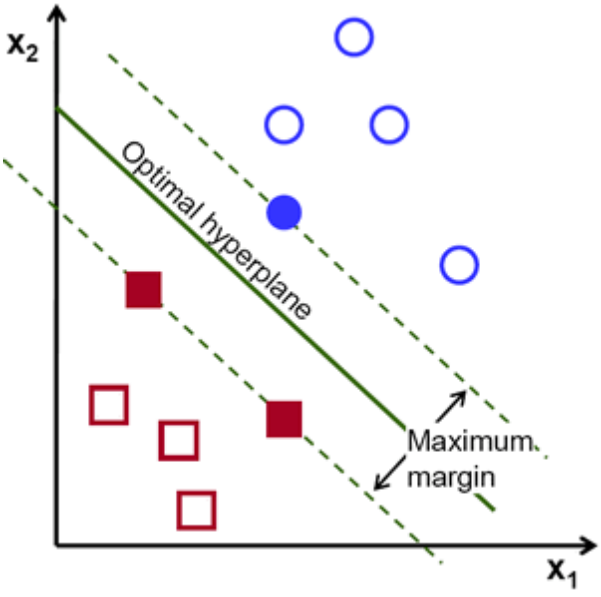
Saturday, December 17, 2022 2:02 PM

Support Vector Machine -

- SVM is used to classify the data points in distinct classes.
- It is a supervised learning method used for both regression and classification.
- SVM is used to find a Hyperplane or Decision Boundary which can separate classes of data points.
- There can multiple hyperplanes possible but we have to select the hyperplane with max margin.

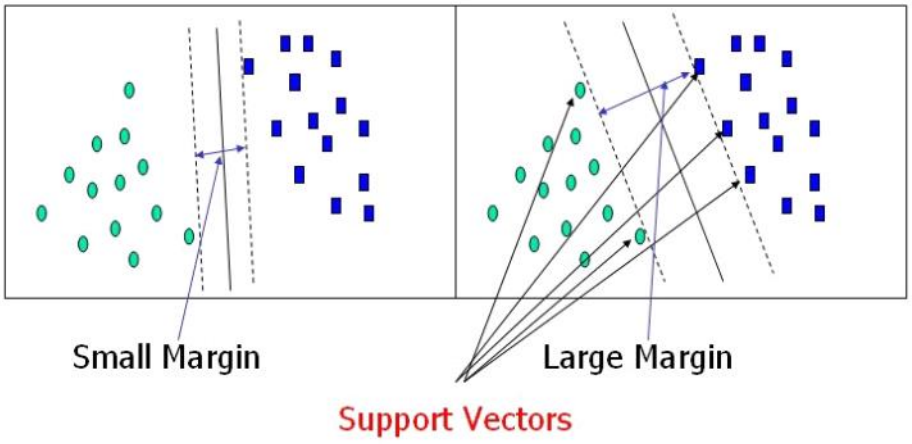


Possible Hyperplanes

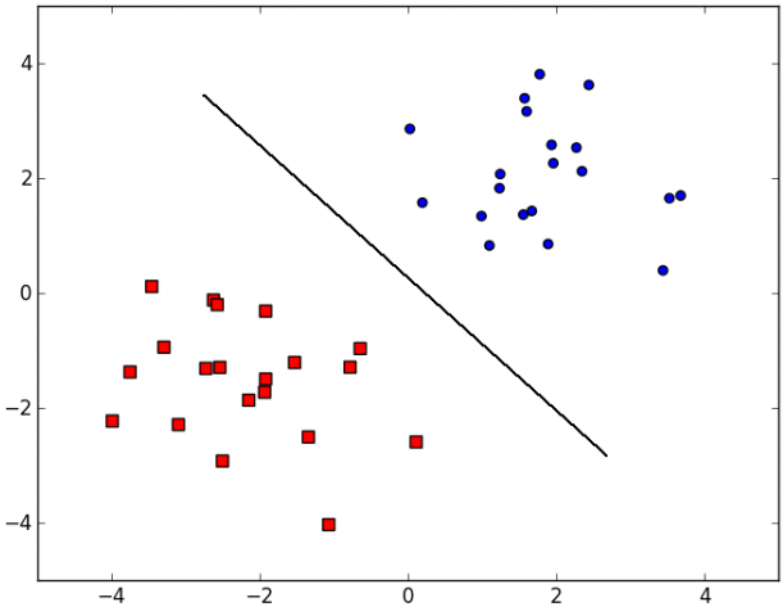


Maximal Margin Hyperplane

- The support vectors are the data points from the classes which are nearest to the Hyperplane.
- The Marginal lines are passing through these support vectors
- Support vectors are very important to decide hyperplane.

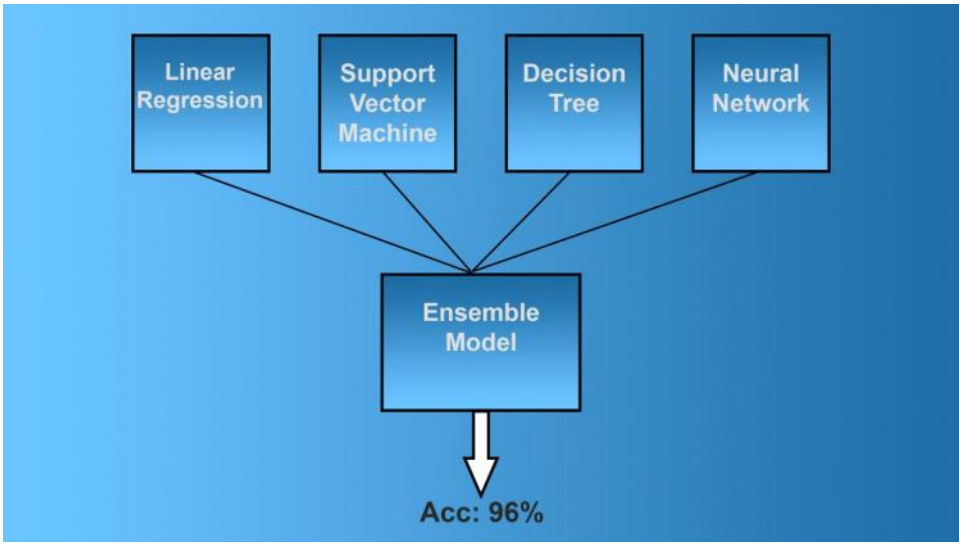


- Used to improve the performance of the ML model so that it can classify data perfectly.
- Some data points are linearly separable.
- For that we use Linear SVM



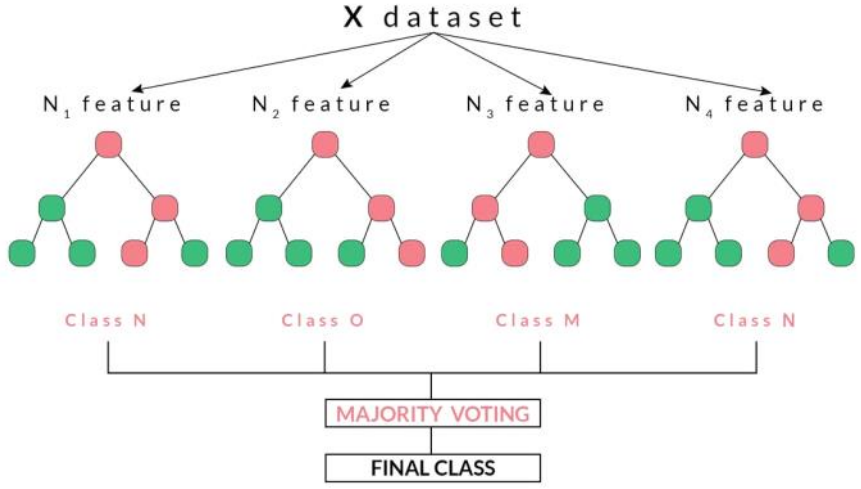
Ensemble Learning -

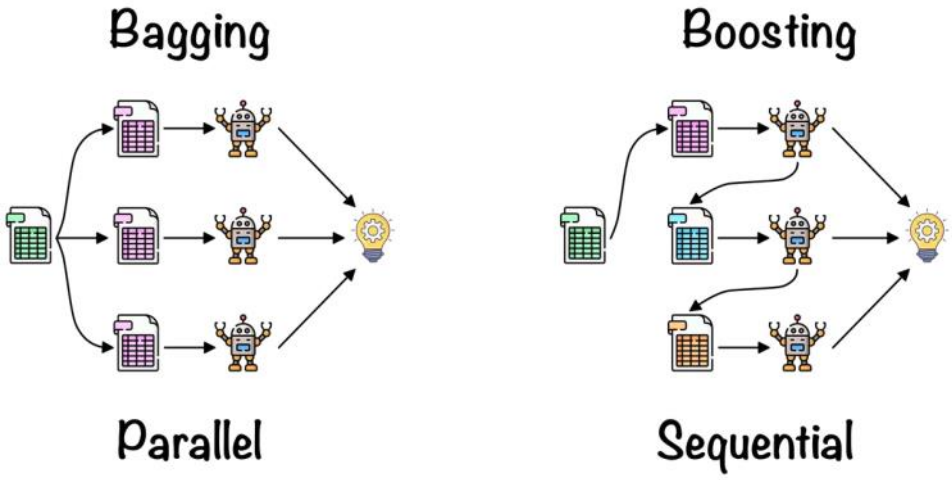
- Ensemble learning is a ML model that combines outcomes of multiple models and generates more accurate outcome.
- It combines predictions of multiple models and generates more accurate prediction.
- The multiple models are called as Base Learners
- They can use different algorithms to generate the outcome and this creates heterogeniuos ensemble
- If they are using same algo then provide them with different training dataset
- The outcome of this base learners is combined into single model called as Ensemble model



Random Forest -

- It is a supervised learning algo
- It works on the principle of ensemble learning
- It uses decision trees to generate outcome
- We have given with a dataset with N attributes
- We will select some random K records of data and then put them into new dataset
- This is called as sampling or Bagging or Bootstrapping
- From this sample data we generate a Decision tree
- While generating tree at each stage of node selection, we choose random subset of attributes
- From this subset we select a node
- Similarly we again select random K records from dataset and generate new decision trees
- Now we have multiple decision trees
- Now, for a given input we pass this input to each of the decision tree and get there outcomes
- By majority voting we decide to which class the given input should be assigned.





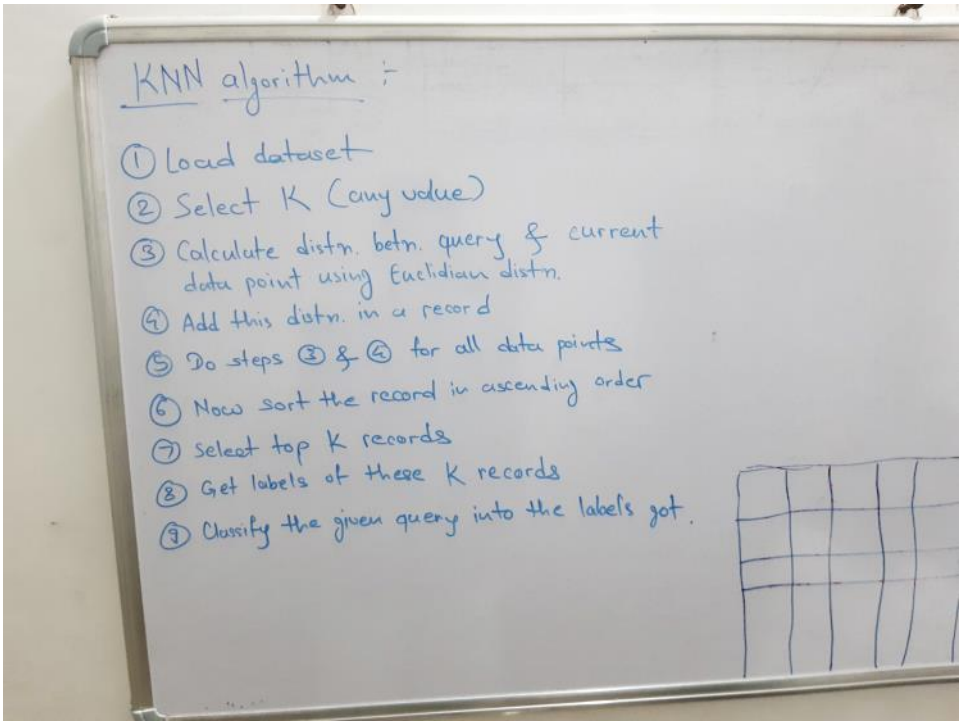
Random Forest works on Parallel Bagging.

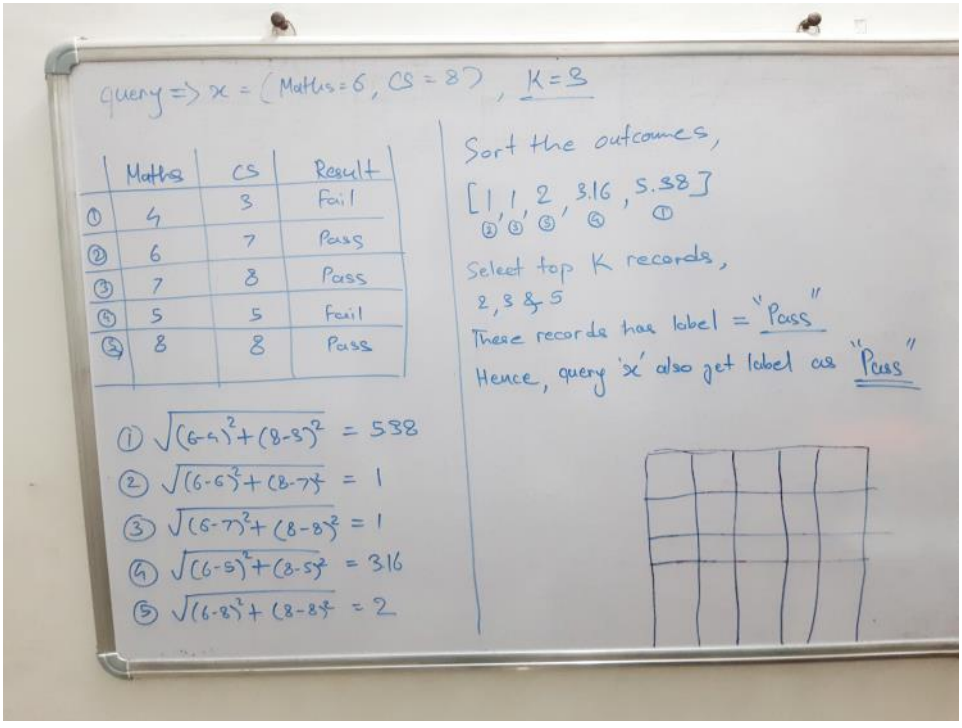
Decision tree vs Random Forest -

Decision Tree	Random Forest
A decision tree is a tree-like model of decisions along with possible outcomes in a diagram.	A classification algorithm consisting of many decision trees combined to get a more accurate result as compared to a single tree.
There is always a scope for overfitting, caused due to the presence of variance.	Random forest algorithm avoids and prevents overfitting by using multiple trees.
Decision trees require low computation, thus reducing time to implement and carrying low accuracy.	This consumes more computation. The process of generation and analyzing is time-consuming.
It is easy to visualize. The only task is to fit the decision tree model.	This has complex visualization as it determines the pattern behind the data.

KNN -

- KNN is a supervised learning algo used for classification or regression
- It acts on the principle that similar things exists in close proximity or similar things are close to each other



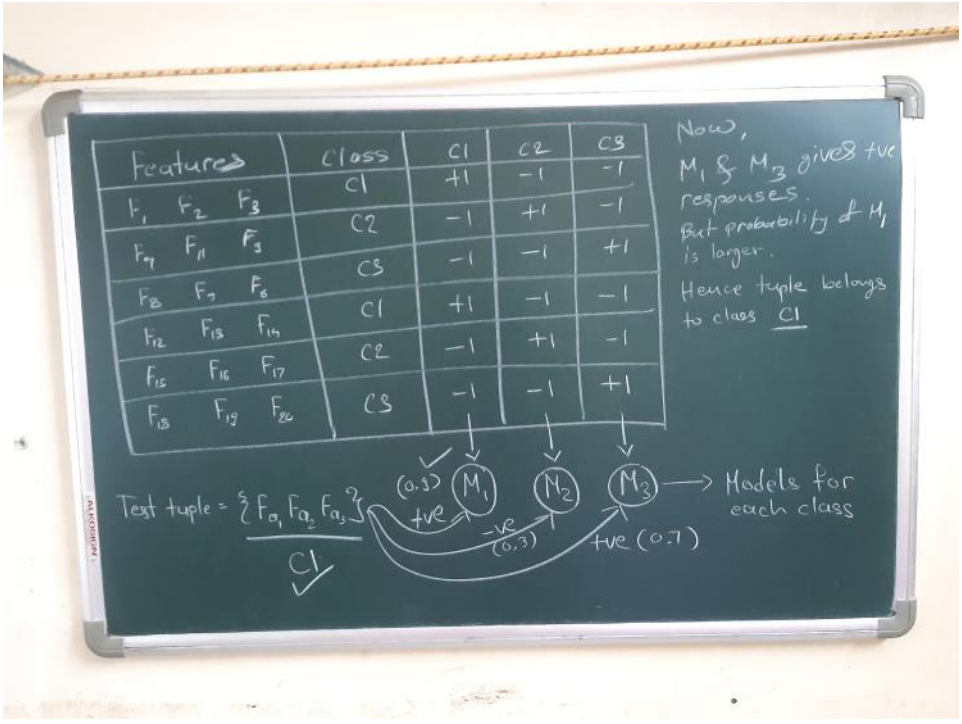


Multiclass Classification -

- The classification in which the data can be classified into multiple classes is known as Multiclass classification.
- This type of data is given to a ML algorithm and it gives a classifier which can classify a data into various classes
- Ex. Part of speech tagging
  In part of speech tags are noun, verb, adverb, etc.
  If we give input as "John" then the classifier will classify it as NOUN.

One vs all -

- In this approach we generate as many models or classifiers as number of classes are present in the dataset
- In other words for each class there is separate classifier
- 1st we give training dataset to the ML algo
- Then as per each tag/class we generate a separate classifier
- Then we give some test tuples to test the Model
- The classifier either gives a positive or negative response
- If more than one classifier gives positive response to the tuple then there probabilities are checked
- The classifier for which the probability of belonging for the tuple is more, it is classified with that tag

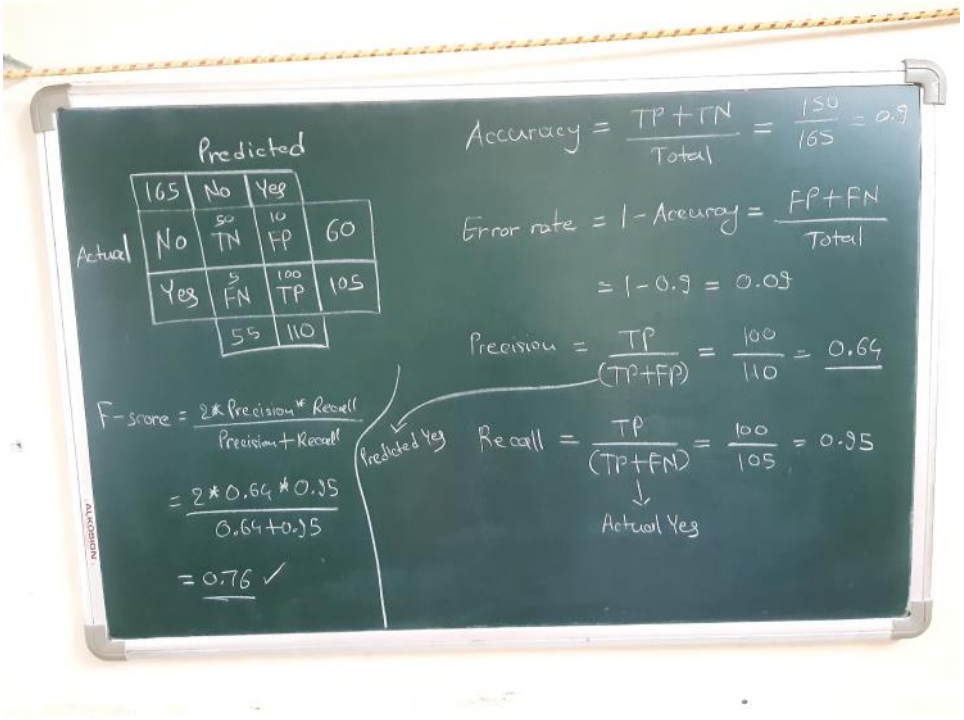


One vs one -

- In this approach if there are N classes are present in the dataset then we generate N(N-1)/2 classifiers
- Means there is one classifier for a pair of classes
- Unlike One vs All there is one classifier for one class

Confusion metrics -

TP = You predicted positive and it's true.
TN = You predicted negative and it's true.
FP/Type 1 error = You predicted positive and it's false.
FN/Type 2 error = You predicted negative and it's false.



Accuracy -

From all the classes (positive and negative), how many of them we have predicted correctly.

Recall -

from all the positive classes, how many we predicted correctly.

Precision -

from all the classes we have predicted as positive, how many are actually positive.

Error Rate -

By how much difference rate the actual and predicted values differ.

F-score -

It is difficult to compare two models with low precision and high recall or vice versa. So to make them comparable, we use F-Score. F-score helps to measure Recall and Precision at the same time.

$$F\text{-}measure = \frac{2*Recall*Precision}{Recall + Precision}$$

Micro-average Method -

In Micro-average method, you sum up the individual true positives, false positives, and false negatives of the system for different sets and the apply them to get the statistics.

True positive (TP1)= 12
False positive (FP1)=9
False negative (FN1)=3

Then precision (P1) and recall (R1) will be 57.14 and 80

True positive (TP2)= 50
False positive (FP2)=23
False negative (FN2)=9

Then precision (P2) and recall (R2) will be 68.49 and 84.75

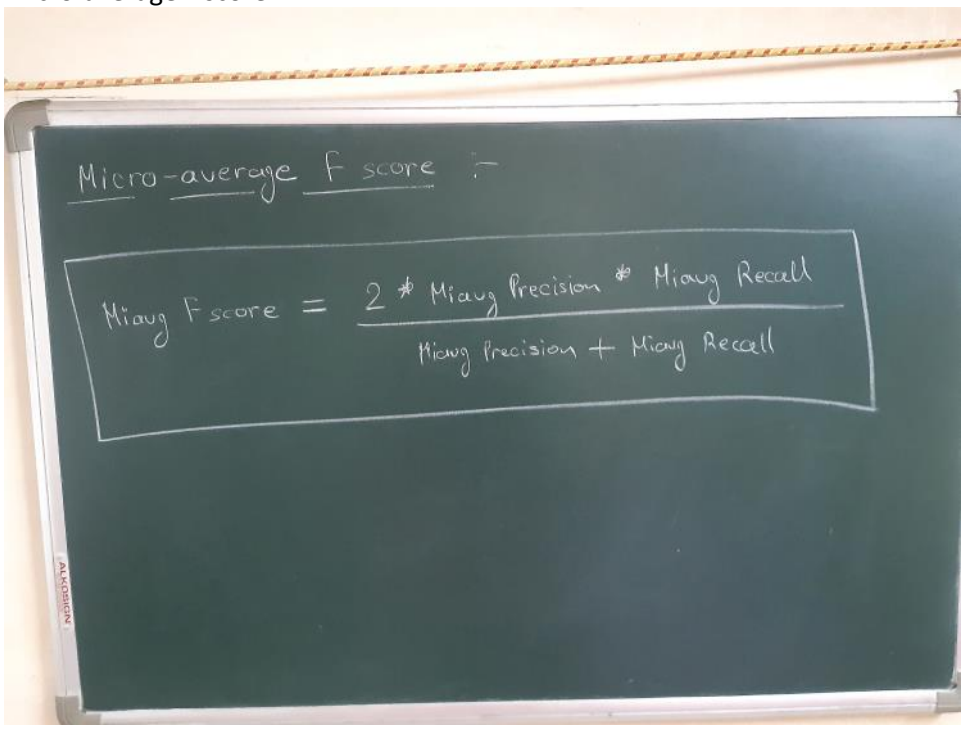
Micro-average of precision = (TP1+TP2)/(TP1+TP2+FP1+FP2) = (12+50)/(12+50+9+23) = 65.96
Micro-average of recall = (TP1+TP2)/(TP1+TP2+FN1+FN2) = (12+50)/(12+50+3+9) = 83.78

Macro-average Method -

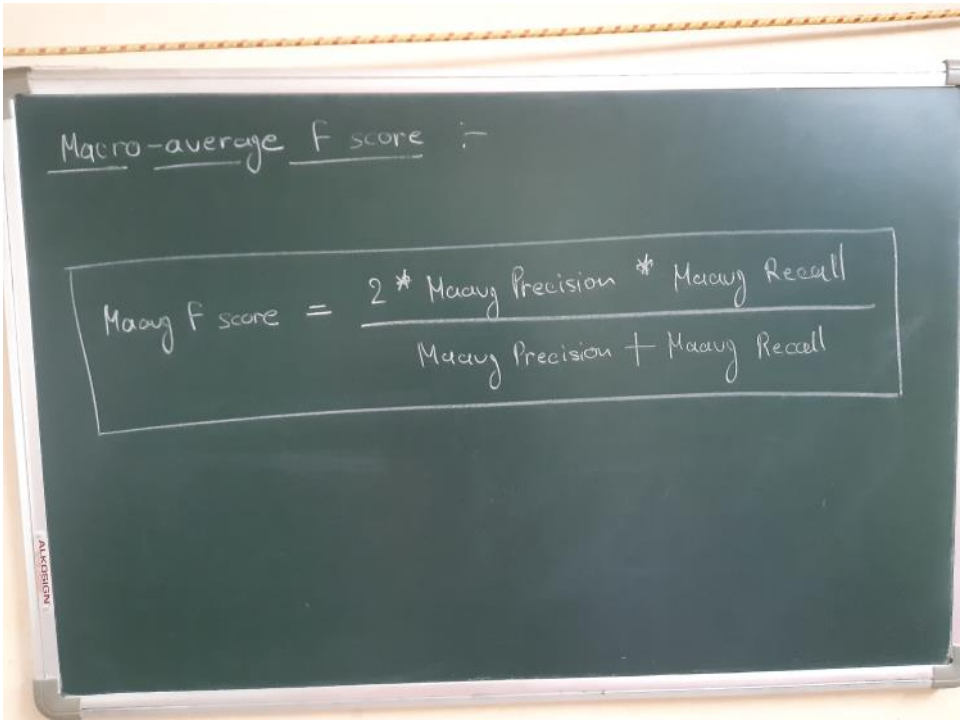
The method is straight forward. Just take the average of the precision and recall of the system on different sets.

Macro-average precision = (P1+P2)/2 = (57.14+68.49)/2 = 62.82
Macro-average recall = (R1+R2)/2 = (80+84.75)/2 = 82.25

Micro-average F score -



Macro-average F score -


Macro-average F score :-
Maavg F score = 2 * Maavg Precision * Maavg Recall / Maavg Precision + Maavg Recall