UNIT IV

CHAPTER 4 Supervised Learning : Classification

Syllabus

Classification: K-nearest neighbour, Support vector machine.

Ensemble Learning: Bagging, Boosting, Random Forest, Adaboost.

Binary-vs-Multiclass Classification, Balanced and Imbalanced Multiclass Classification Problems,Variants of Multiclass Classification: One-vs-One and One-vs-All

Evaluation Metrics and Score: Accuracy, Precision, Recall, Fscore, Cross-validation, Micro- Average Precision and Recall, Micro-Average F-score, Macro-Average Precision and Recall, Macro-Average F-score.

4.1 CLASSIFICATION

GQ. What are the basic concepts of classification ?

We have studied classification and different classifiers earlier. Let's recall some key concepts of classification.

(1) **Classification** is a form of data analysis that extracts models describing data classes. It is a Supervised learning technique that is used to identify the category of new observations on the basis of training data. A classifier, or classification model, predicts categorical labels (classes). Numeric prediction models continuous-valued functions. Classification and numeric prediction are the two major types of prediction problems.

(2) **Decision tree induction** is a top-down recursive tree induction algorithm, which uses an attribute selection measure to select the attribute tested for each non-leaf node in the tree. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. ID3, C4.5, and CART are examples of such algorithms using different attribute selection measures. Tree pruning algorithms attempt to improve accuracy by removing tree branches reflecting noise in the data. Early decision tree algorithms typically assume that the data are memory resident.

(3) **Naive Bayesian classification** is based on Bayes' theorem of posterior probability. It assumes class-conditional independence that the effect of an attribute value on a given class is independent of the values of the other attributes. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object. Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

(4) **A rule-based classifier** uses a set of IF-THEN rules for classification. Rules can be extracted from a decision tree. These rules are easily interpretable and thus these classifiers are generally used to generate descriptive models. The condition used with "if" is called the antecedent and the predicted class of each rule is called the consequent. Rules may also be generated directly from training data using sequential covering algorithms.

4.2 K-NEAREST NEIGHBOR (KNN)

GQ. Explain KNN - algorithm.

K-means is an unsupervised learning algorithm used for clustering problems whereas KNN is a supervised learning algorithm used for classification and regression problems.

This is the basic difference between K-means and KNN algorithm.

4.2.1 K-NN Algorithm

K-NN algorithm at the raining phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example

- Suppose we have an image of a creature that looks similar to crow and Kingfisher, but we want to know either it is a Crow or Kingfisher.
- So for this identification, we can use KNN Algorithm as it works on a similarity measure. Our KNN-model will find the similar features of the new data set to the Crow and Kingfisher images and based on the most similar features it will put it in either Crow or Kingfisher category.
- KNN is one of the simplest machine learning algorithm based on supervised learning technique.
- KNN algorithm assumes the similarity between the new case or data and available cases and put the new case into the category that is most similar to the available category.
- KNN algorithm stares all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well-defined category by using KNN algorithm.
- KNN-algorithm can be used for regression as well as for classification but mostly it is used for the classification problems.
- KNN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called as lazy learner algorithm because it does not lean from the training set immediately, instead it stores the dataset and at the time of classification, it performs an action on the dataset.

4.2.2 Need for KNN-Algorithm

GQ. What is the need of KNN

- Suppose we have two categories, i.e. Crow A and Kingfisher B and we have a new datapoint x_1, so this data point will lie in which of these categories.
- To solve this type of problem, we need a KNN algorithm and with this we can easily identify the category or class of a particular dataset.
- The K-NN working can be explained on the basis of the below algorithm.

Step 1 : Select the number K of the neighbors.

Step 2 : Calculate the euclidean distance of K number of neighbor.

Step 3 : Take the K nearest neighbors as per the calculate Euclidean distance.

Step 4 : Among these K-neighbors, count the number of data points in each category.

Step 5 : Assign the new data points to that category for which the number of the neighbor is maximum.

Now the model is ready to implement.

4.2.3 Selection of Value of K

- There is no particular method to determine the best value for 'K' in KNN-algorithm.
- So we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value of K such as K = 1 or K = 2 can be noisy and lead to effects of outliers in the model.
- Large values of K are good but it may find some difficulties.

4.2.4 Advantages of KNN Algorithm

GQ. Describe advantages and disadvantages of KNN algorithm.

1. It is simple to implement,
2. It is robust to the noisy training data.
3. It can be more effective if the training data is large.

4.2.5 Disadvantage of KNN Algorithm

1. Always needs to determine the value of K which may be complex some time.
2. The computation cost is high because of calculating the distance between the data points for all the training samples.

4.2.6 KNN Classification and Regression

GQ. Compare KNN Classification and Regression.

- In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k-nearest neighbor (K is a positive integer, typically small). If K = 1, then the object is simply assigned to the class of that single nearest neighbor.
- In K-NN regression, the output is the property value for the object. This value is the average of the values of a K nearest neighbors.
- K-NN is a type of classification where the function is only approximated locally and all computation is deferred until function evolution. Since this algorithm relies on distance for classification, if the features represent different physical units or come in vastly different scales then normalising the training data can improve its accuracy dramatically.
- Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones, e.g. a common weighting scheme consists in giving each neighbor a weight of $\frac{1}{d}$, where d is the distance to the neighbor.
- The neighbors are taken from a set of objects for which the class (for K-NN classification) is known. This can be thought of as the training set for the algorithm, though no explicit training is required.
- A peculiarity of K-NN algorithm is that it is sensitive to the local structure of the data.

4.2.7 Parameter Selection

- The best choice of K depends upon the data, generally larger values of K reduces effect of the noise on the classification, but make boundaries between classes less distinct. A good value of K can be selected by various trial techniques. The special case where the class is predicted to be the class of the closest training sample (i.e. when K = 1) is called the nearest neighbor algorithm.

- The accuracy of the KNN algorithm is severely degraded by the presence of noisy or irrelevant features, or if the feature scales are not consistent with their importance. A particularly popular approach is the use of evolutionary algorithm to optimise feature scaling. Another popular approach is to scale features by the mutual information of the training data with the training classes.
- In binary classification problem, it is helpful to choose K to be an odd number as this avoids tied votes. One popular way of choosing the empirically optimal K in this setting is via bootstrap method.

4.2.8 The 1-Nearest Neighbour Classifier

- The most intuitive nearest neighbor type classifier is the one nearest neighbor classifier that assigns a point x to the class of its closest neighbor in the feature space, i.e. $C_n^{1\,nn}(x) = y_{(1)}$.
- As the size of training data set approaches ∞, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayer error rate (the minimum achievable error rate given the distribution of the data).

4.2.9 The Weighted Nearest Neighbour Classifier

- The K-nearest neighbor classifier can be viewed as assigning the K nearest neighbors a weight $\frac{1}{k}$ and all others '0' weight. This can be generalized to weighted nearest neighbor classifier. That is, where the i^{th} nearest neighbor is assigned a weight W_{n_i} with

$$\sum_{i=1}^{n} W_{ni} = 1$$

4.2.10 Properties

GQ. Describe KNN properties

- K-NN is a special case of a variable bandwidth, kernel density with a uniform kernel.
- The naive version of the algorithm is easy to implement by computing the distances from the test example to all stored examples, but it is computationally intensive for large training sets using an approximate nearest neighbor search algorithm makes K-NN computationally tractable even for large data sets. Many nearest neighbor search algorithms have been proposed over the years, those generally seek to reduce the number of distance evaluation actually performed.
- K-NN has some strong consistency results. As the amount of data approaches infinity, the two-class K-algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate. Various improvements to the K-NN are possible by using proximity graphs :

For multi-class KNN classification, an upper bound error rate of

$$R^* \le R_{K\text{-}NN} \le R^*\left[2 - \frac{MR^*}{(M-1)}\right]$$

- Where R* is the Bayes error rate (which is the minimal error rate possible), $R_{K\text{-}NN}$ is K-NN error rate, and M is the number of classes in the problem. For M = 2 and as the Byesian error rate R* approaches zero, the limit reduces the "not more than twice the Bayesian error rate."

4.2.11 Feature Extraction

- When the input data to an algorithm is too large to be processed and it is suspected to be redundant, then the input data will be transformed into a reduced representation set of features (also named feature vector). Transforming the input data into the set of features is called 'feature extraction'.
- If the features extracted are carefully chosen, it is expected that the feature set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. Feature extraction is performed on raw data prior to applying K-NN algorithm on the transformed data in the feature space.

4.2.12 Distance Functions

Euclidean : $\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$ **Manhatten :** $\sum_{i=1}^{k} |x_i - y_i|$

Minkoswski : $\left[\sum (|x_i - y_i|)^q\right]^{1/q}$

The above three distance measures are only valid for continuous variables. But in case of categorical variables, we use 'Hamming distance' which is a measure of number of instances.

Hamming distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

4.2.13 Data Points

- The number of data points that are taken into consideration is determined by the k-values. Thus, the k value is the 'core of the algorithm'. KNN classifier determines the class of a data point by the majority voting principle. If K is set to 5, the classes of 5 closest points are checked.
- KNN algorithm can also be used for regression problems.

4.2.14 Classification Accuracy

Classification accuracy of the KNN algorithm is found to be adversely affected by the presence of outliers, in the experimental datasets. An outlier score based on rank difference can be assigned to the points in these datasets by taking into consideration the distance and the density of their local and neighborhood points.

(A) Advantages

(i) Nonparametric architecture,

(ii) Simple and powerful,

(iii) Requires no training time

(iv) KNN can be used for recommendation systems. Although in the real world, more sophisticated algorithms are used for the recommendation system, KNN is not suitable for high dimensional data, but KNN is an excellent baseline approach for the system.

(v) KNN is not limited to merely predicting groups or values of data points. It can also be used in detecting anomalies. Identifying anomalies can be the end goal in itself, such as in fraud detection.

(B) Disadvantages

(i) Memory intensive

(ii) Classification and estimation are slow.

(iii) The value of K in the KNN-algorithm is related to the error rate of the model. Overfitting imply that the model is well on the training data but has poor performing when the new is coming.

4.2.15 Applications of KNN

(I) Applications of KNN in finance

(i) Forecasting stock market

(ii) Predict the price of a stock on the basis of company performance measures and economic data.

(iii) Currency exchange rate.

(iv) Bank bankruptcies

(v) Understanding and managing financial risk

(vi) Trading futures

(vii) Credit rating

(viii) Loan management

(ix) Bank customer profiling

(x) Money laundering analysis.

(II) Medicine

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.
- Identify the risk factors for prostate cancer, based on clinical and demographic variables.
- The KNN algorithm has been also applied for analyzing micro-array gene expression data, where the KNN algorithm has been coupled with genetic algorithms, which are used as a search tool.
- Other applications include the prediction of solvent accessibility in protein molecules, the detection of intrusions in computer systems, and the management of databases of moving objects such as computer with wireless connections.
- KNN is not limited to merely predicting groups or values of data points. It can also be used in detecting anomalies. Identifying anomalies can be the main aim, such as in fraud detection.

4.3 SUPERVISED LEARNING : SUPPORT VECTOR MACHINE

UQ. Define Support Vector Machine. Explain how margin is computed and optimal hyper-plane is decided ? (Ref. - Dec. 19, 10 Marks)

4.3.1 Optimal Decision Boundary

- A solution to the classification problem is a rule that partitions the features and assigns each and all the features and partition to the same class.
- The **'boundary'** of this partitioning is the decision boundary of the rule.
- The boundary that this rule produces is the optimal decision boundary.
- A decision boundary is the region of a problem space in which the output label of a **classifier** is **ambiguous**.
- If the decision surface is a hyperplane, then the classification problem is linear, and the classes are **linearly separable**.
- Decision boundaries are not always clear-cut. That is, the transition from one class in the **feature space** to another is not discontinuous, but gradual.

GQ. What is decision boundary in decision tree ?

- The first node of the tree called the "root node" contains the number of instances of all the classes respectively.
- Basically, we have to draw a line called "decision boundary" that separates the instances of different classes into different regions called "decision regions".

GQ. What is decision boundary ? How do you draw it and what is the advantage of it ?

A decision boundary is a line (in the case of two features), where all (or most) samples of one class are on one side of that line, and all samples of the other class are on the opposite side of the line. The line separates the two classes.

Maximum Margin Linear Separators

UQ. What is SVM ? Explain the following terms: separating hyperplane, margin and support vectors with suitable example. (Ref. - May 15, 4 Marks)

UQ. What are the key terminologies of Support Vector Machine ? (Ref. - May 16, 5 Marks)

UQ. What is Support Vector Machine ? (Ref. - May 17, Dec. 19, 4 Marks)

UQ. Illustrate Support Vector machine with neat labeled sketch. (Ref. - May 19, 4 Marks)

(1) **Support Vector Machine is a type of supervised learning that can be used for classification or regression. Even if the data points are unseen (not from the training dataset), support vector machine classifies the data properly.**

(2) **Let's take an example of dataset that belongs to two different categories, and the distribution of data is proper means the data is separated from each other properly.**

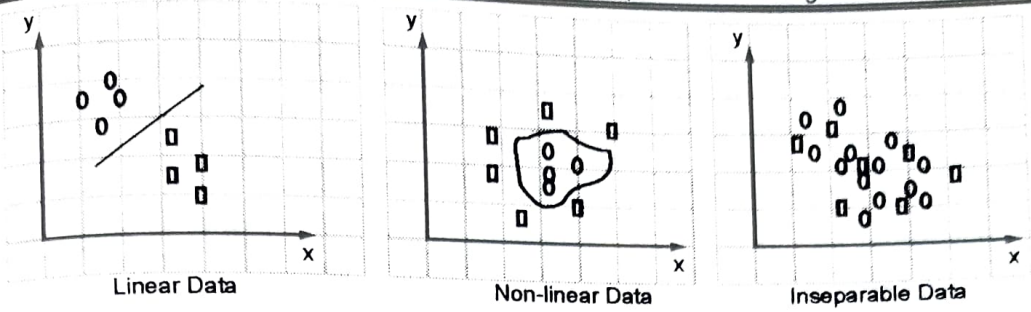

Fig. 4.3.1 : Different types of data

(3) In this case we can draw a straight line (decision boundary) on the graph in such a way that the input space is divided in to two regions.

(4) Data points that belongs to one category lies on one side of the decision boundary and the data points of other category lies on the opposite side. Such type of data is called as linearly separable data.

(5) **Separating hyperplane** is the line which is used to separate the dataset. If we are using simple 2-dimensional plots then it's just a line. We require a plane to separate the data if data is 3 dimensional. So we can say that if data is N dimensional, we require N-1 dimensional hyperplane.

(6) We want that our classifier should be designed in a manner that if a data point is far away from the decision boundary then we will be more confident about the prediction we have made.

(7) We would like to find the data point near to the separating hyperplane and also make sure that this point should be far away from the separating line as possible. This is called as **margin**. We would like to find the greatest possible margin, because if we trained our classifier on limited data or made a mistake, we would want it to be as robust as possible.

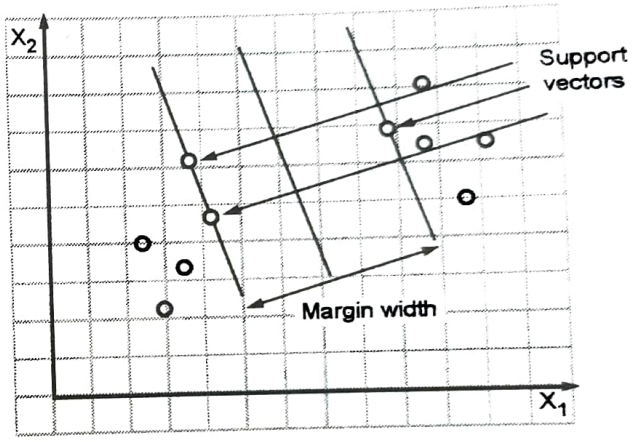

Fig. 4.3.2 : Support vectors and Margin

(8) **Support vectors** are the points which are nearest to the separating hyperplane. We have to maximize the distance between the support vectors and the separating line.

Distance between a hyperplane (w, b) and a point x is calculated as,

$$\text{Distance} = \frac{|\, w^t x + b \,|}{||w||}$$

4.4 ENSEMBLE LEARNING

GQ. Explain the concept of Ensemble learning. **(5 Marks)**

- An ensemble is a **machine learning model that combines** the predictions from two or more models.
- The models that contribute to the ensemble are called as ensemble members.
- They may be of the same type or of different types.
- They may or may not be trained on the same training data.

Remarks

1. An ensemble method is a technique that uses multiple independent similar or different models to derive an output or make some predictions.

   For example, a random forest is an ensemble of multiple decision trees.

2. **Ensemble Learning :** This uses a single machine learning algorithm. It is an unpruned decision tree. It trains each model on a different sample of the same training dataset.

   The predictions made by the ensemble members are combined using simple statistics, such as voting or averaging.

4.5 STUMPING ENSEMBLES

A decision stump is a machine learning model consisting of a one-level decision tree. That is, it is a decision tree with one internal node (the root) which is immediately connected to the terminal nodes (its levels).

A decision stump makes a prediction based on the value of just a single input feature. Sometimes they are also called as **1-rules**.

[An example of a decision stump that discriminates between two of three classes of **iris** flower data set : **iris versicolor** and **iris verginica.** The petal width is in centimetres. This particular stump achieves 94% accuracy on the iris dataset for these two classes.]

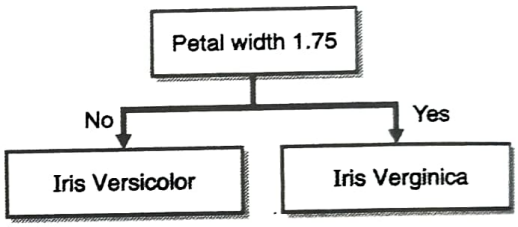

Fig. 4.5.1

Depending on the type of the input features, several variations are possible. For normal features, one may build a stump which contains a leaf for each possible feature value, or a stump with the two leaves, one of which corresponds to some chosen category, and the other leaf to all the other categories.

For binary features these two schemes are identical. A missing value may be treated as a yet another category.

4.5.1 For Continuous Features

Usually, some threshold feature value is selected, and the stump contains two leaves-for values below and above the threshold. However, rarely, multiple thresholds may be chosen and the stump therefore contains three or move leaves.

Decision stumps are often used as components (called "weak learners" or "base learner") in **machine learning ensemble** techniques such as **bagging** and **boosting**.

4.5.2 Remarks

(1) Meaning of stump in decision tree

- A decision stump is a decision tree, which uses only a single attribute for splitting.
- For discrete attributes, this means that the tree consists only of a single interior node (i.e., the root has only leaves as successor nodes).

- If the attribute is numerical, the tree may be more complex.

(2) Are decision stumps linear ?

A decision stump is not a linear model. The decision boundary can be a line, even if the model is not linear.

4.6 BOOSTING IN MACHINE LEARNING

GQ. What is Boosting in machine learning ?

- Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using a weak models in series.
- Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model.
- This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.
- AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification.
- AdaBoost is short for **Adaptive Boosting** and is a very popular boosting technique that combines multiple "weak classifiers" into a single "Strong Classifiers".
- AdaBoost is the best starting point for understanding boosting algorithms. It is adaptive in the sense that subsequent classifiers built are tweaked in favour of those instances misclassified by previous classifiers.
- It is sensitive to noisy data and outliers.
- AdaBoost uses multiple iterations to generate a single composite strong learner. It creates a strong learner by iteratively adding weak learners.
- During each phase of training, a new weak learner is added to the ensemble, and a weighting vector is adjusted to focus on examples that were misclassified in previous rounds. The result is a classifier that has higher accuracy than the weak learner classifiers.

4.7 TECHNIQUES TO IMPROVE CLASSIFICATION ACCURACY

GQ. Mention Techniques to Improve Classification Accuracy .

- In machine learning, no matter if we are facing a classification or a regression problem, the choice of the model is extremely important to have any chance to obtain good results.
- This choice can depend on many variables of the problem: quantity of data, dimensionality of the space, distribution hypothesis, etc.
- In ensemble learning theory, we call weak learners (or base models) models that can be used as building blocks for designing more complex models by combining several of them.

- The idea of ensemble methods is to try reducing bias and/or variance of such weak learners by combining several of them together in order to create a strong learner (or ensemble model) that achieves better performances.
- To outline the definition and practicality of Ensemble Methods, here we have used example of Decision tree classifier. However, it is important to note that Ensemble Methods do not only pertain to Decision Trees.
- A decision tree determines the predictive value based on series of questions and conditions. For instance, the simple Decision Tree shown in Fig. 4.7.1 determines on whether an individual should play outside or not.
- The tree takes several weather factors into account, and given each factor either makes a decision or asks another question. In this example, every time it is overcast, we will play outside.
- However, if it is raining, we must ask if it is windy or not? If windy, we will not play.
- But given no wind, tie those shoelaces tight because were going outside to play.

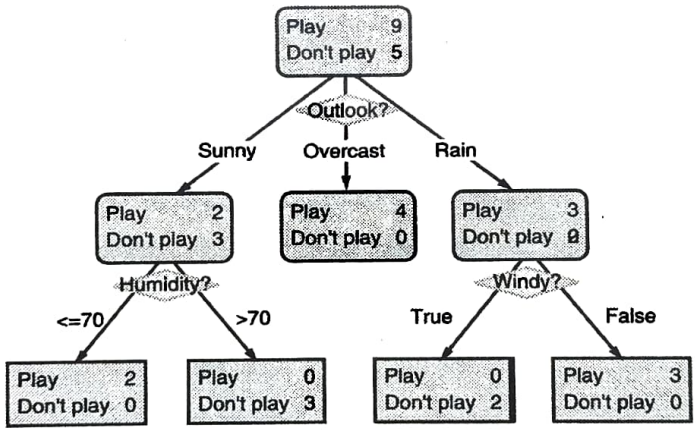

Fig. 4.7.1 : A decision tree to determine whether to play outside or not

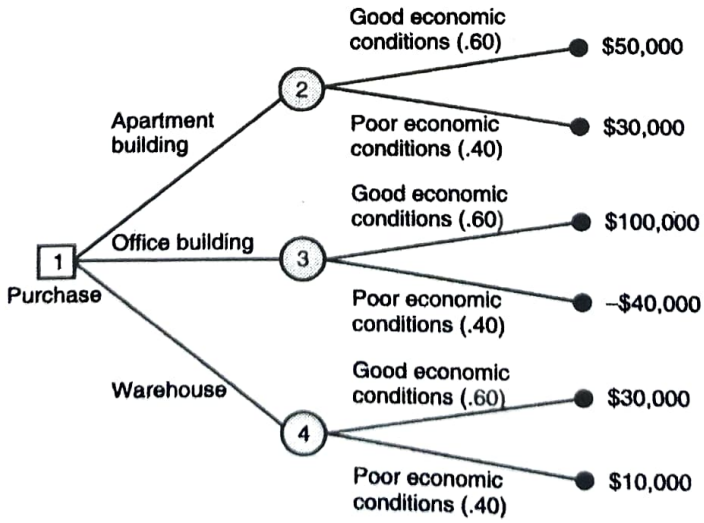

Fig. 4.7.2 : A decision tree to determine whether or not to invest in real estate

- Decision Trees can also solve quantitative problems as well with the same format. In the Tree to the left, we want to know whether or not to invest in a commercial real estate property. Is it an office building?
- A Warehouse? An Apartment building? Good economic conditions? Poor Economic Conditions? How much will an investment return? These questions are answered and solved using this decision tree.
- When making Decision Trees, there are several factors we must take into consideration: On what features do we make our decisions on? What is the threshold for classifying each question into a yes or no answer? In the first Decision Tree, what if we wanted to ask ourselves if we had friends to play with or not.
- If we have friends, we will play every time. If not, we might continue to ask ourselves questions about the weather. By adding an additional question, we hope to greater define the Yes and No classes.
- This is where Ensemble Methods come into picture! Rather than just relying on one Decision Tree and hoping we made the right decision at each split, Ensemble Methods allow us to take a sample of Decision Trees into account, calculate which features to use or questions to ask at eac h split, and make a final predictor based on the aggregated results of the sampled Decision Trees.

4.7.1 Introducing Ensemble Methods

- **Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model which helps to improve machine learning results.
- This approach allows the production of better predictive performance compared to a single model. Basic idea is to learn a set of classifiers and to allow them to vote. Ensembles tend to be more accurate than their component classifiers.

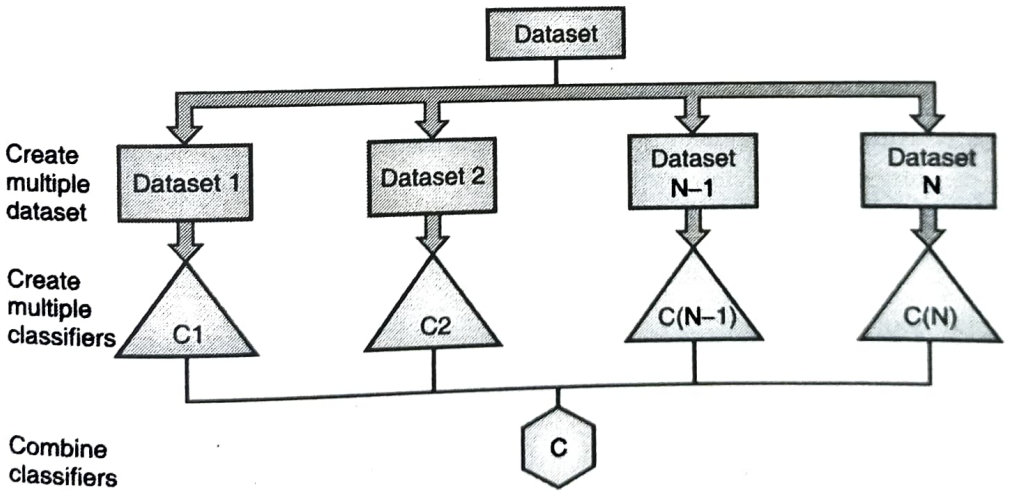

Fig. 4.7.3 : An overview of Ensemble methods/learning

GQ. Explain different types of ensemble classifiers.

- Different types of ensemble classifiers are:
  1. Bagging
  2. Boosting and AdaBoost
  3. Random Forests

4.7.2 Bagging (Bootstrap Aggregating)

- This approach combines **Bootstrapping** and **Aggregation** to form one ensemble model, that's why the name is **Bagging**.
- Consider yourself as a patient and you would like to have a diagnosis made based on the symptoms. Instead of asking one doctor, you may choose to ask several.
- If a certain diagnosis occurs more than any other, you may choose this as the final or best diagnosis.
- That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote.
- If we replace each doctor by a classifier, and that's the basic idea behind bagging.
- Naturally, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.
- Given a sample of data, multiple bootstrapped subsamples are pulled.
- A Decision Tree is formed on each of the bootstrapped subsamples.
- Each training set is a bootstrap sample. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor.

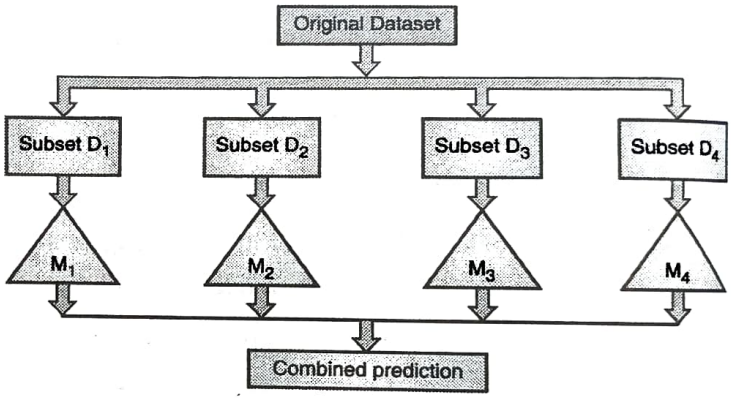

Fig. 4.7.4 : Bagging

- To classify an unknown tuple, X, each classifier, M_i, returns its class prediction, which counts as one vote. The bagged classifier, M*, counts the votes and assigns the class with the most votes to X.
- Bagging often considers homogeneous weak learners, learns them independently from each other in parallel and combines them following some kind of deterministic averaging process Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.

4.7.3 Boosting and AdaBoost

- Consider the same example that was taken in previous section, you as a patient, you have certain symptoms.
- Now, instead of consulting one doctor, you choose to consult several.
- Suppose you assign weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made.
- The final diagnosis is then a combination of the weighted diagnoses. This is the basic idea behind boosting.

- Boosting often considers homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy.
- In boosting, weights are also assigned to each training tuple.
- A series of k classifiers is iteratively learned. After a classifier, M_i, is learned, the weights are updated to allow the subsequent classifier, $M_i + 1$, to "pay more attention" to the training tuples that were misclassified by M_i.
- The final boosted classifier, M*, combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.
- In **adaptative boosting** (often called "adaboost"), we try to define our ensemble model as a weighted sum of L weak learners. It's a popular boosting algorithm.
- The basic idea is that when we build a classifier, we want it to focus more on the misclassified tuples of the previous round.
- Some classifiers may be better at classifying some "difficult" tuples than others. In this way, we build a series of classifiers that complement each other.
- We are given D, a data set of d class-labeled tuples, $(X_1, y_1),(X_2, y_2),...,(X_d, y_d)$, where y_i is the class label of tuple Xi. Initially, AdaBoost assigns each training tuple an equal weight of 1/d. Generating k classifiers for the ensemble requires k rounds through the rest of the algorithm.
- In round i, the tuples from D are sampled to form a training set, D_i, of size d. Sampling with replacement is used. This indicates the same tuple may be selected more than once.
- Each tuple's chance of being selected is based on its weight. A classifier model, M_i, is derived from the training tuples of D_i.
- Its error is then calculated using D_i as a test set. The weights of the training tuples are then adjusted according to how they were classified.
- If a tuple was incorrectly classified, its weight is increased. If a tuple was correctly classified, its weight is decreased.
- A tuple's weight reflects how difficult it is to classify. The higher the weight, the more often it has been misclassified.
- These weights will be used to generate the training samples for the classifier of the next round. This is how, a series of classifiers that complement each other are built.
- To compute the error rate of model M_i, we sum the weights of each of the tuples in D_i that M_i misclassified.

$$\text{error}(M_i) = \sum_{j=1}^{d} w_j \times \text{err}(X_j)$$

where $err(X_j)$ is the misclassification error of tuple X_j: If the tuple was misclassified, then $err(X_j)$ is 1; otherwise, it is 0. If the performance of classifier Mi is so poor that its error exceeds 0.5, then we abandon it. Instead, we try again by generating a new D_i training set, from which we derive a new M_i.

- The error rate of M_i affects how the weights of the training tuples are updated.
- If a tuple in round i was correctly classified, its weight is multiplied by $error(M_i)/(1 - error(M_i))$.
- Once the weights of all the correctly classified tuples are updated, the weights for all tuples (including the misclassified ones) are normalized so that their sum remains the same as it was before.
- To normalize a weight, we multiply it by the sum of the old weights, divided by the sum of the new weights. As a result, the weights of misclassified tuples are increased and the weights of correctly classified tuples are decreased.
- To predict a class label for a tuple X, boosting assigns a weight to each classifier's vote, based on how well the classifier performed. The lower a classifier's error rate, the more accurate it is, and therefore, the higher its weight for voting should be. The weight of the classifier's vote is calculated as :

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

- For each class, c, we sum the weights of each classifier that assigned class c to X.
- The class with the highest sum is the "winner" and is returned as the class prediction for tuple X.
- Bagging is less susceptible to model overfitting. While bagging and boosting, both can significantly improve accuracy in comparison to a single model, boosting tends to achieve greater accuracy.

4.8 RANDOM FOREST

GQ. Explain random forest algorithm in detail.

4.8.1 Random Forest

- Random forest is a supervised machine learning algorithm that is used widely in classification and Regression problems.
- It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

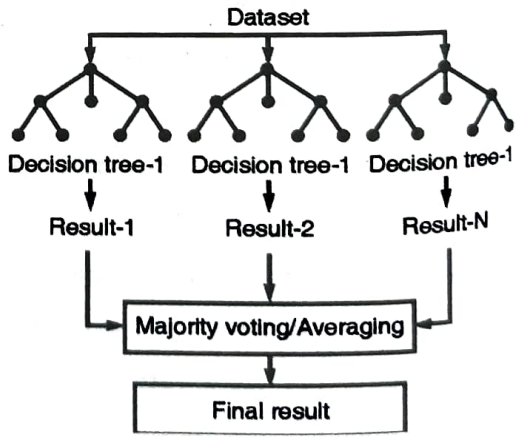

Fig. 4.8.1

4.8.2 Random Forest Algorithm

- Random forest Algorithm can handle the data set containing **continuous variables** in case of regression and **categorical variables** in case of classification.

- It performs better results for classification problems.

Working of Random Forest Algorithm

Ensemble Technique

- Ensemble means combining multiple models. Here a collection of models make predictions rather than an individual model.
- Ensemble uses two types of methods
  1. **Bagging :** It creates a different training subset from sample training data with replacement. The final output is based on majority voting. For example, Random forest.
  2. **Boosting :** It creates sequential models such that the final model has the highest accuracy. Then it combines weak learners into strong learners. For example, ADA BOOST, XG BOOST.

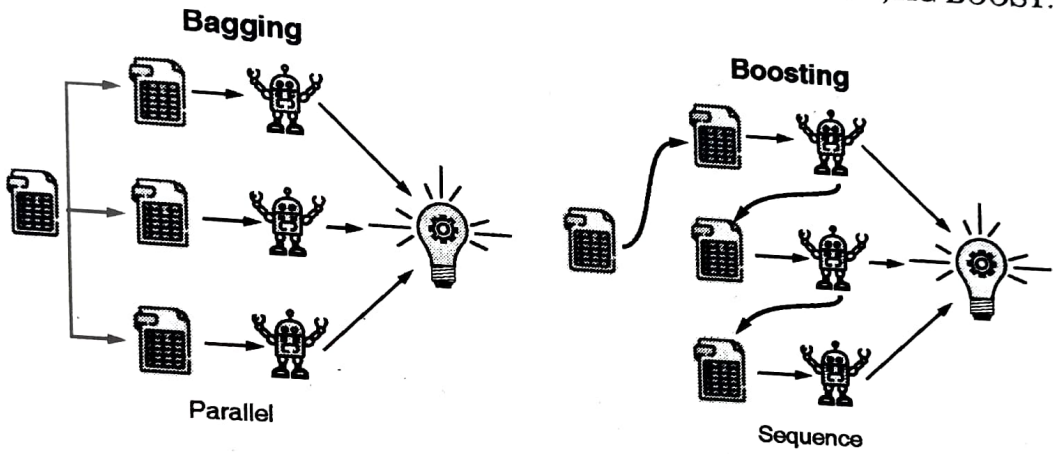

Fig. 4.8.2

Remark

Random forest works on the Bagging principle.

4.8.3 Bagging

- Bagging is also known as Bootstrap Aggregation. It is an ensemble technique used by random forest.
- Bagging chooses a random sample from the data set.
- Hence each model is generated from the samples (Bootstrap samples) provided by the original data with replacement. It is known as Row sampling.

4.8.4 Row Sampling

- This step of row sampling with replacement is called bootstrap.
- Each model is trained independently and it generates results. The final output is based on majority voting after combining the results of all models.

- The output that is based on majority voting is known as **aggregation.**
- We observe that the bootstrap sample is taken from the actual data (Bootstrap sample 01, Bootstrap sample 02 and Bootstrap sample 03) with a replacement.
- Clearly each sample will not contain unique data.
- Now, the model, Model 01, Model 02, Model 03 obtained from this bootstrap sample are trained independently.

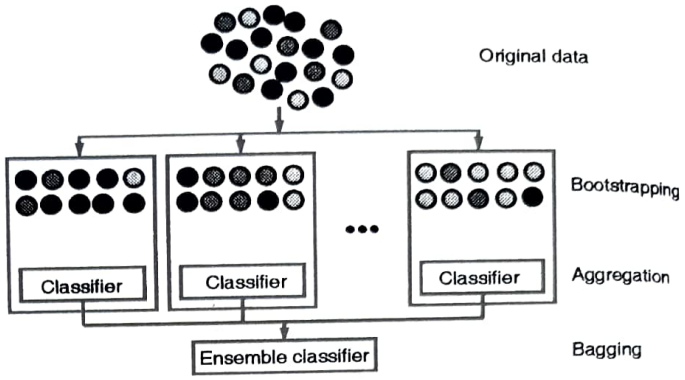

Fig. 4.8.3

- Each model generates results. And based on majority voting final output is obtained

☞ **Steps involved in random forest algorithm**

- **Step 1 :** In Random forest n number of random records are taken from the data set having k number of records.
- **Step 2 :** Individual decision trees are constructed for each sample.
- **Step 3 :** Each decision tree will generate an output.
- **Step 4 :** Final output is based on **majority voting or averaging** for classification and regression respectively.

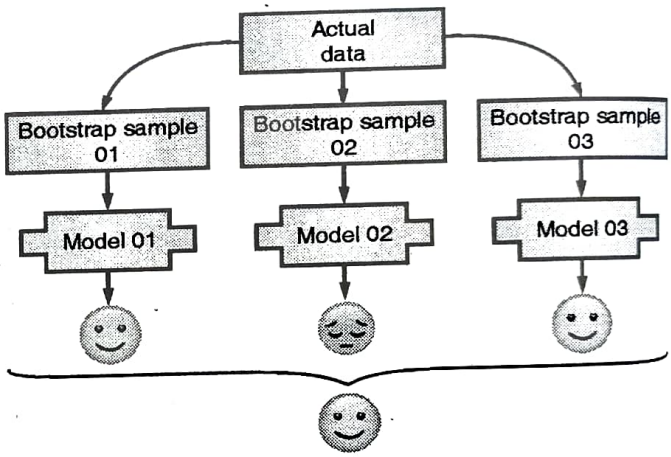

Fig. 4.8.4

4.8.5 Important Features of Random Forest

GQ. What are features of random forest.

1. **Diversity :** Not all attributes / Variables / features are considered while making an individual tree, each tree is different.
2. **Immune to the curse of dimensionality :**
   Since each tree does not consider all the features, the feature-space is reduced.
3. **Parallelisation :** Each tree is created independently out of different data and attributes. This implies that we can make full use of CPU to build random forests.
4. **Train-Test split :** In a random forest we don't have to segregate the data for training and testing. It is because there will always be 30% of the data which is not seen by the decision-tree.
5. **Stability :** Stability arises because the result is based on majority voting / averaging.

4.8.6 Difference Between Decision Tree and Random Forest

GQ. Mention difference between decision tree and random forest.

Actually, Random forest is a collection of decision trees, still there are differences in their behaviour.

Sr. No.	Decision Trees	Random Forest
1.	Decision trees normally suffer from the problem of overfitting. Since it is allowed to grow without any control.	Random forests are created from subsets of data and the final output is based on average or majority ranking and hence the problem of overfitting is taken care of.
2.	A single decision tree is faster in computation.	It is comparatively slower.
3.	When a dataset with features is taken as input by a decision tree, it will formulate some set of rules to do prediction.	Random forest randomly selects observations, builds a decision tree and the average result is taken. It does not use any set of formulas.

We conclude that if the trees are diverse and acceptable then random forests are much more successful than decision trees.

☞ **Important Hyper parameters**

Hyper parameters are used in random forests to

(i) Increase the performance power and (ii) Predictive power of models or

(iii) To make the model faster.

☞ **(I) Following hyper parameters increase the predictive power**

1. **n-estimators :** Number of trees the algorithm builds before averaging the predictions.
2. **Max-features :** Maximum number of features-random forest split a node.
3. **Mini-sample-leaf :** It determines the minimum number of leaves to split on internal node.

☞ **(II) Following hyper parameters increases the speed**

1. **n-jobs :** It tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is – 1, there is no limit.
2. **Random-state :** It controls the randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyper parameters and the same training data.
3. **OOB-score :** OOB implies out of the bag. It is a random forest cross-validation method. In this method, one-third of the sample is not used to train the data instead it is used to evaluate the performance. These samples are called as out of bag samples.

4.8.7 Advantages and Disadvantages of Random Forest Algorithm

GQ. Explain advantages and disadvantages of random forest algorithm.

☞ **Advantages**

1. It can be used in classification and regression problems.
2. If solves the problem of overfitting as output is based on majority voting or averaging.
3. Even if data contains missing values or null values, it performs well.
4. Each decision tree created is independent of the other. It means that is shows the property of parallelisation.
5. It is highly stable as the average answers given by a large number of trees are taken.
6. If maintains diversity as all the attributes are not considered while making each decision tree even through it is not true in all cases.
7. It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature-space is reduced.
8. We need not segregate data to train and test, as there will always be 30% of the data which is not seen by the decision tree which is made out of **bootstrap**.

☞ **Disadvantages**

1. Random forest is highly complex when compared to decision trees where decisions can be made by following the path of the tree.
2. Training time is more compared to other models due to its complexity.

Whenever it has to make a prediction, each decision tree has to generate output for the given input data.

☞ **Remark**

1. Random forest can handle binary, continuous and categorical data.
2. Random forest can handle missing values.
3. Random forest is a fast, simple, flexible and robust model with some limitations.

4.9 COMPARISON BETWEEN BOOSTING RANDOM FOREST AND BOOSTING

GQ. Give comparison between boosting random forest and boosting.

Sr. No.	Random Forest	Boosting
1.	Random forest is a bagging technique	The decision trees are built additively, i.e. each decision tree is built one after another
2.	It is not a boosting technique.	In boosting, one is learning from other, which in turn boosts the learning.
3.	The trees in random forest run parallel.	Boosting is an approach to increases the complexity of models that suffer from high bias, that is, models that underfit the training data.

Sr. No.	Random Forest	Boosting
4.	There is no interaction between these trees while building the trees.	Boosting reduces error mainly by reducing bias, and also to some extent variance, by aggregating the output from many models.
5.	Random forest and bagging are "bagging" algorithms that aim to reduce the complexity of models that overfit the training data.	XG boost always gives more importance to functional space when reducing the cost of a model.
6.	Random forest uses fully grown decision trees (low bias, high variance). It tackles the error reduction in the opposite way : by reducing variance.	Gradient boosting may not be a good choice it there is lot of noise, as it can result in overfitting.
7.	Random forest tries to give more preferences to hyper parameters to optimise the model.	They also tend to be harder to tune than random forest.
8.	Random forest may not be give better performance compared to gradient boosting.	Gradient boosting trees can be more accurate than random forests.
9.	The basic idea in random forest is : although a single tree may be inaccurate, the collective decisions of a bunch of trees are likely to be right most of the time.	We train gradient boosting trees to correct each other's errors, they can capture complex patterns in the data.
10.	Forests are more robust and more accurate than a single tree. But, they are harder to interpret since each classification decision or regression output is not one but multiple decision paths.	If the data is noisy, the boosted trees may overfit and start modelling the noise.

4.10 BALANCED-IMBALANCED MULTI-CLASSIFICATION

- Class imbalance is a problem that occurs in machine learning classification problems. It says that the target class's frequency is highly imbalanced, i.e., the occurrence of one of the classes is very high compared to the other classes present.
- In other words, there is a bias or skewness towards the majority class present in the target.
- Fir example, suppose a binary classification has 10000 rows, and the minority target has only 100 rous. In this case the ratio is 100 : 1. This problem is a problem of a class-imbalance.
- Some frequently occurring examples are : Fraud detection, churn prediction, medical diagnosis, e-mail classification etc.

4.10.1 Handling of Multiclass imbalanced Data

There are different methods of handling imbalance data, the most common methods are (i) oversampling and (ii) creating synthetic strategy.

(I) Oversampling Technique

SMOTE

SOMTE is an oversampling technique that generates synthetic samples from the data set which increases the predictive power for minority classes.

There is no loss of information but it has a few limitations.

☞ **Limitations**

1. SMOTE is not very good for high dimensionality data.
2. Overlapping of classes may happen and can introduce more noise to the data.

So, to avoid this problem, we can assign weights for the class manually with the **'class-weight'** parameter.

Why to use class-weight?

Class weights modify the loss function directly by giving a penalty to the class with different weights.

It implies that, (we are)

Purposely increasing the power of the minority class and reducing the power of the majority class.

Hence, it gives better results than SMOTE,

(ii) Creating synthetic strategy

A few more techniques for getting the weights which can work for imbalanced learning problems :

1. Skleam utils
2. Counts to length, Ratio
3. Smoothen weights.
4. Sample weight strategy.

1. Skleam utils

Using skleam we can compute the class weight. Adding those weight to the minority classes, it can help the performance while classifying the models.

2. Counts to length, Ratio

Here, just divide the number of counts of each class with the number of rows.

3. Smoothen weights technique

This is the preferable method of choosing weights.

The log function smooths the weights for the imbalanced class.

4. Sample weight strategy.

Sample weight is an array of the same length as data, containing weights to apply to the model's loss for each sample.

This means that one should pass a weight for each class that is to be classified.

4.11 BALANCED ACCURACY

- Balanced accuracy is useful for multiclass classification.
- For a balanced dataset, the scores tend to be the same as Accuracy.

Issues with Balanced Accuracy

We mention some situations

(i) When data is balanced.

(ii) When model provides a wide range of possible outcomes (probability).

(iii) When the model is to give more preference to its positives than negatives.

(iv) In multiclass classification, where importance is not placed on some classes than others, bias can take place. It is because all classes have the same weights regardless of class frequency.

When there is a high skew, then balanced accuracy is not a perfect judge for the model.

4.12 VARIANTS OF MULTICLASS CLASSIFICATION

- In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of three or more classes.
- Many classification algorithms, for example multinomial logistic regression, can turn binary algorithms into multinomial classifiers by a variety of strategies.
- The existing multi-class classifications techniques can be categorized into :

(i) transformation to binary,
(ii) extension from binary, and
(iii) hierarchical classification

(i) Transformation to Binary

We discuss problem of multiclass classification to multiple binary classification.

It can be categorised into one vs rest and one vs one.

(a) One-vs-rest [or one-vs-all]

- One-vs-all strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives.

- This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label.
- The training algorithm for one-vs-all learner constructed from a binary classification learner L is as follows :

Inputs

1. L, a learner (training algorithm for binary classifiers)
2. Samples X
3. labels y where $y_i, \in \{1, ..., K\}$ is the label for the sample X_i.

Output

1. a list of classifiers f_k for KE {1, ...,K}

Procedure

For each K in {1, ..., K}

(i) Construct a new label vector Z where $Z_i = Y_i$; if $Y_k = K$ and $Z_i = 0$ otherwise.

(ii) Apply L to X, Z to obtain f_k

- Making decisions means applying all classifiers to an unseen sample x and predicting the label k for which the corresponding classifier reports the highest confidence score.
- This strategy is popular, but it is a heuristic and suffers from several problems.
- Firstly, the scale of the confidence values may differ between the binary classifiers.
- Second, even if the class distribution is balanced, the binary classification learners see unbalanced distribution. It is because the set of negatives is much larger than the set of positives.

(b) One-vs-one

- In the one-vs-one reduction. One trains $\frac{K(K-1)}{2}$ binary classifiers fro a K-way multiclass problem. Each receives the samples of a pair of classes from the original training set, and learns to distinguish these two classes.
- At prediction time, a voting scheme is applied : All $\frac{K-(K-1)}{2}$ classifiers are applied to an unseen sample and the class that gets the highest number of '+1' predictions, gets predicted by the combined classifier.

- Like one vs all, one vs one suffers from ambiguities in that some regions of its input space may receive the same number of votes.

(ii) Extension from binary

- Here we discuss strategies of extending the existing binary classifiers to solve multi-class classification problems.
- Several algorithms are developed and they base on neural networks, decision trees. K-nearest neighbours, naïve Bayes, support vector machines to address multi-class classification problems. We have already discussed these techniques.

(iii) Hierarchical classification

- Hierarchical classification divides the out space into a tree of the multi-class classification problem.
- Each parent node is divided into multiple child nodes and the process is continued till each child node represents only one class.

4.13 CLASSIFICATION METRIC

GQ. Explain Classification Metric.

Classification models have discrete output, hence we need a metric that compares discrete classes in some form. Classification metrics evaluate a models performance and tell how good or bad the classification is

4.13.1 Performance Metrics in Machine Learning

- Performance metrics are a part of every machine learning. They tell us if we are making progress, and then put a number on it. All machine learning models, like linear regression, need a metric to judge performance.
- Every machine learning task can be broken down to either regression or classification.

4.13.2 Metrics are Different from Loss Functions

- Loss functions show a measure of model performance. They are used to train a machine learning model (using some kind of optimisation like gradient descent or stochastic gradient descent) and they are generally differentiable in the model's parameters.
- Metrics are used to monitor and measure the performance of a model (during training and testing) and don't need to be differentiable.

4.13.3 Classification Metrics

The most frequent classification evaluation metric should be 'Accuracy'. We mention some aspect

(1) The confusion matrix for a 2-class classification problem

(2) The key-classification metrics : Accuracy, Recall, precision and F1-score.

(3) The difference between Recall and Precision in specific cases

(4) Decision thresholds

(1) Confusion matrix

- Evaluation of the performance of a classification model is based on the counts of test records correctly or incorrectly predicted by the model.
- The confusion matrix provides a more insightful picture which is not only the performance of a predictive model, but also it tells, which classes are being predicted correctly and incorrectly. And also it tells, what type of errors are being made.
- We see how the 4-classification metrics are calculated (TP, FP, FN, TN)
- We also present our predicted value compared with actual value in a confusion matrix

	Actual value	
	Positive	**Negative**
Positive	TP (True Positive)	FP (False Positive)
Negative	FN (False Negative)	TN (True Negative)

Fig. 4.13.1

True Positive (TP)	Observation is positive, and is predicted to be positive
False Negative (FN)	Observation is positive but is predicted negative
True Negative (TN)	Observation is negative and is predicted to be negative
False Positive (FP)	Observation is negative, but is predicted positive

The confusion matrix is useful for measuring recall (also known as sensitivity) precision, specificity, accuracy and also AUC ROC curve

4.13.4 The Equations of 4 Key Classification Metrics

(i) $$\text{Accuracy} = \text{ACC} = \frac{TP + TN}{TP + TN + FP + FN}$$

(ii) $$\text{Precision : Precision} = \frac{TP}{TP + FP}$$

(iii) $$\text{Recall : Recall} = \frac{TP}{TP + FN}$$

(iv) $$F_1 \text{ score} : F_1 = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

Recall versus precision

(a) **Precision** is the ratio of 'True positives' to all the positives predicted by the model.

Low precision : Model predicts the more false positives, the lower the precision.

(b) **Recall (sensitivity) :** It is the ratio of **True Positives** to all the positives in your dataset.

Low recall : The more false negatives the model predicts, the lower the recall.

We mention example to illustrate the difference in the above cases

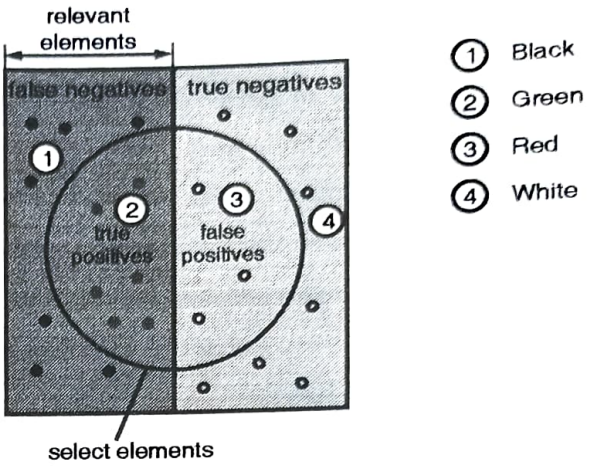

Fig. 4.13.2

(i) The result of TP will indicate that the COVID 19 residents diagnosed with COVID 19

(ii) The result of TN indicates healthy residents are with good health.

(iii) The result FP indicates that those actually healthy residents are predicted as COVID 19 residents

(iv) The result of FN indicates that those actual COVID 19 residents are predicted as the healthy residents

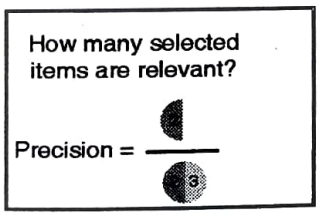

Fig. 4.13.3

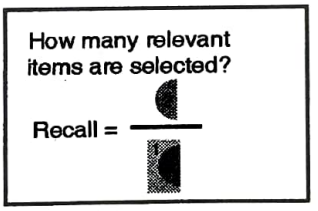

Fig. 4.13.4

- Let us study which picture will have the hightest cost ?
- If we predict COVID 19 residents as healthy patients, there would be a massive number of COVID 19 infections. The cost of **false negatives** is much higher than the cost of **false positives.**

4.14 STEPS IN ML MODELING

GQ. Describe steps in ML modeling in details.

Step 1 : Data Collection

- Given the problem you want to solve, you will have to investigate and obtain data that you will use to feed your machine.
- The quality and quantity of information you get are very important since it will directly impact how well or badly your model will work.

- You may have the information in an existing database or you must create it from scratch.
- You could collect the samples from a website and extracting data.
- From RSS feed or an API
- From device to collect wind speed measurement
- Publicly available data.

Step 2 : Data pre-processing

- Once you have the input data, you need to check whether it's in a useable format or not.
- Some algorithm can accept target variables and features as string; some need them to be integers.
- Some algorithm accepts features in a special format.
- Looking at the data you have passed in a text editor to check collection and preparation of input data steps are properly working and you don't have a bunch of empty values.
- You can also check at the data to find out if you can see any patterns or if there is anything obvious, such as a few data points greatly differ from remaining set of the data.
- Plotting data in 1, 2 or 3 dimensions can also help.
- Distil multiple dimensions down to 2/3 so that you can visualize the data.
- The importance of this step is that it makes you understand that you don't have any garbage value coming in.

Step 3 : Model Selection

- There are various models that we can select according to the objective that we might have: we will use algorithms of classification, prediction, linear regression, clustering, i.e. k-means or K-Nearest Neighbour, Deep Learning, i.e. Neural Networks, Bayesian, etc.
- There are various models to be used depending on the data you are going to process such as images, sound, text, and numerical values.
- In the 4.14.1 table, we will see some models and their applications that we can apply in our projects:

Table 4.14.1 : Model Selection

Model	Applications
Logistic Regression	Price prediction
Fully connected networks	Classification
Convolutional Neural Networks	Image processing
Recurrent Neural Networks	Voice recognition
Random Forest	Fraud Detection
Reinforcement Learning	Learning by trial and error
Generative Models	Image creation
K-means	Segmentation
k-Nearest Neighbors	Recommendation systems
Bayesian Classifiers	Spam and noise filtering

Step 4 : Model training

(1) Training

- A training set comprises of training examples which will be used to train machine learning algorithms.
- Good clean data from the first two steps is given as input to the algorithm. The algorithm extracts information or knowledge. This knowledge is mostly stored in a format that is readily useable by machine.
- In case of unsupervised learning, training step is not there because target value is not present. Complete data is used in the next step.
- You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.
- Remember to initialize the weights of your model randomly -the weights are the values that multiply or affect the relationships between the inputs and outputs- which will be automatically adjusted by the selected algorithm the more you train them.

(2) Testing

- To test machine learning algorithms what's usually done is to have a training set of data and a separate dataset, called a test set.
- In this step the information learned in the previous step is used. When you are checking an algorithm, you will test it to find out whether it works properly or not. In supervised case, you have some known values that can be used to evaluate the algorithm.
- In case of unsupervised, you may have to use some other matrices to evaluate the success. In either case, if you are not satisfied, you can again go back to step 4, change some things and test again.
- Mostly problem occurs in collection or preparation of data and you will have to go back to step 1.

(3) Process of Training and Testing

- Suppose we want to use a machine learning algorithm for classification. The next step is to train the algorithm, or allows it to learn. To train the algorithm we give as a input a quality data called as training set.
- Each training example has some features and one target variable. The target variable is what we will be trying to predict with our machine learning algorithms. In a training dataset the target variable is known. The machine learns by finding some relationship between the target variable and the features. In the classification tasks the target variables are known as classes. It is assumed that there will be a limited number of classes.
- The class or target variable that the training example belongs to is then compared to the predicted value, and we can get an idea about the accuracy of the algorithm.

- **Example :** First we will see some terminologies that are frequently used in machine learning methods. Let's take an example that we want to design a classification system that will classify the instances in to either Acceptable or Unacceptable. This kind of system is a fascinating topic often related with machine learning called *expert systems*.
- Four features of the various cars are stored inTable 4.14.2. The features or the attributes selected are Buying_Price, Maintenance_Price, Lug_Boot and Safety. Examples belong to Table 4.14.2 represents a record comprises of features.
- In Table 4.14.2 all the features are categorical in nature and takes limited disjoint values. The first two features represent the buying price and maintenance price of a car such as high, medium and low. Third feature shows the luggage capacity of a car as small, medium or big. Fourth feature represents whether the car has safety measures or not, which takes the value as low, medium or high.
- Classification is one of the important task in machine learning. In this application we want to evaluate the car out of a group of other cars. Suppose we have allinformation about car's Buying_Price, Maintenance_Price, Lug_Boot and Safety.
- Classification method is used to evaluate a given car as Acceptable or Unacceptable. Many machine learning algorithms are there that can be used for classification. The target or the response variable in this example is the evaluation of a car.
- Suppose we have selected a machine learning algorithm to use for classification. The main task in the classification is to train the algorithm, or allow it to learn. We give the experienced data as the input to train the algorithm which is called as training data.
- Let's assume training dataset contains 14 training records in Table 4.14.2. Suppose each training record has four features and one target or the response variable, as shown in Fig. 4.14.1. The machine learning algorithm is used to predict the target variable.
- In classification task the target variable takes a discrete value, and in the task of regression its value could be continuous.
- In a training dataset we have the value of target variable. The relationship that exists between the features and the target variable is used by machine for learning.
- The target variable is the evaluation of the car. Classes are the target variables in the classification task. In classification systems it is assumed that classes are to be of limited number.
- Attributes or features are the individual values that, when combined with other features, make up a training example. This is usually columns in a training or test set.
- A training dataset and a testing dataset, is used to test machine learning algorithms. First the training dataset is given as input to the program. Program uses this data to learn. Next, the test set is given to the program.
- The program decides which instance of test data belongs to which class. The predicted output is compared with the actual output of the program, and we can get an idea about the accuracy of the algorithm. There are best ways to use all the information in the training dataset and test dataset.

- Assume in car evaluation classification system, we have tested the program and it meets the desired level of accuracy.
- Knowledge representation is used to check what the machine has learned. There are many ways in which knowledge can be represented.
- We can use set of rules or a probability distribution to represent the knowledge.
- Many algorithms represent the knowledge which is more interpretable to humans than others. In some situations we may not want to build an expert system but we are interested only in the knowledge representation that's acquired from training a machine learning algorithm.

Table 4.14.2 : Car evaluation classification based on four features

Buying_Price	Maintenance_ Price	Lug_Boot	Safety	Evaluation?
High	High	Small	High	Unacceptable
High	High	Small	Low	Unacceptable
Medium	High	Small	High	Acceptable
Low	Medium	Small	High	Acceptable
Low	Low	Big	High	Acceptable
Low	Low	Big	Low	Unacceptable
Medium	Low	Big	Low	Acceptable
High	Medium	Small	High	Unacceptable
High	Low	Big	High	Acceptable
Low	Medium	Big	High	Acceptable
High	Medium	Big	Low	Acceptable
Medium	Medium	Small	Low	Acceptable
Medium	High	Big	High	Acceptable
Low	Medium	Small	Low	Unacceptable

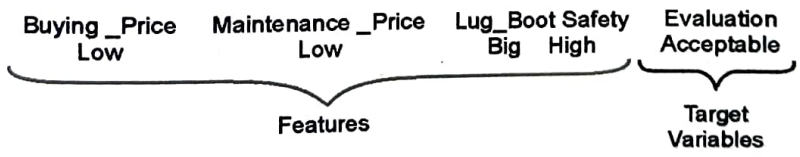

Fig. 4.14.1 : Features and target variable identified

(4) K-fold Cross validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
- The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.

- When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.
- Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data.
- That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
- It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.
- The general procedure is as follows :
  (1) Shuffle the dataset randomly.
  (2) Split the dataset into k groups
  (3) For each unique group:
    (i) Take the group as a hold out or test data set
    (ii) Take the remaining groups as a training data set
    (iii) Fit a model on the training set and evaluate it on the test set
    (iv) Retain the evaluation score and discard the model
  (4) Summarize the skill of the model using the sample of model evaluation scores
- Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.
- The k value must be chosen carefully for your data sample. Three common tactics for choosing a value for k are as follows:
- **Representative :** The value for k is chosen such that each train/test group of data samples is large enough to be statistically representative of the broader dataset.
- **k=10 :** The value for k is fixed to '10', a value that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.
- **k=n :** The value for k is fixed to' n', where n is the size of the dataset to give each test sample an opportunity to be used in the hold out dataset. This approach is called leave-one-out cross-validation.

Step 5 : Model Evaluation

- We have to check the machine designed against our evaluation data set that contains inputs that the model does not know and verify the precision of our already trained model.

- If the accuracy is less than or equal to 50%, that model will not be useful since it would be like tossing a coin to make decisions. If you reach 90% or more, you can have good confidence in the results that the model gives you.

(5) Accuracy

- The formula for accuracy is as follows.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

- When dealing with classification problems we are attempting to predict a binary outcome. Is it fraud or not? Will this person default on their loan or not? Etc.
- So what we care about in addition to this overall ratio is number predictions that were falsely classified positive and falsely classified negative, especially given the context of what we are trying to predict.
- A 99% accuracy rate might be pretty good if we are trying to predict something like credit card fraud, but what if a false negative represents someone who has a serious virus that is apt to spreading quickly?
- Or a person who has cancer? That's why we have to breakdown the accuracy formula even further.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positive, TN = True Negatives,
FP = False Positives and FN = False Negatives.

- A **True Positive** is an outcome where the model correctly predicts the positive class.
- A **True Negative** is an outcome where the model correctly predicts the negative class.
- A **False Positive** is an outcome where the model incorrectly predicts the positive class.
- A **False Negative** is an outcome where the model incorrectly predicts the negative class.

(6) Precision

- Precision talks about how precise/accurate your model is out of those predicted positive, how many of them are actual positive.
- Precision is a good measure to determine, when the costs of False Positive is high. For instance, email spam detection.
- In email spam detection, a false positive means that an email that is non-spam (actual negative) has been identified as spam (predicted spam). The email user might lose important emails if the precision is not high for the spam detection model.

$$\text{Precision} = \frac{TP}{TP + FP}$$

(7) Recall

- Recall actually calculates how many of the Actual Positives our model capture through labelling i as Positive (True Positive).
- Applying the same understanding, we know that Recall shall be the model metric we use to selec our best model when there is a high cost associated with False Negative.
- For instance, in fraud detection or sick patient detection. If a fraudulent transaction (Actua Positive) is predicted as non-fraudulent (Predicted Negative), the consequence can be very bad fo the bank.
- Similarly, in sick patient detection. If a sick patient (Actual Positive) goes through the test and predicted as not sick (Predicted Negative). The cost associated with False Negative will be extremely high if the sickness is contagious.

$$\text{Recall} = \frac{TP}{TP + Fn}$$

(8) Confusion matrix

- A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known.
- The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes) :

N = 165	Predicted : NO	Predicted : YES
Actual : NO	50	10
Actual : YES	5	100

What can we learn from this matrix ?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.

In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates)

- **True Positives (TP) :** These are cases in which we predicted yes (they have the disease), and they do have the disease.

- **True Negatives (TN) :** We predicted no, and they don't have the disease.
- **False Positives (FP) :** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **False Negatives (FN) :** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")
- I've added these terms to the confusion matrix, and also added the row and column totals :

N = 165	Predicted : NO	Predicted : YES	
Actual : NO	TN = 50	FP = 10	60
Actual : YES	FN = 5	TP = 100	105
	55	110	

- This is a list of rates that are often computed from a confusion matrix for a binary classifier:
- **Accuracy :** Overall, how often is the classifier correct?
  - (TP + TN) / total = (100 + 50) / 165 = 0.91
- **Misclassification Rate :** Overall, how often is it wrong?
  - (FP + FN) / total = (10 + 5) / 165 = 0.09
  - equivalent to 1 minus Accuracy
  - also known as "Error Rate"
- **True Positive Rate :** When it's actually yes, how often does it predict yes?
  - TP / actual yes = 100 / 105 = 0.95
  - also known as "Sensitivity" or "Recall"
- **False Positive Rate :** When it's actually no, how often does it predict yes?
  - FP / actual no = 10 / 60 = 0.17
- **True Negative Rate :** When it's actually no, how often does it predict no?
  - TN / actual no = 50 / 60 = 0.83
  - equivalent to 1 minus False Positive Rate
  - also known as "Specificity"
- **Precision :** When it predicts yes, how often is it correct?
  - TP / predicted yes = 100 / 110 = 0.91
- **Prevalence :** How often does the yes condition actually occur in our sample?
  - actual yes / total = 105 / 165 = 0.64

Step 6 : Hyper Parameter Tuning

- If during the evaluation you did not obtain good predictions and your precision is not the minimum desired, it is possible that you have overfitting -or underfitting problems and you must return to the training step before making a new configuration of parameters in your model.
- You can increase the number of times you iterate your training data- termed epochs. Another important parameter is the one known as the "learning rate", which is usually a value that

multiplies the gradient to gradually bring it closer to the global -or local- minimum to minimize the cost of the function.

- Increasing your values by 0.1 units from 0.001 is not the same as this can significantly affect the model execution time. You can also indicate the maximum error allowed for your model. You can go from taking a few minutes to hours, and even days, to train your machine. These parameters are often called Hyperparameters.
- This "tuning" is still more of an art than a science and will improve as you experiment. There are usually many parameters to adjust and when combined they can trigger all your options. Each algorithm has its own parameters to adjust.
- To name a few more, in Artificial Neural Networks (ANNs) you must define in its architecture the number of hidden layers it will have and gradually test with more or less and with how many neurons each layer. This will be a work of great effort and patience to give good results.

Step 7 : Prediction

- You are now ready to use your Machine Learning model inferring results in real-life scenarios.
- In this step a real program is developed to do some task, and once again it is checked if all the previous steps worked as you expected.
- You might encounter some new data and have to revisit step 1-6.

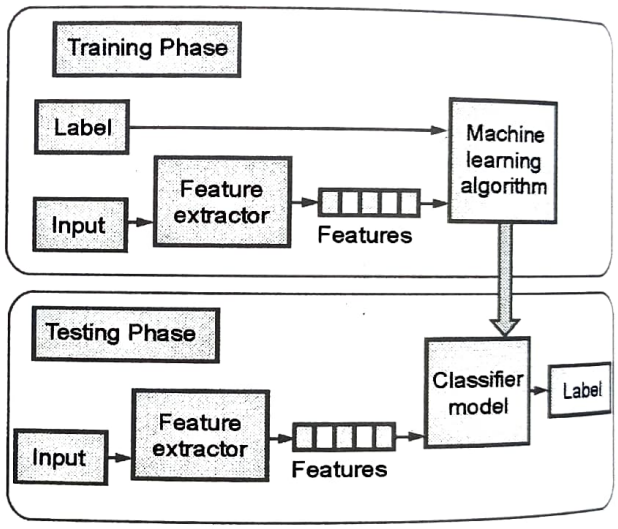

Fig. 4.14.2 : Typical example of Machine Learning Application

4.15 MODEL EVALUATION AND SELECTION

- Once our classification model is ready, we would like an estimate of how accurately the classifier can predict/classify the output class.
- Based on this, we will come to know whether training done is sufficient or not. We can even think of building more than one classifier and then compare their accuracy.
- Let's see now, what is accuracy? How can we estimate it? Are some measures of a classifier's accuracy more appropriate than others? How can we obtain a reliable accuracy estimate?

4.15.1 Metrics for Evaluating Classifier Performance

GQ. Mention various metrics for evaluating classifier performance

- The following list depicts various metrics/measures of evaluating how "accurate" your classifier is at predicting the class label of tuples :

(1) Accuracy (2) Error rate
(3) Precision (4) Sensitivity (Recall)
(5) Specificity (6) F1 score
(7) AOC-ROC (8) Log Loss

- Before discussing these measures, we need to understand with certain terminologies related to Confusion matrix.
- It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes.
- **A confusion matrix** is nothing but a table with two dimensions used for analyzing how well your classifier can recognize tuples of different classes viz. "Actual" and "Predicted" and furthermore, both the dimensions have "True Positives (TP)", "True Negatives (TN)", "False Positives (FP)", "False Negatives (FN)" as shown below:

		Actual Class 1	Actual Class 0	Total
Predicted Class	1	Ture Positives (TP)	False Positives (FP)	P
	0	False Negatives (FN)	True Negatives (TN)	N
		P'	N'	P + N

- **True Positives (TP)** : These refers to the positive tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 1.
- **True Negatives (TN)** : These are the negative tuples that were correctly labeled by the classifier. It is the case when both actual class and predicted class of data point is 0.
- **False Positives (FP)** : These are the negative tuples that were incorrectly labeled as positive. It is the case when actual class of data point is 0 and predicted class of data point is 1.
- **False Negatives (FN)** : These are the positive tuples that were mislabeled as negative. It is the case when actual class of data point is 1 and predicted class of data point is 0.
- Here, TP and TN tell us when the classifier is getting things right, while FP and FN tell us when the classifier is getting things wrong. Now let's understand various evaluation measures.

1. **Accuracy (Recognition rate)**
   - The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier.
   - In other words, Accuracy is the ratio of the number of correct predictions and the total number of predictions. The formula for Accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{P + N}$$

- Accuracy is useful when the target class is well balanced but is not a good choice for the unbalanced classes.
- Whereas other measures, such as sensitivity (or recall), specificity, precision, F, and Fβ, are better suited to the class imbalance problem, where the main class of interest is rare.
- In reality, data is always imbalanced for example Spam email, credit card fraud, and medical diagnosis.
- Hence, if we want to do a better model evaluation and have a full picture of the model evaluation, other metrics such as recall and precision should also be considered.

2. Error/Misclassification rate

- Error rate for a classifier, M, is simply 1 – accuracy(M), where accuracy(M) is the accuracy of the model M.
- We can also write this as :

$$\text{Error rate} = \frac{FP + FN}{P + N}$$

3. Recall/Sensitivity

- Recall is a measure of completeness i.e., what percentage of positive tuples are labeled as such.
- Sensitivity refers to True Positive (recognition) rate which is the proportion of positive tuples that are correctly identified.
- Recall and Sensitivity are similar

$$\text{Recall/Sensitivity} = \frac{TP}{TP + FN} = \frac{TP}{P}$$

4. Specificity

Specificity is the true negative rate which is the proportion of negative tuples that are correctly identified.

$$\text{Specificity} = \frac{TN}{N}$$

5. Precision

Precision is a measure of exactness i.e., what percentage of tuples labeled as positive are actually such.

$$\text{Precision} = \frac{TP}{TP + FP}$$

6. F-Score

- Precision and Recall are combined into a single measure to form F measures (also known as the F1 score or F-score).

- F1 Score is the harmonic mean of precision and recall. It is maximum when Precision is equal to Recall.

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

$$F\beta = \frac{(1 + \beta 2) * \text{Precision} * \text{Recall}}{\beta 2\ \text{Precision} + \text{Recall}}$$

7. AUC-ROC

- The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes.
- And, The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separates the 'signal' from the 'noise'.
- In a ROC curve, the X-axis value shows False Positive Rate/Recall and Y-axis shows True Positive Rate/Sensitivity. Higher the value of X means higher the number of False Positives (FP) than True Negatives (TN), while a higher Y-axis value indicates a higher number of TP than FN. So, the choice of the threshold depends on the ability to balance between FP and FN.
- From the graph shown in Fig. 4.16.1, the greater the AUC, the better is the performance of the model at different threshold points between positive and negative classes.
- This simply means that when AUC is equal to 1, the classifier is able to perfectly distinguish between all Positive and Negative class points.
- When AUC is equal to 0, the classifier would be predicting all Negatives as Positives and vice versa. When AUC is 0.5, the classifier is not able to distinguish between the Positive and Negative classes.

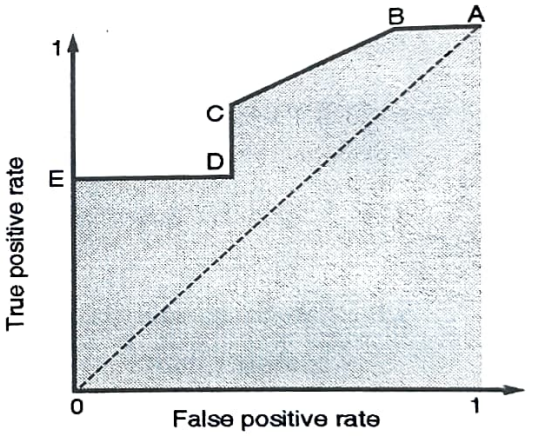

Fig. 4.15.1 : ROC Curve

8. Logarithmic Loss (Log Loss)

- It is also called Logistic regression loss or cross-entropy loss.
- It basically defined on probability estimates and measures the performance of a classification model where the input is a probability value between 0 and 1.
- It can be understood more clearly by differentiating it with accuracy.
- As we know that accuracy is the count of predictions (predicted value = actual value) in our model whereas Log Loss is the amount of uncertainty of our prediction based on how much it varies from the actual label.
- With the help of Log Loss value, we can have more accurate view of the performance of our model.

- Let's calculate these metrics for some reasonable real-world numbers. If we have 100,000 patients, of which 200 (20%) actually have cancer, we might see the following test results:

	Test Positive	Test Negative	Total
Patient Diseased	160	40	200
Patient Healthy	29940	69860	99800
Total	30100	69900	100000

For this data :

$$\text{Sensitivity} = TP/(TP + FN) = 160 / (160 + 40) = 80.0\%$$

$$\text{Specificity} = TN/(TN + FP) = 69,860 / (69,860 + 29,940)$$

$$= 70.0\%$$

- In other words, our test will correctly identify 80% of people with the disease, but 30% of healthy people will incorrectly test positive.
- By only considering the sensitivity (or accuracy) of the test, potentially important information is lost.
- However, classifiers can also be compared with respect to the following additional aspects, in addition to accuracy-based measures :

(1) **Speed :** Speed refers to the computational costs involved in creating and using the given classifier.

(2) **Robustness :** Robustness is the ability of the classifier to make correct predictions when the dataset contains noisy data or data with missing values. Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise and missing values.

(3) **Scalability :** Scalability refers to the ability to construct the classifier efficiently given large amounts of data. Scalability is typically assessed with a series of data sets of increasing size.

(4) **Interpretability :** Interpretability refers to the level of understanding and insight that is provided by the classifier. Interpretability is subjective and therefore it is difficult to assess.

4.16 CROSS-VALIDATION IN MACHINE LEARNING

GQ. What is cross validation in machine learning ?

GQ. Explain methods for cross-validation

- **Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. It is a technique how a statistical model generalises to an independent dataset.**

- In **machine learning**, we need to test the stability of the model. It means based only on the training dataset, we cannot fit our model on the training dataset.
- For this purpose, we test our model on the sample which is not part of the training dataset. And then we deploy the model on that sample.
- This complete process comes under cross-validation.
- The basic steps of cross-validation are :
  - (i) Reserve a subset of the dataset as a validation set.
  - (ii) Provide the training to the model using the training dataset.
  - (iii) Evaluate model performance using the validation set.
- If the model performs well with the validation set, perform the further step, else check for the issues.

4.16.1 Methods used for Cross-Validation

The common methods used for cross-validation are :

(1) Validation set approach
(2) Leave-P-out cross-validation
(3) Leave one out cross- validation
(4) K-fold cross-validation
(5) Stratified K-fold cross-validation

Among these, K-fold cross-validation is easy to understand, and the output is less biased than other methods.

4.16.2 K-Fold Cross-Validation

- In each set (fold) training and the test would be performed precisely once during this entire process. It helps us to avoid overfitting. When a model is trained using all of the data in a single short and give the best performance accuracy.
- This K-fold cross-validation helps us to build the model which is a generalized one.
- To achieve this K-fold cross validation, we have to split the data set into three sets, training, testing and validation, with the challenge of the volume of the data.
- Here test and train data set will support building model and hyper parameter assessment.
- The model is validated multiple times based on the value assigned as a parameter and which is called K and it should be an **INTEGER**.
- The dataset X is divided randomly into K equal-sized parts, X_i, i = 1, 2, ..., K.
- To generate each pair, we keep one of the K parts out as validation set and combine the remaining (K – 1) parts to form the training set.
- Doing this K times, each time leaving out another one of the K parts out, we get K pairs :

$$V_1 = X_1, T_1 = X_2 \cup X_3 \cup ... \cup X_K$$

$$V_2 = X_2, T_2 = X_1 \cup X_3 \cup \ldots \cup X_K$$

$$\vdots$$

$$V_K = X_K, T_K = X_1 \cup X_2 \cup \ldots \cup X_{K-1}$$

- We come across two problems with this. First, to keep the training set large, we allow validation sets that are small.
- Second, the training sets overlap considerably namely, any two training sets share (K – 2) parts.
- K is typically 10 or 30. As K increases, the percentage of training instances increases and we get more robust estimators, but the validation set becomes smaller.
- Also, there is the cost of training the classifier K times, which increases as K is increased.
- As N increases, K can be smaller, if N is small, K should be large to allow large large enough training sets. (N is the number of instances).
- One extreme case of K-fold cross-validation is **leave-one-out** where given a dataset of N instances, only one instance is left out as the validation set (instance) and training uses (N – 1) instances.
- We then get N separate parts by leaving out a different instance at each iteration. This is typically used in applications such as medical diagnosis.

4.16.3 Life Cycle of K-fold Cross-Validation

- Let us have a generalised K-value. If K = 5, it means, we are splitting the given dataset into 5 folds and running the Train and Test.
- During each run, one fold is considered for testing and the rest will be for training and moving on with iterations, the below pictorial representation gives an idea of the flow of the fold-defined size

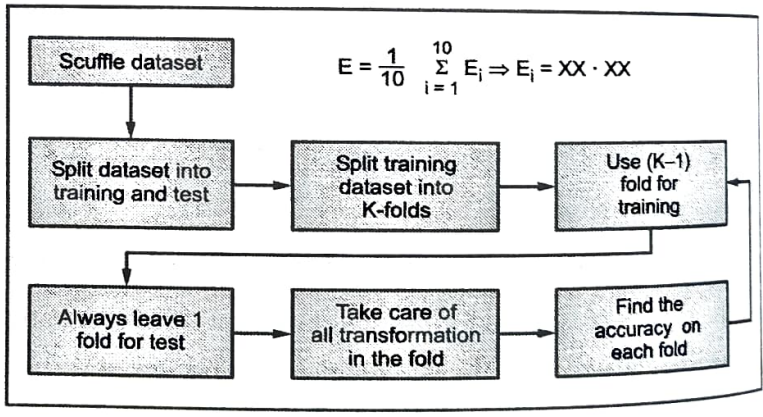

Fig. 4.16.1

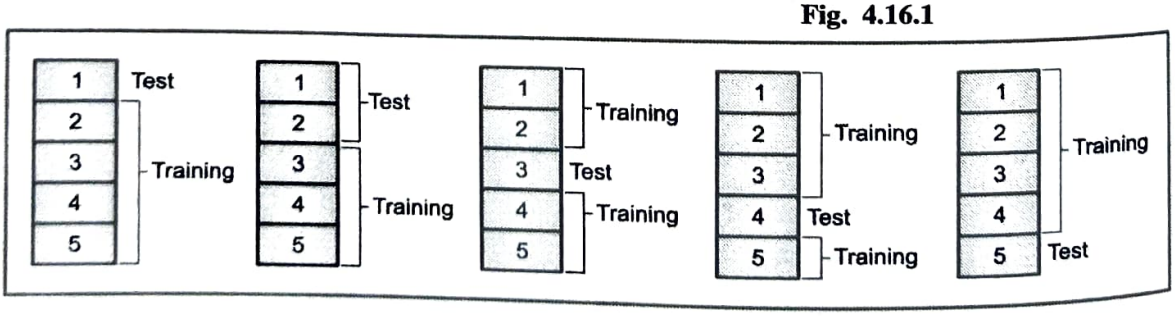

Fig. 4.16.2

- Here, each data-points is used, once in the hold-out set and K – 1 in Training. So during the full iteration at least once, one fold will be used for testing and the rest for training. In the above set, 5-Testing 20 Training.
- In each iteration, we will get an accuracy score and have to sum them and find the mean.
- Here we can understand how the data is spread in a way of consistency and will make a conclusion whether to go for the production with this model (or) NOT.

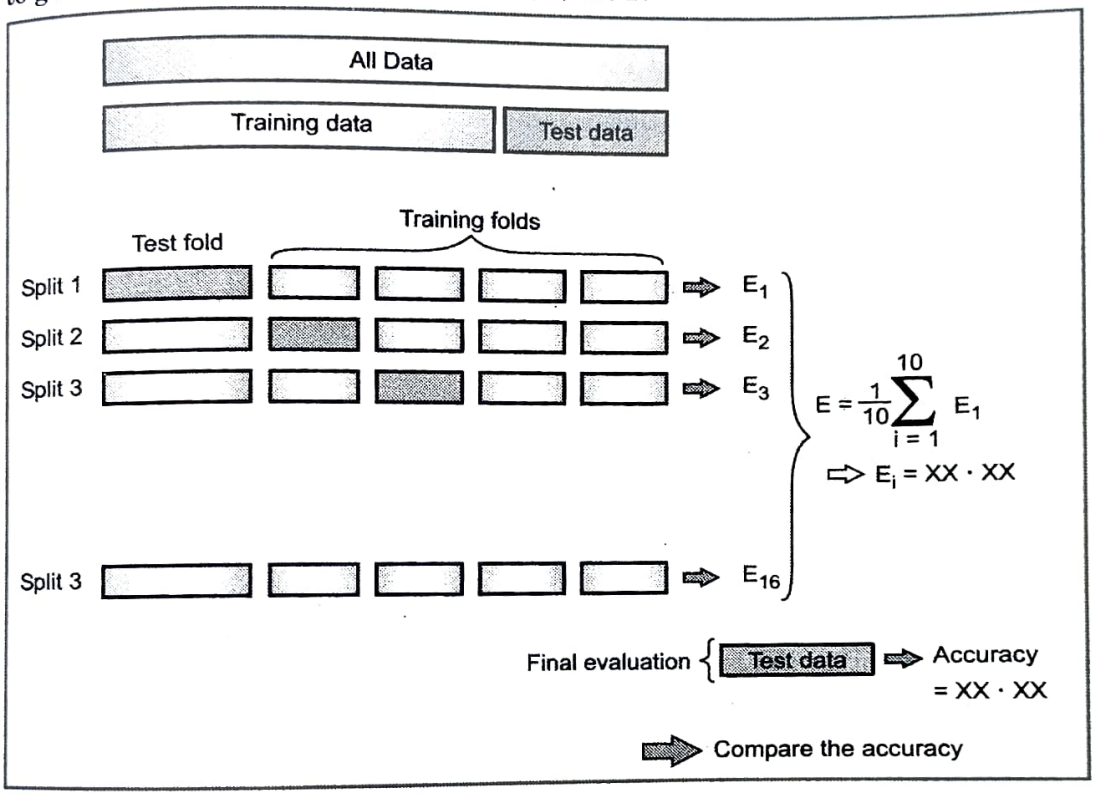

Fig. 4.16.3

4.16.4 Thumb Rules Associated with K-Fold

We discuss a few thumb-rules while dealing with K-fold.

(i) K should be always ≥ 2 and equal to number of records.

If 2, then just 2 iterations

If K = No of records in the dataset, then 1 for testing and n-for training.

(ii) The optimised value for the K is 10 and used with the data of good size, (i.e. commonly used).

(iii) If the K-value is large, then this will lead to less variance across the training set and limit the model currency, difference across the iterations.

(iv) The number of folds is generally inversely proportional to the size of the data set, which means, if the dataset size is too small, the number of folds can increase.

(v) Larger values of K eventually increase the running time of the cross-validation process.

4.16.5 Some Remarks

(1) Short note on K-cross Validation

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called K that refers to the number of groups that a given data sample is to be split into. Hence, the procedure is often called K-fold cross-validation.

(2) Purpose of K-fold Cross-Validation

K-fold cross-validation is one method that attempts to maximise the use of the available data for training and then testing a model. It is particularly useful for assessing model performance, as it provides a range of accuracy scores across (somewhat) different data sets.

(3) Folds in K-fold Cross-Validation

The key configuration parameter for K-fold cross-validation is K that defines the number of folds in which to split a given dataset.

Common values are K = 3, K= 5 and K = 10, and the most popular value used in applied machine learning to evaluate models is K = 10.

(4) Difference between K-fold and cross-validation

Cross-validation score is a function which evaluates a data and returns the score. On the other hand, K-fold is a class, which lets to split data to K-folds.

(5) Does K-fold cross-validation prevent over fitting ?

K-fold cross-validation won't reduce over fitting on its own, but using it will generally gives a better insight on the model, which can help to avoid or reduce over fitting.

4.17 MICRO-AVERAGE PRECISION, RECALL, F-SCORE

Micro-precision measures the precision of the aggregated contributions of all classes.

It is micro-averaged precision.

Precision = 1 $\Rightarrow$ the model's predictions are perfect, all samples classified as the positive class are truly positive.

4.17.1 Emphasis on Common Classes

Micro-averaging puts more emphasis on the common classes in the data set. This is preferred behaviour for multi-label classification labels.

Precision is a metric used in binary classification problems.

Precision is defined as :

$$\textbf{Precision} = \frac{\text{True positive}}{\text{True positive + False positive}}$$

Where **True positive** is when actual positive is predicted positive, and

False positive is when actual negative is predicted positive.

☞ **Micro-Averaging**

- Micro-averaging is used when a problem has more than 2 labels that can be true,
- Micro-averaging is performed by first calculating the sum of all true positives and false positives, over all the classes.
- Then we compute the precision for the sums
- Micro-precision valve can be high even if the model is performing very poorly on a **rare** class, since it gives more weight to the **common classes.**
- For single-label multi-class problems, micro averaging would result in precision, almost equal to accuracy. But that does not give additional information about the model's performance.

Example : Suppose there is a multi-class classification problem with 3 classes (A. B, C).

First we calculate how many true positive (TP) and False positive (FP), we have for each class :

A : 2 T P and 8 F P

B : 1 T P and 5 F P

C : 1 T P and 3 F P

Aggregate of all classes :

$(TP)_{sum}$: 2 + 1 + 1 = 4

$(FP)_{sum}$: 8 + 5 + 3 = 16

Now, precision of the aggregated values :

$$\text{Micro-precision} = \frac{(TP)_{sum}}{(TP)_{sum} + (FP)_{sum}} = \frac{4}{4+16} = \frac{4}{20} = 0.2$$

⏭ 4.18 WHAT IS RECALL ?

The recall is the ratio between the number of **positive samples** correctly classified as **positive** to the total number of **positive** samples.

The recall measures the model's ability to detect **positive samples.**

The higher the recall, the more positive samples detected.

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive + False negative}}$$

$$= \frac{TP}{TP + FN}$$

TP = True positive, FN = False negative

☞ **Remarks :**

(i) Recall is independent of the number of negative sample classifications.

(ii) If the model classifies all positive samples as positive, then Recall will be 1.

(iii) Recall of a machine learning will be low when the value of :

TP + FN (denominator) > TP (Numerator)

(iv) Recall will be high when value of : TP (Numerator) > TP + FN (denominator).

Example : Recall of four different cases where each has the same recall as 0.667 but differs in the classification of negative samples.

Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
X	✓	X	✓	✓	✓	✓	✓
X	✓	✓	✓	✓	✓	✓	✓
X	X	X	X	X	X	✓	X
← A →		← B →		← C →		← D →	

☞ **Here :**

(i) Case A has two negative samples classified as negative.

(ii) Case B has two negative sample classified as negative.

(iii) Case C has only one negative samples classified as negative.

(iv) Case D does not classify any negative sample as negative.

Since recall is independent of how the negative samples are classified in the model, hence we can neglect negative samples and only calculate all samples that are classified as positive.

Negative	Positive	Negative	Positive	Negative	Positive	Negative	Positive
–	✓	–	✓	–	✓	–	✓
–	✓	–	✓	–	✓	–	✓
–	x	–	X	–	X	–	X

Here there are two positive samples classified as positive while only 1 negative sample is correctly classified as negative.

$$\therefore \text{ Recall} = \frac{TP}{TP + FN} = \frac{2}{2+1} = \frac{2}{3} = 0.667$$

4.19 MICRO F1-SCORE

- Micro F-1 score (it is short form of micro-averaged F1 score) is used to assess the quality of multi-label binary problems.
- It measures the F1-score of the aggregated contributions of all classes.
- To select a model based on a balance between precision and recall, assessing F1-score is helpful.
- Micro F1-score = 1 is the best value (**perfect micro-precision** and micro-recall), and 0 is the worst value.
- Observe that precision and recall have the same relative contribution to F-1 score.

4.19.1 Emphasis on Common Labels

Micro-averaging will put more emphasis on the common labels in the dataset, because it gives each sample the Same importance.

This is the preferred behaviours for multi-class classification problems.

Labels that are very rare in the dataset, may not be influencing the overall F1-score if the model is performing well in the other more common genres.

Definition of Micro F1 :

Micro F1-score is defined as the harmonic mean of the precision and recall :

$$\text{Micro F1-score} = 2\left[\frac{(\text{Micro precision}) \cdot (\text{Micro-recall})}{(\text{micro precision}) + (\text{micro-recall})}\right]$$

4.19.2 Micro-averaging

Micro-averaging is used when a problem has 2 or more labels that can be true.

Micro-averaging F1-score is performed by first calculating the sum of all true positives, false positives, and false negatives over all the labels.

Then we compute micro-precision and micro-recall from the sums.

Finally we calculate the harmonic mean to get the micro F1-score.

4.19.3 Difference between Precision and Recall in Machine Learning

Sr. No.	Precision	Recall
1.	It helps to measure the ability to classify positive samples in the model.	It helps to measure how many positive samples were correctly classified by the ML model.
2.	While calculating the precision of a model, we have to consider both positive as well as negative samples that are classified.	While calculating the Recall of a model, we need consider all positive samples while all negative samples will be neglected.
3.	When the model classifies most of the positive samples correctly as well as many false positive samples, then the model is said to be a high recall and low precision model.	When a model classifies a sample as positive, but it can only classify a few positive samples, then the model is said to be high accuracy, high precision and low recall model.
4.	The precision of a machine learning model is dependent on both the negative and positive samples.	Recall of a machine learning model is dependent on positive samples and independent of negative samples.
5.	In precision we consider all positive samples that are classified as positive either correctly or incorrectly.	The recall cares about correctly classifying all positive samples. It does not consider if any negative sample is classified as positive.

4.19.4 The Need of Precision and Recall in Machine Learning Models

The use of precision and recall varies according to the type of problem being solved :

1. If there is a need of classifying all positive as well as negative samples as positive whether they are classified correctly or incorrectly, then we use precision.
2. But, if the aim is to detect only all positive samples, then use Recall. Here, we need not bother how negative samples are correctly or incorrectly classified the samples.

4.20 MACRO-AVERAGE, PRECISION, RECALL AND F-SCORE

4.20.1 Macro-Average Precision

- We calculate the precision for each class separately in an one vs All way. Then we take average of all precision values.
- For example, for 3 classes; a, b, c. We calculate p_a, p_b, p_c and **macro average** will be $\frac{p_a + p_b + p_c}{3}$.

4.20.2 Weighted Precision

- It is similar to macro precision, except that we take number of instances for each class into consideration as well.
- These acts as weights.
- For example, for 3 classes a, b, c; if number of instances are A, B, C, respectively, then the weighted precision is :

$$\frac{Ap_a + Bp_b + Cp_c}{A + B + C}$$

4.20.3 Macro-recall Precision

- Macro-recall measures the average recall per class. (It is macro-averaged recall).
- Macro-recall = 1 implies that the model's
- Predictions are perfect, all truly positive samples were predicted as the positive class.
- And all classes are treated equally.
- Macro-recall will be low for models that only perform well on the common classes but performing poorly on the rare classes.
- Recall is a metric used in binary classification problem.

Recall is defined as

$$\text{Recall} = \frac{\text{Truly positive}}{\text{Truly positive + False negative}}$$

$$= \frac{TP}{TP + FN}$$

Where truly positive is when actual positive is predicted positive as

False negative is when actual positive is predicted negative.

4.20.4 The Macro-averaged F1-Score

- The macro F1-score is computed using the arithmetic mean of all the per-class F1 scores.

  This method treats all classes equally regardless of their support values.
- Thus macro-averaging score is to be used when all classes need to be treated equally to evaluate the overall performance of the classifier with respect to the most frequent class-labels.
- The macro-average of F-score is the harmonic mean.

$$\text{F1-Score} = 2\left[\frac{(\text{Precision}) \times (\text{Recall})}{\text{Precision} + \text{Recall}}\right]$$

What is a good F-measure score :

F1	Interpretation
70.9	Very good
0.8 – 0.9	Good
0.5 – 0.8	O. K.
< 0.5	Not good

- Thus, F1-score range from 0 to 1, where 1 is a perfect score indicating that the model predicts each observation correctly.
- A good F1-score is dependent on the data you are working with and the use case.

Chapter Ends ...

❑❑❑