National University of Computer & Emerging Sciences, Karachi
Computer Science Department

Spring 2023, Lab Manual – 10

Course Code: AI-2002	**Course: Artificial Intelligence Lab**
Instructor(s):	**Kariz Kamal, Saeeda Kawal, Sania Urooj, Shakir Hussain, Omer Qureshi, Ali Fatmi**

Contents:

I. Supervised Learning
   - Linear Regression
   - Decision Trees
   - Entropy
   - K-Nearest Classifier
   - SVMs (Support Vector Machines)

II. Unsupervised Learning
   - K-Means Clustering
   - Anomaly Detection Algorithms
   - Generative Algorithms

III. Performance Measuring Tools
   - Confusion Matrix
   - ROC

1) Supervised Learning

Supervised learning and Unsupervised learning are two common types of machine learning techniques. Supervised Learning involves training a machine learning model on a labeled dataset, where the input data is paired with corresponding output data. The goal of the model is to learn a mapping function that can accurately predict the output for new, unseen input

data. Supervised learning is commonly used for tasks such as classification, where the goal is to assign input data to one of several categories, or regression, where the goal is to predict a continuous value.

Types of Supervised Learning Algorithms

1. Linear Regression

Linear regression is a statistical modeling technique used to establish the relationship between a dependent variable and one or more independent variables. The goal of linear regression is to find the linear relationship between the input variables (predictors or independent variables) and the output variable (response or dependent variable).

In simple linear regression, there is only one independent variable, and the linear relationship is defined by a straight line. In multiple linear regression, there are multiple independent variables, and the linear relationship is defined by a hyperplane.

The equation of a linear regression model is expressed as:

$y = b0 + b1x1 + b2x2 + ... + bn*xn$

where y is the dependent variable, x1, x2, ..., xn are the independent variables, b0 is the intercept or bias term, and b1, b2, ..., bn are the coefficients or weights of the independent variables. The goal of linear regression is to estimate the values of the coefficients that best fit the observed data, so that the model can be used to predict the value of the dependent variable for new values of the independent variables.

Key assumptions of effective linear regression

Assumptions to be considered for success with linear-regression analysis:

- **For each variable:** Consider the number of valid cases, mean and standard deviation.

- **For each model:** Consider regression coefficients, correlation matrix, part and partial correlations, multiple R, R2, adjusted R2, change in R2, standard error of the estimate, analysis-of-variance table, predicted values and residuals. Also, consider 95-percent-confidence intervals for each regression coefficient, variance-covariance matrix, variance inflation factor, tolerance, Durbin-Watson test, distance measures (Mahalanobis, Cook and leverage values), DfBeta, DfFit, prediction intervals and case-wise diagnostic information.
- **Plots:** Consider scatterplots, partial plots, histograms and normal probability plots.
- **Data:** Dependent and independent variables should be quantitative. Categorical variables, such as religion, major field of study or region of residence, need to be recoded to binary (dummy) variables or other types of contrast variables.
- **Other assumptions:** For each value of the independent variable, the distribution of the dependent variable must be normal. The variance of the distribution of the dependent variable should be constant for all values of the independent variable. The relationship between the dependent variable and each independent variable should be linear and all observations should be independent.

Least-Squares Regression

The most common method for fitting a regression line is the method of least-squares. This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line (if a point lies on the fitted line exactly, then its vertical deviation is 0). Because the deviations are first squared, then summed, there are no cancellations between positive and negative values.

Evaluation Metrics for Linear Regression The strength of any linear regression model can be assessed using various evaluation metrics. These evaluation metrics usually provide a measure of how well the observed outputs are being generated by the model. The most used metrics are, 1. Coefficient of Determination or R-Squared (R2) 2. Root Mean Squared Error (RSME) and Residual Standard Error (RSE)

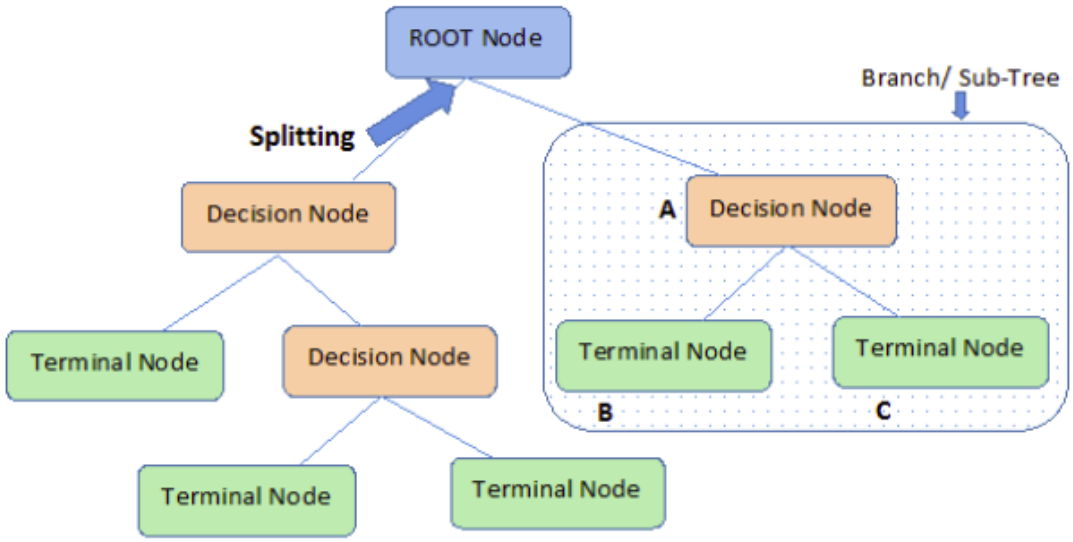
1. **Coefficient of Determination or R-Squared (R2):** R-Squared is a number that explains the amount of variation that is

explained/captured by the developed model. It always ranges between 0 & 1 . Overall, the higher the value of R-squared, the better the model fits the data. Mathematically it can be represented as, R 2 = 1 – (RSS/TSS) Residual sum of Squares (RSS) is defined as the sum of squares of the residual for each data point in the plot/data. It is the measure of the difference between the expected and the actual observed output. Total Sum of Squares (TSS) is defined as the sum of errors of the data points from the mean of the response variable.

2. **Root Mean Squared Error:** The Root Mean Squared Error is the square root of the variance of the residuals. It specifies the absolute fit of the model to the data i.e. how close the observed data points are to the predicted values.

2)Decision Tree

In general, Decision tree analysis is a predictive modeling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are one of the most powerful algorithms that falls under the category of supervised algorithm. Below is an image explaining the basic structure of the decision tree. Every tree has a root node, where the inputs are passed through. This root node is further divided into sets of decision nodes where results and observations are conditionally based. The process of dividing a single node into multiple nodes is called splitting. If a node doesn't split into further nodes, then it's called a leaf node, or terminal node. A subsection of a decision tree is called a branch or sub-tree (e.g. in the box in the image below).



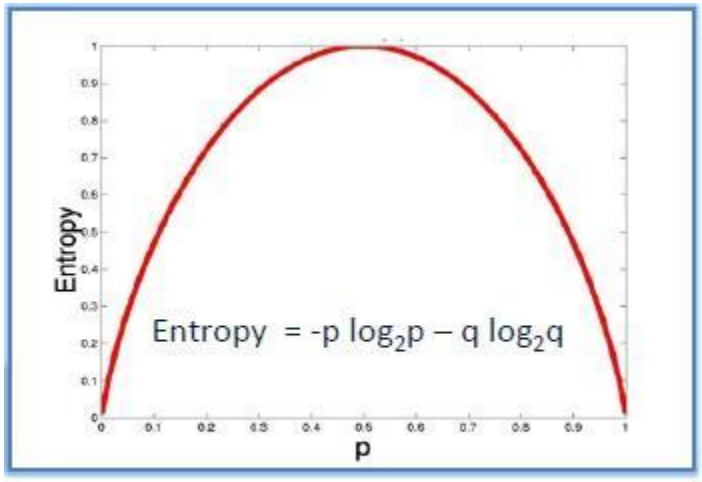
Note:- A is parent node of B and C.

Consider whether a dataset based on which we will determine whether to play football or not.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Entropy

In machine learning, entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.



Entropy = $-0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

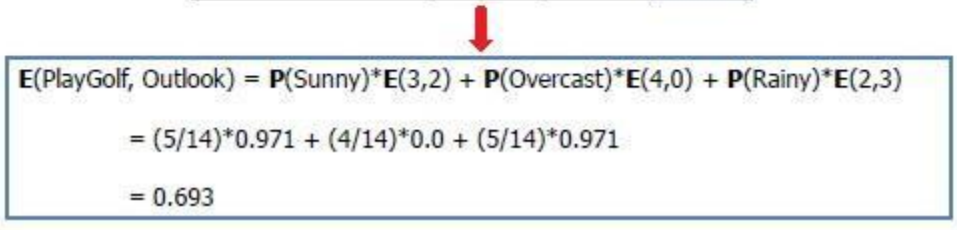
Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

E(S) = -[(9/14)log(9/14) + (5/14)log(5/14)] = 0.94

b) Entropy using the frequency table of two attributes:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



We have to calculated average weighted entropy. i.e, we have found the total of weights of each feature multiplied by probabilities.

E(S, outlook) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3) = (5/14)(-(3/5)log(3/5)-(2/5)log(2/5))+ (4/14)(0) + (5/14)((2/5)log(2/5)-(3/5)log(3/5)) = 0.693

Information Gain

Information gain can be defined as the amount of information gained about a random variable from observing another random variable. It can be considered as the difference between the entropy of parent node and weighted average entropy of child nodes. The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

Decision Tree Algorithm:

Step 1: Calculate entropy of the target.

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

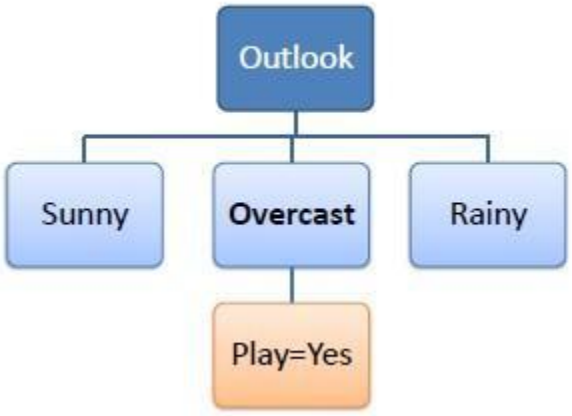
Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Step 4a: A branch with entropy of 0 is a leaf node.

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Step 4b: A branch with entropy more than 0 needs further splitting.

The next step is to find the next node in our decision tree. Now we will find one under sunny. We have to determine which of the following Temperature, Humidity or Wind has higher information gain.

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Calculate parent entropy E(sunny)

E(sunny) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.

Now Calculate the information gain of Temperature. IG(sunny, Temperature)

		play		
		yes	no	total
	hot	0	2	2
Temperature	cool	1	1	2
	mild	1	0	1
				5

E(sunny, Temperature) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4

Now calculate information gain.

IG(sunny, Temperature) = 0.971–0.4 =0.571

Similarly we get

IG(sunny, Humidity) = 0.971

IG(sunny, Windy) = 0.020

Here IG(sunny, Humidity) is the largest value. So Humidity is the node that comes under sunny.

	play	
Humidity	yes	no
high	0	3
normal	2	0

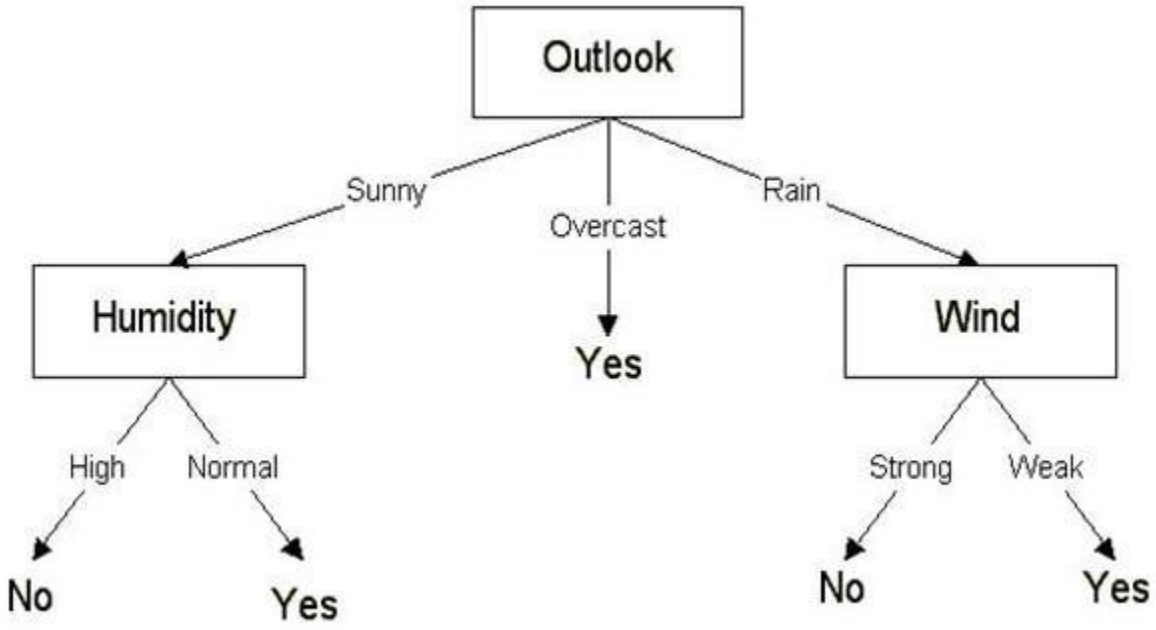
For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.

Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision Tree to Decision Rules

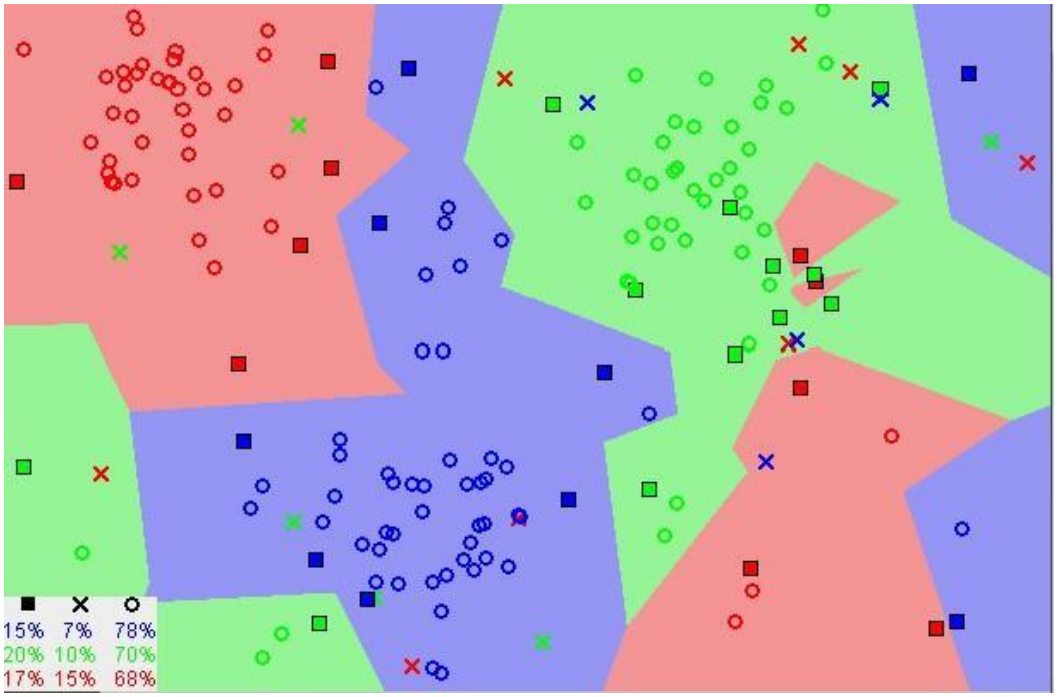
A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

Finally, our decision tree will look as below:



K-Nearest Classifier

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.



Notice in the image above that most of the time, similar data points are close to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) — calculating the distance between points on a graph (for example Euclidian distance).

How KNN algorithm works

Suppose we have height, weight and T-shirt size of some customers and we need to predict the T-shirt size of a new customer given only height and weight information we have. Data including height, weight and T-shirt size information is shown below –

Height (in cms)	Weight (in kgs)	T-Shirt Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L

Step 1: Calculate Similarity based on distance function

Euclidean :

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Manhattan / city - block :

$$d(x,y) = \sum_{i=1}^{m}|x_i - y_i|$$

The idea to use distance measure is to find the distance (similarity) between new sample and training cases and then finds the k-closest customers to new customer in terms of height and weight.

New customer has height 161cm and weight 61kg.

Euclidean distance between first observation and new observation is as follows –

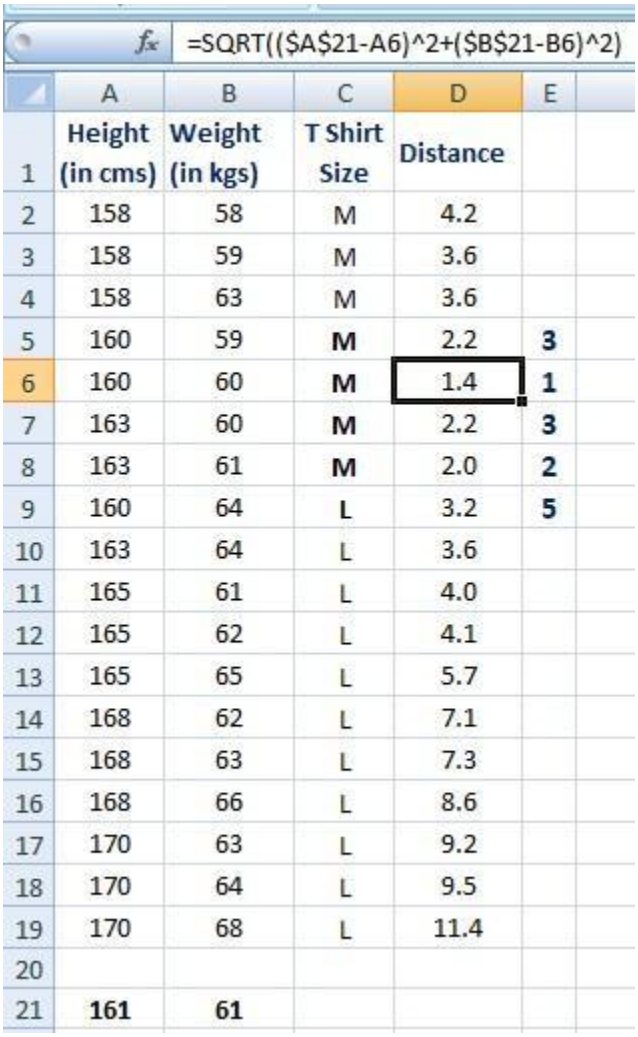
=SQRT((161-158)^2+(61-58)^2)

Similarly, we will calculate distance of all the training cases with new case and calculates the rank in terms of distance. The smallest distance value will be ranked 1 and considered as nearest neighbor.

Step 2: Find K-Nearest Neighbors

Let k be 5. Then the algorithm searches for the 5 customers closest to Monica, i.e. most similar to Monica in terms of attributes, and see what categories those 5 customers were in.

If 4 of them had ‘Medium T shirt sizes’ and 1 had ‘Large T shirt size’ then your best guess for Monica is ‘Medium T shirt. See the calculation shown in the snapshot below –



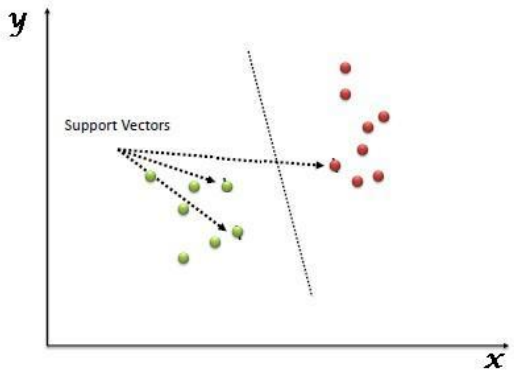
=SQRT((A21-A6)^2+(B21-B6)^2)

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	**M**	2.2	**3**
6	160	60	**M**	1.4	**1**
7	163	60	**M**	2.2	**3**
8	163	61	**M**	2.0	**2**
9	160	64	**L**	3.2	**5**
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	**161**	**61**			

SVM:

Support Vector Machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data

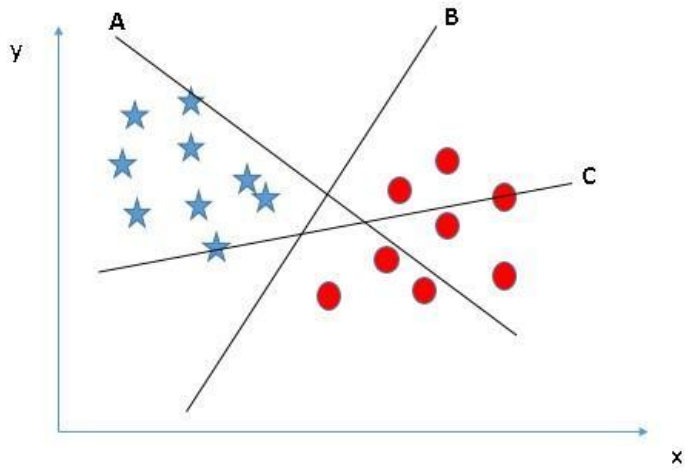
item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



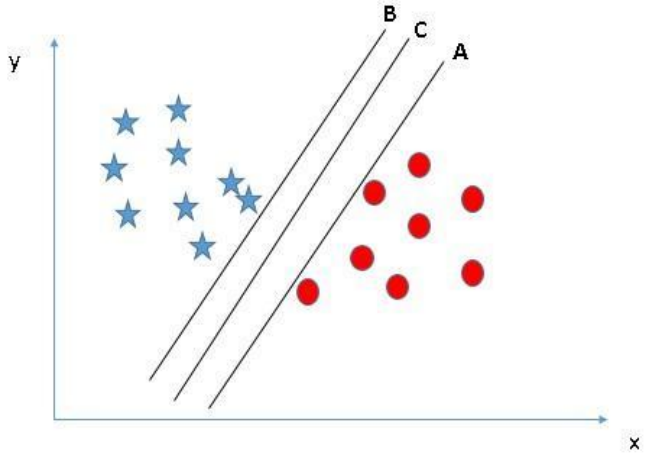
Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

How does SVM work

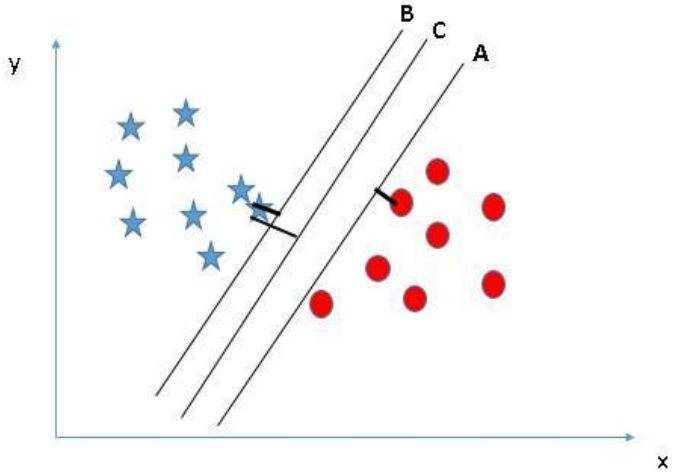
- **Identify the right hyper-plane (Scenario-1):** Here, we have three hyper-planes (A, B, and C). Now, identify the right hyper-plane to classify stars and circles.



- You need to remember a thumb rule to identify the right hyper-plane: “Select the hyperplane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.
- **Identify the right hyper-plane (Scenario-2):** Here, we have three hyper-planes (A, B, and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

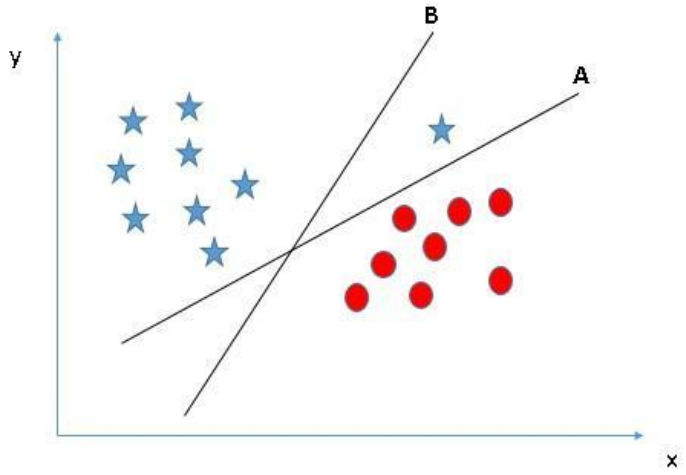


Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.



Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

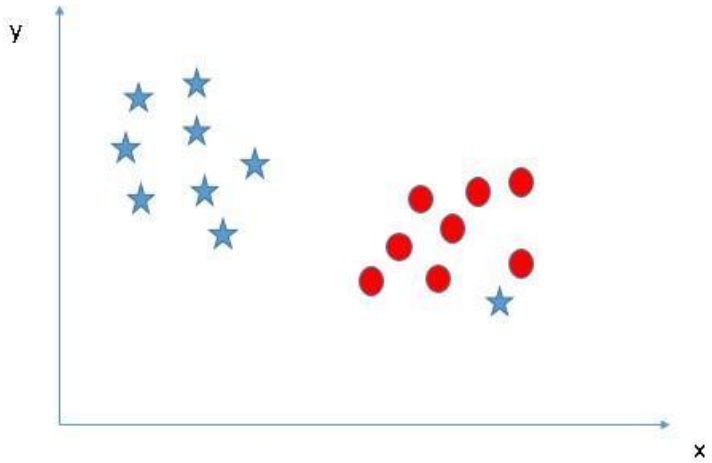
- **Identify the right hyper-plane (Scenario-3):**



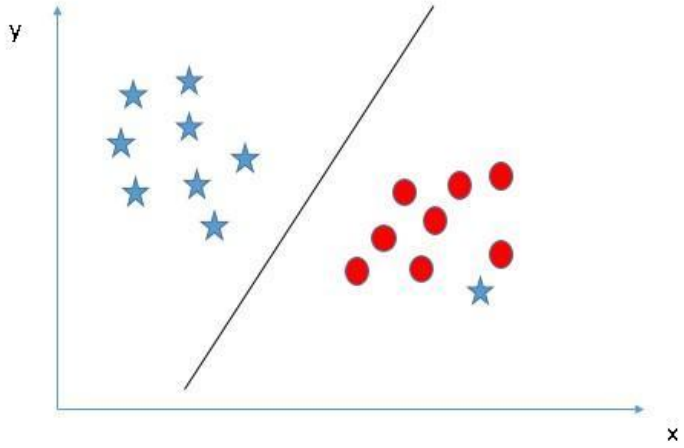
SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly.

Therefore, the right hyper-plane is **A.**

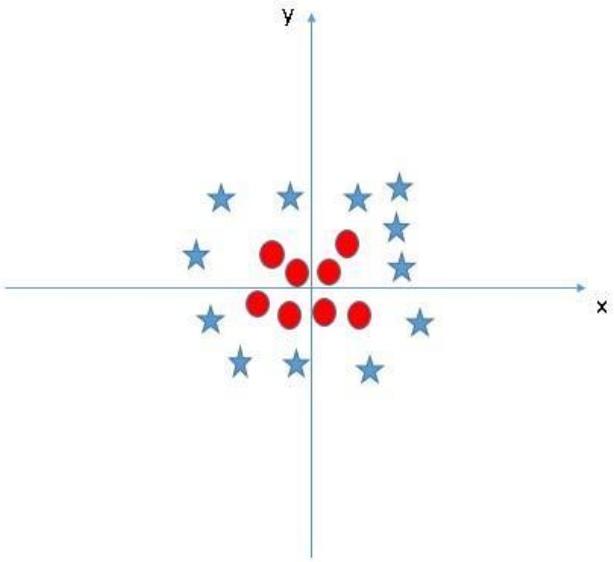
- **Can we classify two classes (Scenario-4)?:** Below, we are unable to segregate the two classes using a straight line, as one of the stars lies in the territory of other(circle) class as an outlier.



- One star at other end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.

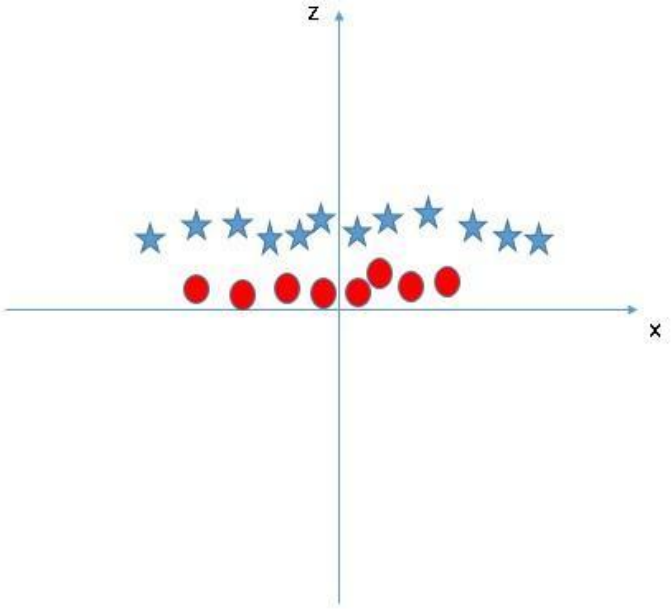


- **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature.

Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

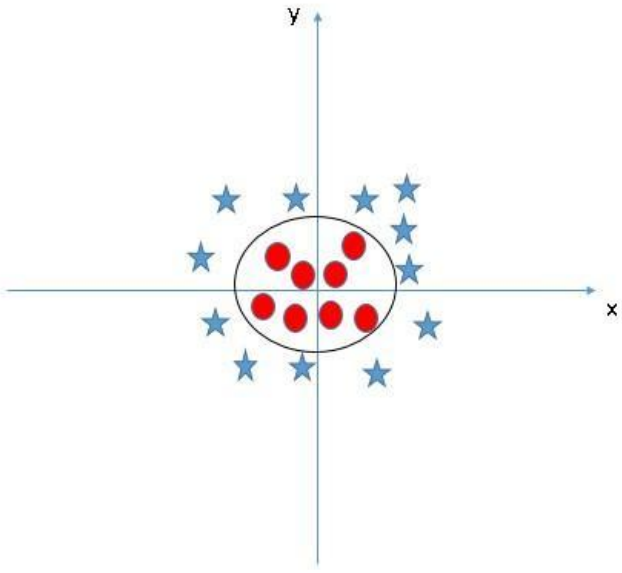


In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, should we need to add this feature manually to have a hyper-plane? No, the

SVM algorithm has a technique called the kernel **trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e; it converts not separable problem to separable problem. It

is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:



3)Unsupervised Learning

Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabeled data.

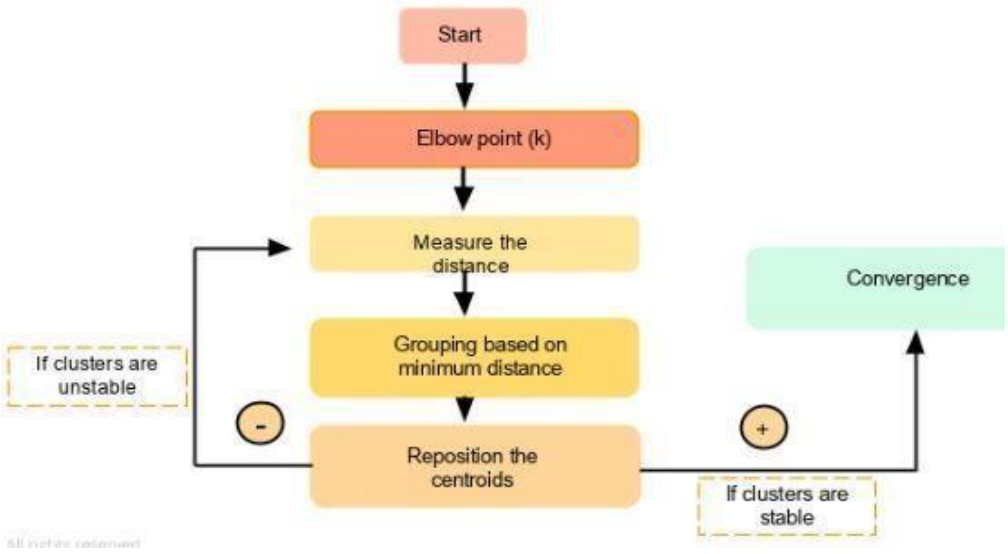
K-Means Clustering

K-Means clustering is an unsupervised learning algorithm. There is no labeled data for this clustering, unlike in supervised learning. K-Means performs the division of objects into clusters that share similarities and are dissimilar to the objects belonging to another cluster. The term 'K' is a number. You need to tell the system how many clusters you

need to create. For example, K = 2 refers to two clusters. There is a way of finding out what is the best or optimum value of K for a given data.

How Does K-Means Clustering Work:

The flowchart below shows how k-means clustering works:



The goal of the K-Means algorithm is to find clusters in the given input data. There are a couple of ways to accomplish this. We can use the trial and error method by specifying the value of K (e.g., 3,4, 5). As we progress, we keep changing the value until we get the best clusters.

Another method is to use the Elbow technique to determine the value of K. Once we get the K's value, the system will assign that many centroids randomly and measure the distance of each of the data points from these centroids. Accordingly, it assigns those points to the corresponding centroid from which the distance is minimum. So each data point will be assigned to the centroid, which is closest to it. Thereby we have a K number of initial clusters. For the newly formed clusters, it calculates the new centroid position.

The position of the centroid moves compared to the randomly allocated one.

Once again, the distance of each point is measured from this new centroid point. If required, the data points are relocated to the new centroids, and the mean position or the new centroid is calculated once again.

If the centroid moves, the iteration continues indicating no convergence. But once the centroid stops moving (which means that the clustering process has converged), it will reflect the result. Let's use a visualization example to understand this better.

We have a data set for a grocery shop, and we want to find out how many clusters this has to be spread across. To find the optimum number of clusters, we break it down into the following steps:

Step 1:

The Elbow method is the best way to find the number of clusters. The elbow method constitutes running K-Means clustering on the dataset.

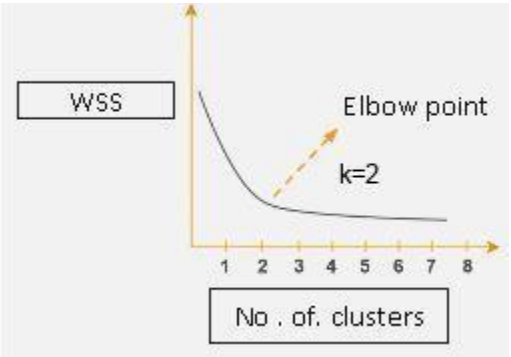
Next, we use within-sum-of-squares as a measure to find the optimum number of clusters that can be formed for a given data set. Within the sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where x_i = data point and c_i = closest point to centroid

The WSS is measured for each value of K. The value of K, which has the least amount of WSS, is taken as the optimum value.

Now, we draw a curve between WSS and the number of clusters.



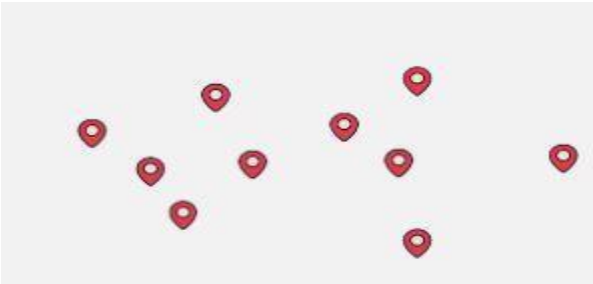
Here, WSS is on the y-axis and number of clusters on the x-axis.

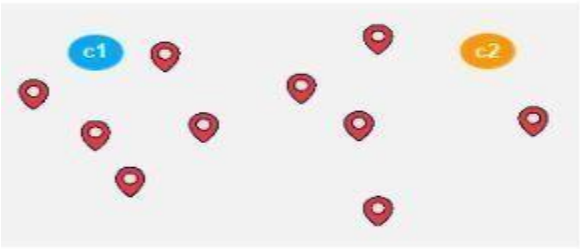
You can see that there is a very gradual change in the value of WSS as the K value increases from 2.

So, you can take the elbow point value as the optimal value of K. It should be either two, three, or at most four. But, beyond that, increasing the number of clusters does not dramatically change the value in WSS, it gets stabilized.

Step 2:

Let's assume that these are our delivery points:



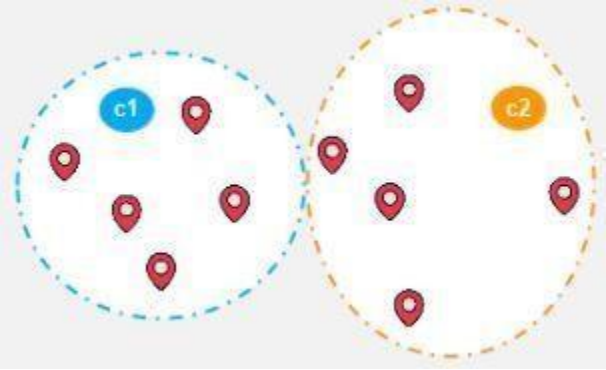


We can randomly initialize two points called the cluster centroids.

Here, C1 and C2 are the centroids assigned randomly.

Step 3:

Now the distance of each location from the centroid is measured, and each data point is assigned to the centroid, which is closest to it. This is how the initial grouping is done:



Step 4:

Compute the actual centroid of data points for the first group.

Step 5:

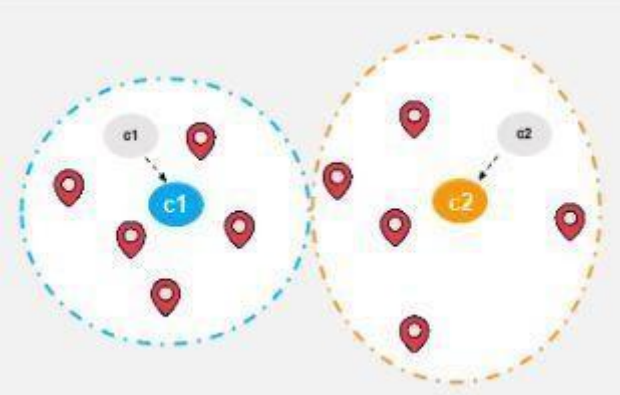
Reposition the random centroid to the actual centroid.

Step 6:

Compute the actual centroid of data points for the second group.

Step 7:

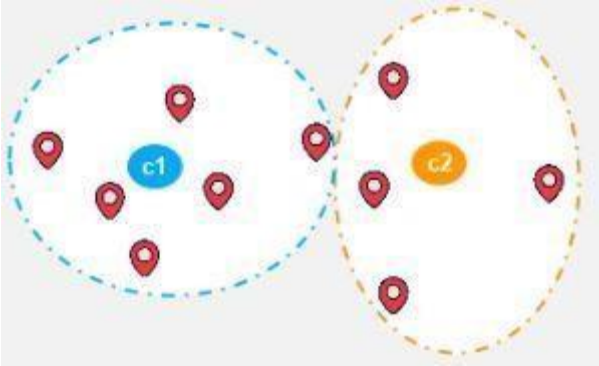
Reposition the random centroid to the actual centroid.



Step 8:

Once the cluster becomes static, the k-means algorithm is said to be converged.

The final cluster with centroids c1 and c2 is as shown below:



K-Means Clustering Algorithm

Let's say we have x1, x2, x3…… x(n) as our inputs, and we want to split this into K clusters.

The steps to form clusters are:

Step 1: Choose K random points as cluster centers called centroids.

Step 2: Assign each x(i) to the closest cluster by implementing Euclidean distance (i.e., calculating its distance to each centroid)

Step 3: Identify new centroids by taking the average of the assigned points.

Step 4: Keep repeating step 2 and step 3 until convergence is achieved Let's take a detailed look at it at each of these steps.

What are the Uses of Clustering?

Clustering has a myriad of uses in a variety of industries. Some common applications for clustering include the following:

- market segmentation
- social network analysis
- search result grouping
- medical imaging
- image segmentation
- anomaly detection

After clustering, each cluster is assigned a number called a cluster ID. Now, you can condense the entire feature set for an example into its cluster ID. Representing a complex example by a simple cluster ID makes clustering powerful. Extending the idea, clustering data can simplify large datasets.

Machine learning systems can then use cluster IDs to simplify the processing of large datasets. Thus, clustering's output serves as feature data for downstream ML systems.

Clustering Algorithms

When choosing a clustering algorithm, you should consider whether the algorithm scales to your dataset. Datasets in machine learning can have millions of examples, but not all clustering algorithms scale efficiently. Many clustering algorithms work by computing the similarity between all pairs of examples. This course focuses on the k-means algorithm

Types of clustering

Several approaches to clustering exist

Centroid-based Clustering

Centroid-based clustering organizes the data into non-hierarchical clusters, in contrast to hierarchical clustering defined below. k-means is the most widely-used centroid-based clustering algorithm. Centroid-based algorithms are efficient but sensitive to initial conditions and outliers. This course focuses on k-means because it is an efficient, effective, and simple clustering algorithm.

Density-based Clustering

Density-based clustering connects areas of high example density into clusters. This allows for arbitrary-shaped distributions as long as dense areas can be connected. These algorithms have difficulty with data of varying densities and high dimensions. Further, by design, these algorithms do not assign outliers to clusters.

Distribution-based Clustering

This clustering approach assumes data is composed of distributions, such as Gaussian distributions. In Figure 3, the distribution-based algorithm clusters

data into three Gaussian distributions. As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases. The bands show that decrease in probability. When you do not know the type of distribution in your data, you should use a different algorithm.

Hierarchical Clustering

Hierarchical clustering creates a tree of clusters. Hierarchical clustering, not surprisingly, is well suited to hierarchical data

4)ROC and Confusion Matrix

1) Confusion Matrix:

Classification Models have multiple categorical outputs. Most error measures will calculate the total error in our model, but we cannot find individual instances of errors in our model. The model might misclassify some categories more than others, but we cannot see this using a standard accuracy measure.

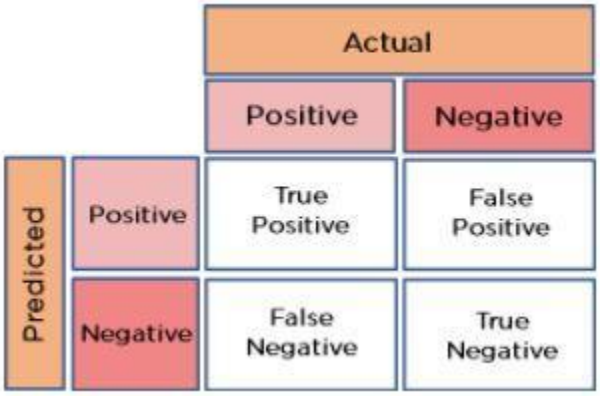
Furthermore, suppose there is a significant class imbalance in the given data. In that case, i.e., a class has more instances of data than the other classes, a model might predict the majority class for all cases and have a high accuracy score; when it is not predicting the minority classes. This is where confusion matrices are useful.

A confusion matrix presents a table layout of the different outcomes of the prediction and results of a classification problem and helps visualize its outcomes.

It plots a table of all the predicted and actual values of a classifier.

How to Create a 2x2 Confusion Matrix?

We can obtain four different combinations from the predicted and actual values of a classifier:



- **True Positive:** The number of times our actual positive values are equal to the predicted positive. You predicted a positive value, and it is correct.
- **False Positive:** The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.
- **True Negative:** The number of times our actual negative values are equal to predicted negative values. You predicted a negative value, and it is actually negative.
- **False Negative:** The number of times our model wrongly predicts negative values as positives. You predicted a negative value, and it is actually positive.

2) ROC

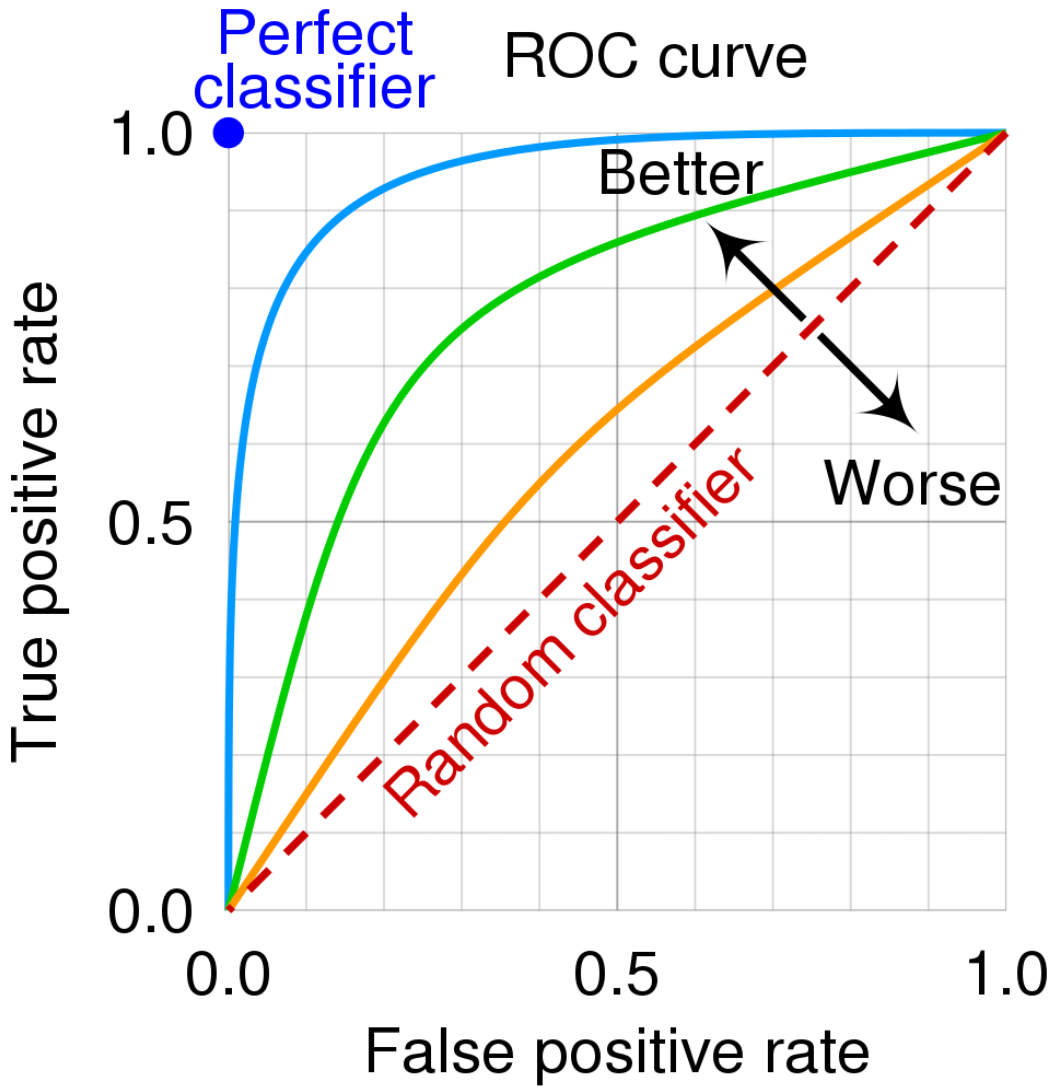
ROC stands for Receiver Operating Characteristic, which is a commonly used tool for evaluating the performance of binary classification algorithms. An ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for a binary classifier at different classification thresholds. The TPR is the proportion of true positives (correctly predicted positive cases) out of all actual positives, while the FPR is the proportion of false positives (incorrectly predicted positive cases) out of all actual negatives.

An ideal classifier will have a TPR of 1 and an FPR of 0, which would be represented by a point in the top left corner of the plot. A completely random classifier would have a diagonal line from the bottom left to the top right corner

of the plot, with an AUC of 0.5. A good classifier will have a curve that is closer to the top left corner, indicating a higher TPR and a lower FPR.

For example, if we had a binary classifier that was predicting whether an email was spam or not, and we plotted its ROC curve, it might look like a curve that rises steeply at the beginning (high TPR and low FPR), then levels off as the threshold increases, eventually approaching the diagonal line as the threshold approaches 1 (low TPR and high FPR). The area under the curve (AUC) would give us an overall measure of the classifier's performance, with higher values indicating better performance.

Sample Figure for an ROC curve



Tasks

1) Refer to the advertising dataset on Kaggle. You are supposed to run a linear regression with Sales as the dependent variable and TV as independent variable. You are required to perform a train/test split of 70-30 and run the linear regression through the stats-model library.

2) Refer to the Iris dataset. You are required to predict the target variable on the basis of the features of the independent variables. Refer to the following values for sepal length, sepal width, petal length and petal width:
   a) [3,4,5,2]
   b) [5,4,2,2]
   c) [7,1,2,4]
   d) [2,4,8,1]
3) Refer to the diabetes dataset on Kaggle. You are required to run a decision tress classifier on the dataset and calculate accuracy and plot the confusion matrix.