



Jntuk R20 ML UNIT-II - just for reference

machine learning (Jawaharlal Nehru Technological University, Kakinada)

UNIT-II

Supervised Learning(Regression/Classification)

Basic Methods: Distance based Methods, Nearest Neighbours, Decision Trees, Naive Bayes,

Linear Models: Linear Regression, Logistic Regression, Generalized Linear Models, Support Vector Machines

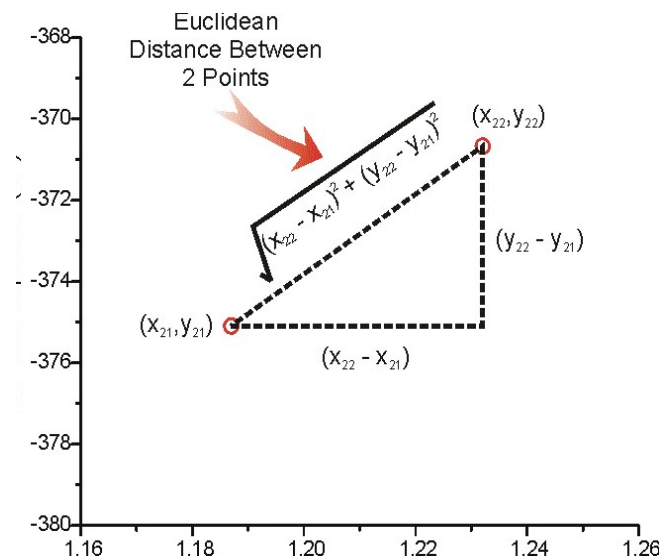
Binary Classification: Multiclass/Structured outputs, MNIST, Ranking.

1. Distance based Methods

Distance based algorithms are machine learning algorithms that classify queries by computing distances between these queries and a number of internally stored exemplars. Exemplars that are closest to the query have the largest influence on the classification assigned to the query.

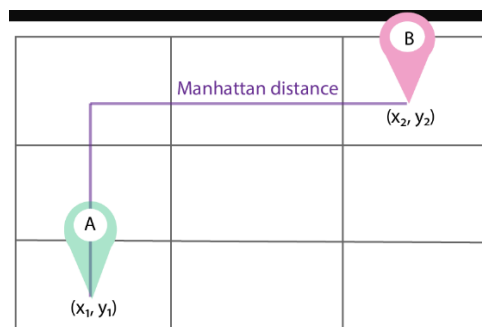
Euclidean Distance

It is the most common use of distance. In most cases when people said about distance, they will refer to Euclidean distance. Euclidean distance is also known as simply distance. When data is dense or continuous, this is the best proximity measure. The Euclidean distance between two points is the length of the path connecting them. The Pythagorean theory gives distance between two points.



Manhattan Distance

- If you want to find Manhattan distance between two different points (x_1, y_1) and (x_2, y_2) such as the following, it would look like the following:
- Manhattan distance = $(x_2 - x_1) + (y_2 - y_1)$
- Diagrammatically, it would look like traversing the path from point A to point B while walking on the pink straight line.



Minkowski Distance

The generalized form of the Euclidean and Manhattan Distances is the Minkowski Distance. You can express the Minkowski distance as

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

The order of the norm is represented by p.

When an order(p) is 1, Manhattan Distance is represented, and when order(p) is 2 in the above formula, Euclidean Distance is represented.

2. Nearest Neighbours

The abbreviation KNN stands for "K-Nearest Neighbour". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements.

The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

KNN calculates the distance from all points in the proximity of the unknown data and filters out the ones with the shortest distances to it. As a result, it's often referred to as a distance-based algorithm.

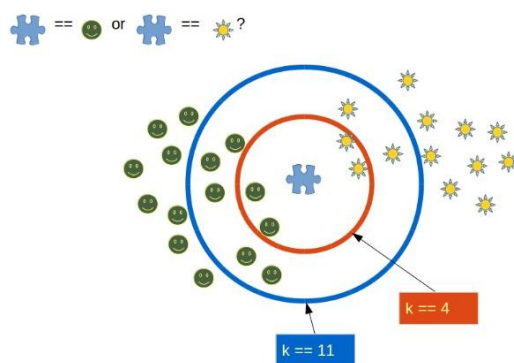
In order to correctly classify the results, we must first determine the value of K (Number of Nearest Neighbours).

It is recommended to always select an odd value of K ~

When the value of K is set to even, a situation may arise in which the elements from both groups are equal. In the diagram below, elements from both groups are equal in the internal "Red" circle (k == 4).

In this condition, the model would be unable to do the correct classification for you. Here the model will randomly assign any of the two classes to this new unknown data.

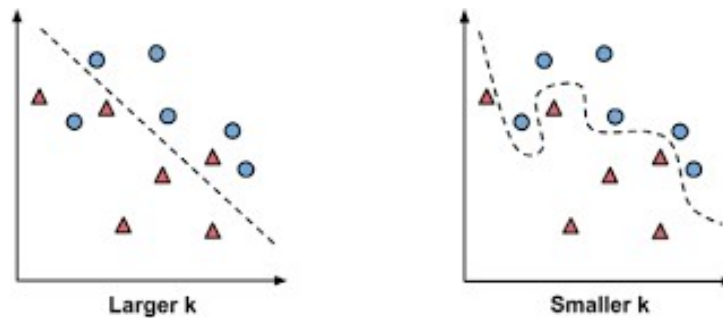
Choosing an odd value for K is preferred because such a state of equality between the two classes would never occur here. Due to the fact that one of the two groups would still be in the majority, the value of K is selected as odd.



Src: <https://images.app.goo.gl/Q8ZKxQ8mhP68yxqn7>

The impact of selecting a smaller or larger K value on the model

- Larger K value: The case of underfitting occurs when the value of k is increased. In this case, the model would be unable to correctly learn on the training data.
- Smaller k value: The condition of overfitting occurs when the value of k is smaller. The model will capture all of the training data, including noise. The model will perform poorly for the test data in this scenario.



Src:

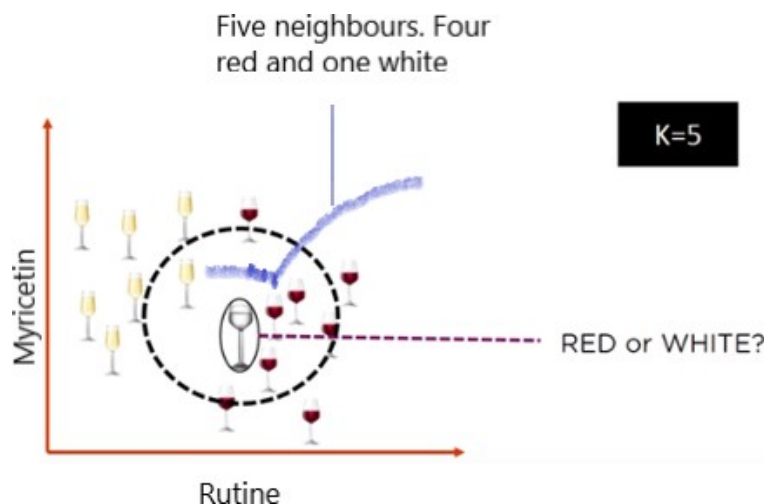
<https://images.app.goo.gl/vXStNS4NeEqUCDXn8>

How does KNN work for 'Classification' and 'Regression' problem statements?

□ Classification

When the problem statement is of 'classification' type, KNN tends to use the concept of "Majority Voting". Within the given range of K values, the class with the most votes is chosen.

Consider the following diagram, in which a circle is drawn within the radius of the five closest neighbours. Four of the five neighbours in this neighbourhood voted for 'RED,' while one voted for 'WHITE.' It will be classified as a 'RED' wine based on the majority votes.



Src:

<https://images.app.goo.gl/Ud42nZn8Q8FpDVcs5>

Real-world example:

Several parties compete in an election in a democratic country like India. Parties compete for voter support during election campaigns. The public votes for the candidate with whom they feel more connected.

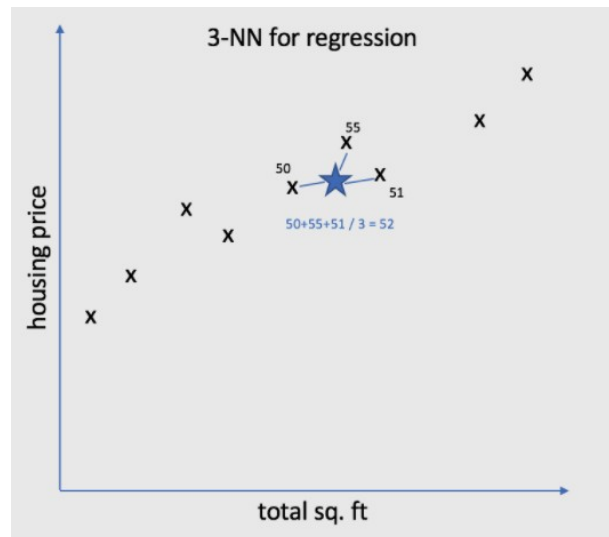
When the votes for all of the candidates have been recorded, the candidate with the most votes is declared as the election's winner.

Regression

KNN employs a mean/average method for predicting the value of new data. Based on the value of K, it would consider all of the nearest neighbours.

The algorithm attempts to calculate the mean for all the nearest neighbours' values until it has identified all the nearest neighbours within a certain range of the K value.

Consider the diagram below, where the value of k is set to 3. It will now calculate the mean (52) based on the values of these neighbours (50, 55, and 51) and allocate this value to the unknown data.



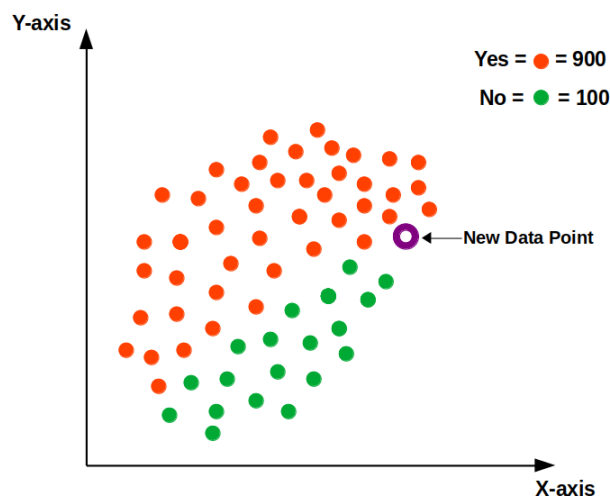
Src: <https://images.app.goo.gl/pzW97weL6vHJByni8>

Impact of Imbalanced dataset and Outliers on KNN

- Imbalanced dataset

When dealing with an imbalanced data set, the model will become biased. Consider the example shown in the diagram below, where the "Yes" class is more prominent.

As a consequence, the bulk of the closest neighbours to this new point will be from the dominant class. Because of this, we must balance our data set using either an "Upscaling" or "Downscaling" strategy.



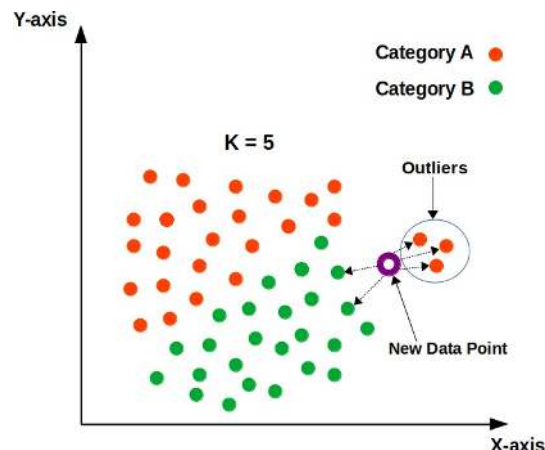
Src: <https://images.app.goo.gl/1XkGHtn16nXDkrTL7>

- Outliers

Outliers are the points that differ significantly from the rest of the data points.

The outliers will impact the classification/prediction of the model. The appropriate class for the new data point, according to the following diagram, should be "Category B" in green.

The model, however, would be unable to have the appropriate classification due to the existence of outliers. As a result, removing outliers before using KNN is recommended.



Src: <https://images.app.goo.gl/K35WtKYCTnGBDLW36>

Importance of scaling down the numeric variables to the same level

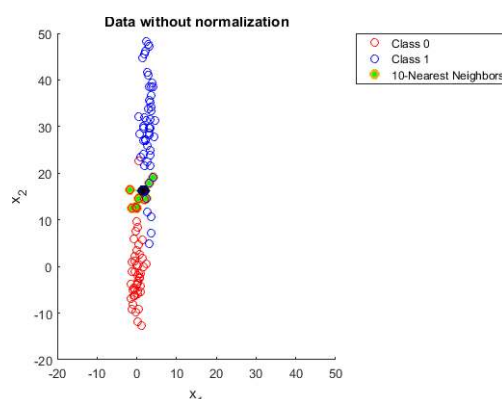
Data has 2 parts: –

1) Magnitude

2) Unit

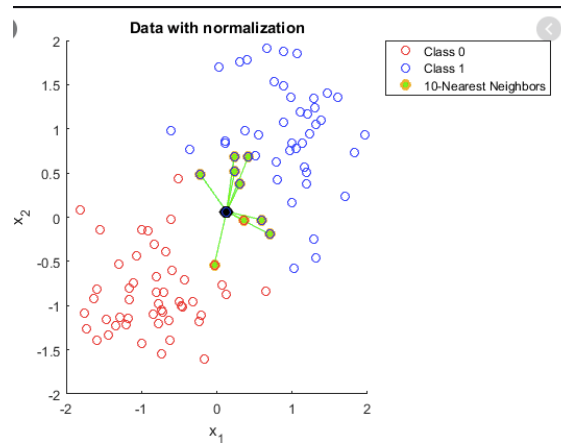
For instance; if we say 20 years then "20" is the magnitude here and "years" is its unit.

Since it is a distance-dependent algorithm, KNN selects the neighbours in the closest vicinity based solely on the magnitude of the data. Have a look at the diagram below; the data is not scaled, so it can not find the closest neighbours correctly. As a consequence, the outcome will be influenced.



Src: <https://images.app.goo.gl/M1oenLdEo427VBGc7>

The data values in the previous figure have now been scaled down to the same level in the following example. Based on the scaled distance, all of the closest neighbours would be accurately identified.



Src: <https://images.app.goo.gl/CtdoNXq5hPVvynre9>

3. Decision Trees

Like SVMs, Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets.

Decision Trees are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today.

The concepts of Entropy and Information Gain in Decision Tree Learning

While constructing a decision tree, the very first question to be answered is, Which Attribute Is the Best Classifier?

The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.

We would like to select the attribute that is most useful for classifying examples.

What is a good quantitative measure of the worth of an attribute? We will define a statistical property, called *information gain*, that measures how well a given attribute separates the training examples according to their target classification.

ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

ENTROPY MEASURES THE HOMOGENEITY OF EXAMPLES

Entropy characterizes the (im)purity of an arbitrary collection of examples.

Given a collection S , containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

where p_{+} , is the proportion of positive examples in S and p_{-} , is the proportion of negative examples in S . In all calculations involving entropy, we define $0 \log 0$ to be .

Let, S is a collection of training examples,

p_+ the proportion of positive examples in S

p_- the proportion of negative examples in S

Examples

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \quad [0 \log_2 0 = 0]$$

$$Entropy([14+, 0-]) = -14/14 \log_2 (14/14) - 0 \log_2 (0) = 0$$

$$Entropy([9+, 5-]) = -9/14 \log_2 (9/14) - 5/14 \log_2 (5/14) = 0.94$$

$$Entropy([7+, 7-]) = -7/14 \log_2 (7/14) - 7/14 \log_2 (7/14) = 1/2 + 1/2 = 1$$

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data.

Now, the *information gain* is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

More precisely, the information gain, $Gain(S, A)$ of an attribute A , relative to a collection of examples S , is defined as,

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where $Values(A)$ is the set of all possible values for attribute A , and S_v is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$)

For example, suppose S is a collection of training-example days described by attributes including *Wind*, which can have the values *Weak* or *Strong*.

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14) Entropy(S_{Weak}) \\ &\quad - (6/14) Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

Information gain is precisely the measure used by ID3 to select the best attribute at each step in growing the tree.

The use of information gain is to evaluate the relevance of attributes.

Q) Decision Tree construction using Entropy and information gain / Classification

- ID3 Algorithm:

Ross Quinlan developed ID# algorithm in 1987. It is greedy algorithm for decision tree construction. The ID3 algorithm is inspired from CLSs algorithm by Hunt in 1966. CLSs algorithm start with training object set, $O=\{o_1, o_2, \dots, o_n\}$ from a universal where each object is described by a set of m attributes aj_1, aj_2, \dots, aj_k is selected and a tree node structure is formed to represent A_j .

In ID3 system, a relatively small number of training examples are randomly selected from a large set of objects O through a window. Using these a preliminary decision tree is constructed. The tree is then tested by scanning all the objects in O to see if there are any expectations to the tree. A new subset is formed using the original examples together with some of the expectations found during the scan. This process is repeated until no exceptions are found. The resulting decision tree is can be used to classify new objects.

- ID3 is the way in which the attributes are ordered for use in the classification process.
- Attributes which discriminate best are selected for the evaluation first.
- This requires computing an estimate of the expected information gain using all available attributes and then selecting the attribute having the largest expected gain.
- This attribute is assigned to root node.
- The attribute having the next largest gain is assigned to the next level of nodes in the tree and so on until the leaves of the tree have been reached.
- A decision tree is created by recursive selection of the best attribute in a top-down manner to use at the current node in the tree.
- When a particular attribute is selected as the current node, it creates its child nodes, one for each possible value of the selected attribute.
- The next step is to partition the samples using the possible values of the attribute and to assign these subsets of examples to the appropriate child node.
- The process is repeated for every child node until we get a positive or negative result for all nodes associated with the particular sample.

ID3 Algorithm

1. Establish a table R with classification attributes $(aj_1, aj_2, \dots, aj_k)$.
2. Compute classification Shannon entropy by

$$H(x) = - \sum_{i=1}^n p(x_i) \log_b p(x_i)$$

3. For each attribute in R , calculate information gain using classification attribute.

$$I_G(S, A) = I_E(S) - \sum^n (p(\in_{S^A}^A) \times I_E(\in_{S^A}^A))$$

$$\in_{S^A}^A = \text{Subset } A \text{ of } S.$$

4. Select attribute with the highest gain to be the next node in the tree.
5. Remove node attribute, creating reduced table R_s .
6. Repeat steps 3-5 until all the attributes have been used, or the same classification value remains for all rows in the reduced table.

Information gain for attribute A on set S is defined by taking the entropy of S and subtracting from it the summation of entropy of each subset of S, determined by the values of A multiplied by each subset's proportion of S.

Example:

Play Tennis Dataset

Day	Outlook	Temp	Humidity	Wind	Tennis?
D1	Sunny	Hot	High	Weak	NO
D2	Sunny	Hot	High	Strong	NO
D3	Cloudy	Hot	High	Weak	YES
D4	Rain	Mild	High	Weak	YES
D5	Rain	Cool	Normal	Weak	YES
D6	Rain	Cool	Normal	Strong	NO
D7	Cloudy	Cool	Normal	Strong	YES
D8	Sunny	Mild	High	Weak	NO
D9	Sunny	Cool	Normal	Weak	YES
D10	Rain	Mild	Normal	Weak	YES
D11	Sunny	Mild	Normal	Strong	YES
D12	Cloudy	Mild	High	Strong	YES
D13	Cloudy	Hot	Normal	Weak	YES
D14	Rain	Mild	High	Strong	NO

(1)

$$\rightarrow \text{Entropy } S[9,5] = -(9/14) \log_2(9/14) - (5/14) \log_2(5/14)$$

$$= 0.407 + 0.530$$

$$I\left(\begin{smallmatrix} S_1 & S_2 \\ \text{yes} & \text{no} \end{smallmatrix}\right) = 0.93$$

Entropy for outlook

for outlook - sunny $\begin{cases} \text{yes} - 2 (S_{11}) \\ \text{no} - 3 (S_{21}) \end{cases}$

$$I(S_{11}, S_{21}) = -(2/5) \log_2(2/5) + -(3/5) \log_2(3/5)$$

$$= 0.5286 + 0.4414$$

$$= 0.970$$

for outlook - cloudy $\begin{cases} \text{yes} - 4 \\ \text{no} - 0 \end{cases}$

$$I(S_{12}, S_{22}) = -(4/4) \log_2(4/4) + -(0/4) \log_2(0/4)$$

$$= 0$$

for outlook - Rain $\begin{cases} \text{yes} - 3 \\ \text{no} - 2 \end{cases}$

$$I(S_{13}, S_{23}) = -(3/5) \log_2(3/5) + -(2/5) \log_2(2/5)$$

$$= 0.4414 + 0.5286$$

$$= 0.970$$

$$E(\text{outlook}) = ((5/14) \times 0.970) + ((4/14) \times 0) + ((5/14) \times 0.970)$$

$$= 0.346 + 0 + 0.347$$

$$= 0.693$$

$$\text{Infogain}(\text{outlook}) = I(S_1, S_2) - E(\text{outlook})$$

$$= 0.932 - 0.693$$

$$= 0.238$$

⇒ Entropy for temp

for temp = hot $\begin{matrix} \text{yes} - 2 \\ \text{no} - 2 \end{matrix}$

$$I(S_{11}, S_{21}) = -(2/4) \log_2(2/4) + (2/4) \log_2(2/4) \\ = 0.5 + 0.5$$

for temp = mild $\begin{matrix} \text{yes} - 4 \\ \text{no} - 2 \end{matrix}$

$$I(S_{12}, S_{22}) = -(4/6) \log_2(4/6) + (2/6) \log_2(2/6) \\ = 0.350 + 0.4717 \\ = 0.8217$$

for temp = cool $\begin{matrix} \text{yes} - 3 \\ \text{no} - 1 \end{matrix}$

$$I(S_{13}, S_{23}) = -(3/4) \log_2(3/4) + (1/4) \log_2(1/4) \\ = 0.302 + 0.4999 \\ = 0.8019$$

$$E(\text{temp}) = ((4/14) \times 1) + ((6/14) \times 0.8217) + ((4/14) \times 0.8019) \\ = 0.285 + 0.381 + 0.231 \\ = 0.8977$$

$$\text{gain}(\text{temp}) = I(S_1, S_2) - E(\text{temp}) \\ = 0.932 - 0.8977 \\ = 0.0343$$

→ Entropy for humidity

for humidity $\begin{cases} \text{high} - 3 \\ \text{normal} - 4 \end{cases}$

$$\begin{aligned} I(S_{11}, S_{21}) &= (-3/7) \log_2(-3/7) + (-4/7) \log_2(-4/7) \\ &= 0.52 + 0.461 \\ &= 0.985 \end{aligned}$$

$$\begin{aligned} I(S_{12}, S_{22}) &= (-6/7) \log_2(-6/7) + (-1/7) \log_2(-1/7) \\ &= 0.1904 + 0.406 \\ &= 0.5916 \end{aligned}$$

$$\begin{aligned} E(\text{humidity}) &= ((7/14) \times 0.985) + ((7/14) \times 0.5916) \\ &= 0.78835 \end{aligned}$$

$$\begin{aligned} \text{Info gain}(\text{humidity}) &= I(S_1, S_2) - E(\text{humidity}) \\ &= 0.932 - 0.78835 \\ &= 0.14365 \end{aligned}$$

→ Entropy for wind

for wind $\begin{cases} \text{weak} - 8 \\ \text{strong} - 6 \end{cases}$

$$\begin{aligned} \text{fair } I(S_{11}, S_{21}) &= (-6/8) \log_2(-6/8) + (-2/8) \log_2(-2/8) \\ &= 0.323 + 0.497 \\ &= 0.8080 \end{aligned}$$

$$\begin{aligned} I(S_{21}, S_{22}) &= (-3/6) \log_2(-3/6) + (-3/6) \log_2(-3/6) \\ &= 0.500 + 0.500 \\ &= 1 \end{aligned}$$

$$\begin{aligned}
 E(\text{wind}) &= ((8/14) \times 0.8080) + ((6/14) \times 1) \\
 &= 0.467 + 0.428 \\
 &= 0.890
 \end{aligned}$$

$$\begin{aligned}
 \text{Infogain}(\text{wind}) &= I(S_1, S_2) - E(\text{wind}) \\
 &= 0.932 - 0.890 \\
 &= 0.041
 \end{aligned}$$

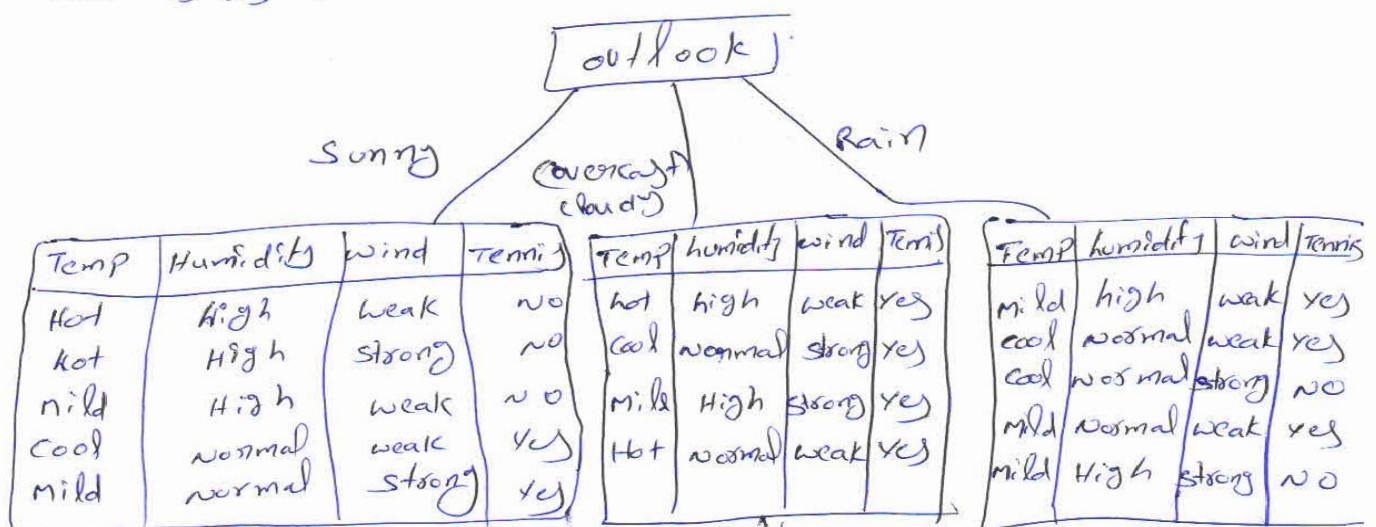
$$\text{Finally } \text{Infogain}(S, \text{outlook}) = 0.238$$

$$\text{Infogain}(S, \text{Temp}) = 0.0343$$

$$\text{Infogain}(S, \text{Humidity}) = 0.143$$

$$\text{Infogain}(S, \text{wind}) = 0.041$$

among all outlook is having highest infogain. So outlook is selected as best split. According to that the decision tree is as follow.



↓
all class labels are same. so replace with leaf node

for outlook sunny

$$I(\text{outlook}, S_1, S_2) = (-4/5) \log_2(4/5) + (-3/5) \log_2(3/5) \\ = 0.974$$

→ Entropy for temp $\begin{cases} \text{hot} - 2 \\ \text{mild} - 2 \\ \text{cool} - 1 \end{cases}$

for hot $\begin{cases} \text{yes} - 0 \\ \text{no} - 2 \end{cases}$

$$I(S_{11}, S_{21}) = (-0/2) \log_2(0/2) + (-2/2) \log_2(2/2) \\ = 0$$

for mild $\begin{cases} \text{yes} - 1 \\ \text{no} - 1 \end{cases}$

$$I(S_{12}, S_{22}) = (-1/2) \log_2(1/2) + (-1/2) \log_2(1/2) \\ = 1$$

for cool $\begin{cases} \text{yes} - 1 \\ \text{no} - 0 \end{cases}$

$$I(S_{13}, S_{23}) = (-1/1) \log_2(1/1) + (-0/1) \log_2(0/1) \\ = 0$$

$$E(\text{temp}) = (2/5 \times 0) + (2/5 \times 1) + (1/5 \times 0) \\ = 0.4$$

$$I(\text{outlook}, \text{temp}) = 0.974 - 0.4 \\ = 0.574$$

→ Entropy for humidity $\begin{cases} \text{high} - 3 \\ \text{normal} - 2 \end{cases}$

for high $\begin{cases} \text{yes} - 0 \\ \text{no} - 3 \end{cases}$

$$I(S_{11}, S_{12}) = (-0/3) \log_2(0/3) + (-3/3) \log_2(3/3) \\ = 0$$

→ for normal $\begin{cases} \text{yes} = 2 \\ \text{no} = 0 \end{cases}$

$$I(S_{12}, S_{22}) = (-2/2) \log_2(-2/2) + (-0/2) \log_2(0/2)$$

$$= 0$$

$$E(\text{humidity}) = 0$$

$$\text{info gain}(\text{humidity}) = 0.974 - 0$$

$$= 0.974$$

→ for wind $\begin{cases} \text{weak} = 3 \\ \text{strong} = 2 \end{cases}$

for weak $\begin{cases} \text{yes} = 1 \\ \text{no} = 2 \end{cases}$

$$I(S_{11}, S_{12}) = (-1/3) \log_2(-1/3) + (-2/3) \log_2(-2/3)$$

$$= 0.525 + 0.39$$

$$= 0.91$$

for strong $\begin{cases} \text{yes} = 1 \\ \text{no} = 1 \end{cases}$

$$I(S_{12}, S_{22}) = (-1/2) \log_2(-1/2) + (-1/2) \log_2(-1/2)$$

$$= 1$$

$$E(\text{wind}) = (3/5) \times 0.918 + (2/5) \times 1$$

$$= 0.946$$

$$\text{info gain}(\text{wind}) = 0.974 - 0.946$$

$$= 0.019$$

so finally for the left subtree the info gains are as follows

$$\text{info gain}(S, \text{temp}) = 0.574$$

$$\text{info gain}(S, \text{humidity}) = 0.974$$

$$\text{info gain}(S, \text{wind}) = 0.019$$

Among all humidity is having highest info gain.

(4)

→ for - outlook - rainy.

$$\text{info gain}(s_1, s_2) = -(3/5) \log_2(3/5) - (2/5) \log_2(2/5)$$

$$= 0.974$$

entropy for temp $\left\{ \begin{array}{l} \text{hot} - 0 \\ \text{mild} - 3 \\ \text{cool} - 2 \end{array} \right.$

No. of tuples for hot are equal to zero

for mild $\left\{ \begin{array}{l} \text{yes} - 2 \\ \text{no} - 1 \end{array} \right.$

$$I(s_{12}, s_{22}) = -(2/3) \log_2(2/3) - (1/3) \log_2(1/3)$$

$$= 0.918$$

for cool $\left\{ \begin{array}{l} \text{yes} - 1 \\ \text{no} - 1 \end{array} \right.$

$$I(s_{13}, s_{23}) = -(1/2) \log_2(1/2) - (1/2) \log_2(1/2)$$

$$= 1$$

$$E(\text{temp}) = 0 + (3/5) \times 0.918 + (2/5) \times 1$$

$$= 0.9508$$

$$\text{info gain}(\text{temp}) = 0.974 - 0.9508$$

$$= 0.0232$$

entropy for humidity $\left\{ \begin{array}{l} \text{high} - 2 \\ \text{normal} - 3 \end{array} \right.$

for high $\left\{ \begin{array}{l} \text{yes} - 1 \\ \text{no} - 1 \end{array} \right.$

$$I(s_{11}, s_{21}) = -(1/2) \log_2(1/2) - (1/2) \log_2(1/2)$$

$$= 1$$

for normal $\begin{cases} \text{yes} - 2 \\ \text{no} - 1 \end{cases}$

$$I(S_{12}, S_{22}) = -\left(\frac{2}{3}\right) \log_2\left(\frac{2}{3}\right) - \left(\frac{1}{3}\right) \log_2\left(\frac{1}{3}\right) \\ = 0.91$$

$$\text{Entropy (humidity)} = \left(\frac{2}{5}\right) \times 1 + \left(\frac{3}{5}\right) \times 0.91 \\ = 0.4 + 0.546 \\ = 0.946$$

$$\text{Infogain (humidity)} = 0.974 - 0.946 \\ = 0.028$$

→ Entropy for wind $\begin{cases} \text{weak} - 3 \\ \text{strong} - 2 \end{cases}$

for weak $\begin{cases} \text{yes} - 3 \\ \text{no} - 0 \end{cases}$

$$I(S_{11}, S_{21}) = -\left(\frac{3}{3}\right) \log_2\left(\frac{3}{3}\right) - \left(\frac{0}{3}\right) \log_2\left(\frac{0}{3}\right) \\ = 0$$

for strong $\begin{cases} \text{yes} - 0 \\ \text{no} - 2 \end{cases}$

$$I(S_{12}, S_{22}) = -\left(\frac{0}{2}\right) \log_2\left(\frac{0}{2}\right) - \left(\frac{2}{2}\right) \log_2\left(\frac{2}{2}\right) \\ = 0$$

$$\text{Entropy (wind)} = 0$$

$$\text{Infogain (wind)} = 0.974 - 0 \\ = 0.974$$

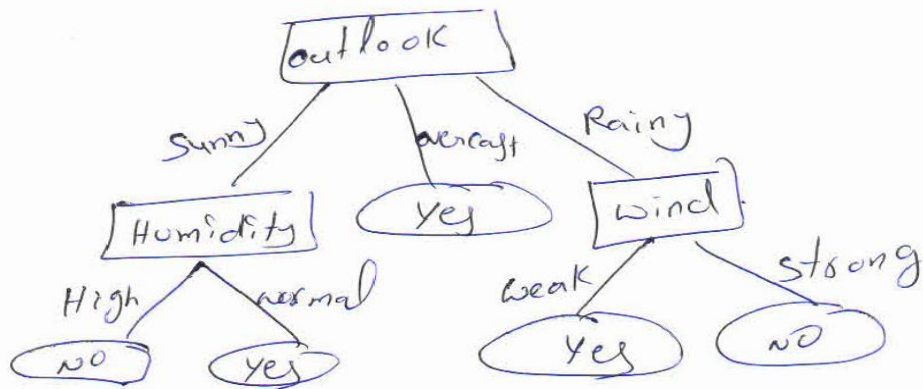
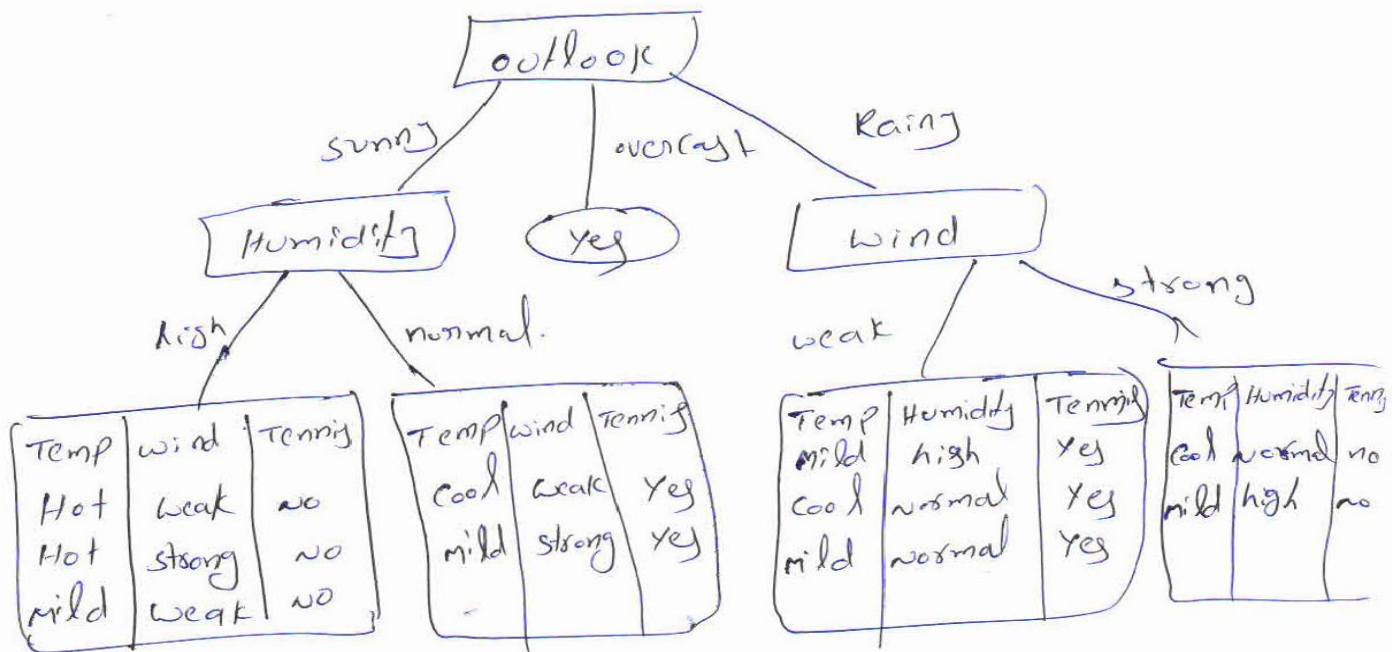
So finally for the right subtree the infogain values are as follows:

$$\text{Infogain}(S, \text{temp}) = 0.0232$$

$$\text{Infogain}(S, \text{humidity}) = 0.028$$

$$\text{Infogain}(S, \text{wind}) = 0.974$$

Among all wind is having highest info gain, so it will be the root for right subtree. Finally the final decision tree is as follows!



Q) Decision Tree to Decision Rules/ Rule Extraction from Trees / Making Predictions

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.

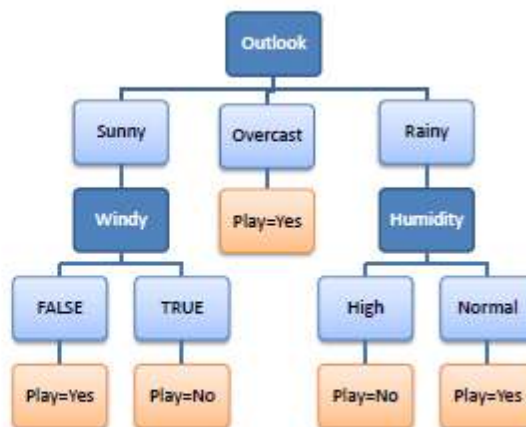
R_1 : IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2 : IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

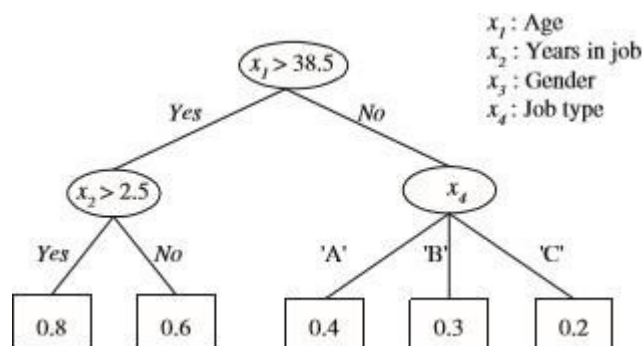
R_3 : IF (Outlook=Overcast) THEN Play=Yes

R_4 : IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5 : IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



A decision tree does its own feature extraction. The univariate tree only uses the necessary variables and after the tree is built, certain features may not be used at all. As shown in fig. x_1, x_2 and x_4 variables are used but not x_3 . It is possible to use a decision tree for feature extraction: we build a tree and then take only those features used by the tree as inputs to another learning method.



Main advantage of decision tree is interpretability: the decision nodes carry conditions that are simple to understand. Each path from the root to a leaf corresponds to one conjunction of tests as all these conditions must be satisfied to reach leaf node. These paths together can be written down as a set of IF-THEN rules, called a rule base.

For example, the decision tree of fig can be written down as the following set of rules:

R1: IF (age>38.5) AND (years-in-job>2.5) THEN $y=0.8$

R2: IF (age>38.5) AND (years-in-job ≤ 2.5) THEN $y=0.6$

R3: IF (age ≤ 38.5) AND (job-type = 'A') THEN $y=0.4$

R4: IF (age ≤ 38.5) AND (job-type = 'B') THEN $y=0.3$

R5: IF (age ≤ 38.5) AND (job-type = 'C') THEN $y=0.2$

Such a rule base allows knowledge extractor, it can be easily understood and allows experts to verify the model learned from data. For each rule, one can also calculate the percentage of training data covered by the rule namely rule support. The rules reflect the main characteristics of the dataset. They show the important features and split positions. For instance, in this example, we see that in terms of our purpose (y),

- people who are thirty –eight years old or less are different from people who are thirty- nine or more years old.
- And in the latter group, it is the job type that makes them different.
- In the former group no of years in job is the best discriminating characteristic.

In case of classification tree, there may be more than one leaf labelled with the same class. In such a case, these multiple conjunctive expressions corresponding to different paths can be combined as a disjunction. The class region then corresponds to a union of these multiple patches, each patch corresponding to the region defined by one leaf. For example

IF($x \leq w_{10}$)OR(($x_1 > w_{10}$) AND ($x_2 \leq w_{20}$)) Then C1

Pruning rules is possible for simplification. Pruning a subtree corresponds to pruning terms from a number of rules at the same time. It may be possible to prune a term from one rule without touching other examples. For example, in the previous rule set, for R3, if we see that all whose job-type = 'A' have outcomes close to 0.4, regardless of age, R3 can be pruned as

R3` : IF (job-type = 'A') THEN $y = 0.4$

Once the rules are pruned we cannot write them back as a tree anymore.

Q) Training and Visualizing a Decision Tree

To understand Decision Trees, let us just build one and take a look at how it makes predictions.

The following code trains a DecisionTreeClassifier

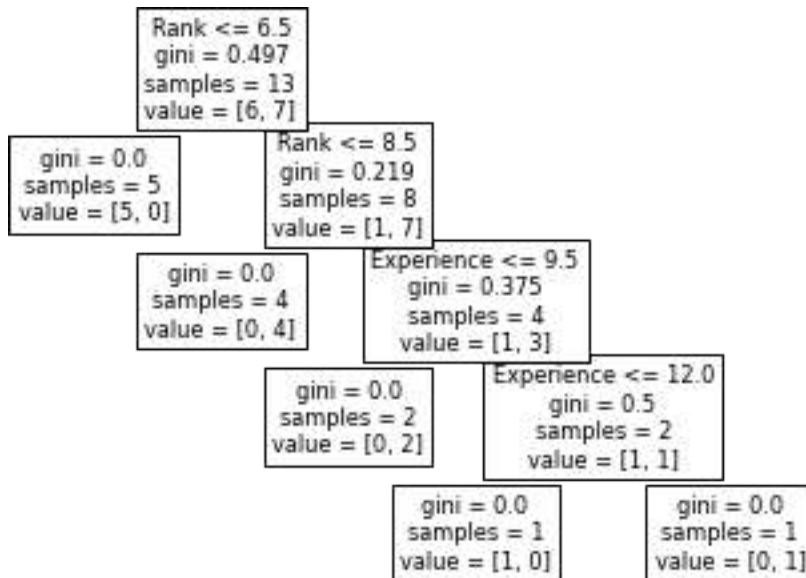
```
import sys
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
df = pd.read_csv("DTree.csv")
print(df)
#mapping charater data to numeric
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)
print(df)
features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
```



```
y = df['Go']
```

```
dtree = DecisionTreeClassifier()
dtree = dtree.fit(X, y)
tree.plot_tree(dtree, feature_names=features)
```

Output:-



Q) Decision Tree for Regression

Decision Trees are also capable of performing regression tasks. Let us build a regression tree using Scikit-Learn's `DecisionTreeRegressor` class, training it on a noisy quadratic dataset with `max_depth=2`:

#importing Required Packages

```
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.datasets import load_iris
```

```
from sklearn import tree
```

```
import matplotlib.pyplot as
```

```
plt #Loading Dataset
```

```
iris = load_iris()
```

```
X = iris.data[:, 2:] # petal length and
```

```
width y = iris.target
```

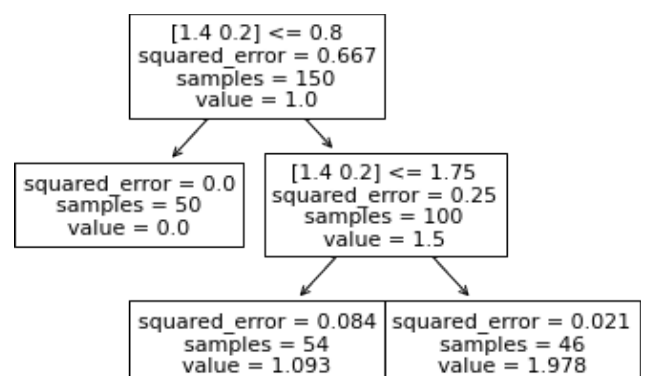
```
#Fitting the model
```

```
tree_reg = DecisionTreeRegressor(max_depth=2)
```

```
tree_reg.fit(X, y)
```

```
#Visualizing the Tree
```

```
tree.plot_tree(tree_reg, feature_names=X)
```



Q) The CART Training Algorithm

Scikit-Learn uses the *Classification And Regression Tree* (CART) algorithm to train Decision Trees (also called “growing” trees). The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k

CART algorithm uses Gini Impurity to split the dataset into a decision tree. It does that by searching for the best homogeneity for the sub nodes, with the help of the Gini index criterion.

Gini index/Gini impurity: The Gini index is a metric for the classification tasks in CART. It stores the sum of squared probabilities of each class. It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient. It works on categorical variables, provides outcomes either “successful” or “failure” and hence conducts binary splitting only.

The degree of the Gini index varies from 0 to 1,

- Where 0 depicts that all the elements are allied to a certain class, or only one class exists there.
- The Gini index of value 1 signifies that all the elements are randomly distributed across various classes, and
- A value of 0.5 denotes the elements are uniformly distributed into some

classes. Mathematically, we can write Gini Impurity as follows:

$$Gini = 1 - \sum_{i=1}^n (p_i)^2$$

where p_i is the probability of an object being classified to a particular class. Classification tree

A classification tree is an algorithm where the target variable is categorical. The algorithm is then used to identify the “Class” within which the target variable is most likely to fall. Classification trees are used when the dataset needs to be split into classes that belong to the response variable (like yes or no)

Regression tree

A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value. Regression trees are used when the response variable is continuous. For example, if the response variable is the temperature of the day.

Advantages of CART

- Results are simplistic.
- Classification and regression trees are Nonparametric and Nonlinear.
- Classification and regression trees implicitly perform feature selection.
- Outliers have no meaningful effect on CART.
- It requires minimal supervision and produces easy-to-understand

models. Limitations of CART

- Overfitting.
- High Variance.
- low bias.
- the tree structure may be unstable.

Applications of the CART algorithm

- For quick Data insights.
- In Blood Donors Classification.
- For environmental and ecological data.
- In the financial sectors.

4. Naive Bayes Classifier

Answer: Already covered in previous units

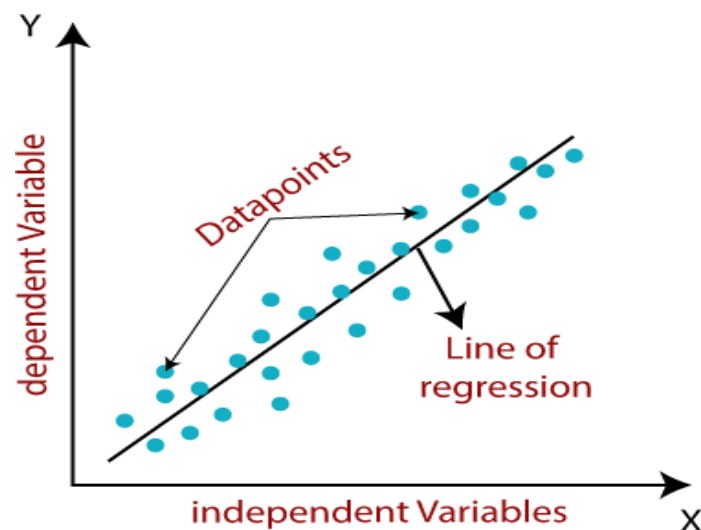
Chapter-II

Linear Models: Linear Regression, Logistic Regression, Generalized Linear Models, Support Vector Machines

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image:



Mathematically, we can represent a linear regression as:

$$y = a_0 + a_1x + \epsilon$$

Here,

Y= Dependent Variable (Target Variable)

X= Independent Variable (predictor Variable)

a_0 = intercept of the line (Gives an additional degree of freedom)

a_1 = Linear regression coefficient (scale factor to each input value).

ϵ = random error

The values for x and y variables are training datasets for Linear Regression model representation.

Types of Linear Regression

Linear regression can be further divided into two types of the algorithm:

- Simple Linear Regression:
If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.

- Multiple Linear regression:
If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

1. Simple linear regression

Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable. The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.

The key point in Simple Linear Regression is that the *dependent variable must be a continuous/real value*. However, the independent variable can be measured on continuous or categorical values.

Simple Linear regression algorithm has mainly two objectives:

- ▢ Model the relationship between the two variables. Such as the relationship between Income and expenditure, experience and Salary, etc.
- ▢ Forecasting new observations. Such as Weather forecasting according to temperature, Revenue of a company according to the investments in a year, etc.

Simple Linear Regression Model:

The Simple Linear Regression model can be represented using the below equation:

$$y = a_0 + a_1x + \epsilon$$

Where,

a_0 = It is the intercept of the Regression line (can be obtained putting $x=0$)

a_1 = It is the slope of the regression line, which tells whether the line is increasing or decreasing.

ϵ = The error term. (For a good model it will be negligible)

Implementation of Simple Linear Regression Algorithm using Python

Problem Statement example for Simple Linear Regression:

Here we are taking a dataset that has two variables: salary (dependent variable) and experience (Independent variable). The goals of this problem is:

- ▢ We want to find out if there is any correlation between these two variables
- ▢ We will find the best fit line for the dataset.
- ▢ How the dependent variable is changing by changing the independent variable.

Here, we will create a Simple Linear Regression model to find out the best fitting line for representing the relationship between these two variables.

To implement the Simple Linear regression model in machine learning using Python, we need to follow the below steps:

Step-1: Data Pre-processing

The first step for creating the Simple Linear Regression model is data pre-processing. We have already done it earlier in this tutorial. But there will be some changes, which are given in the below steps:

a) First, we will import the three important libraries, which will help us for loading the dataset, plotting the graphs, and creating the Simple Linear Regression model.

```
# Importing the  
libraries import numpy  
as np
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

b) Next, we will load the dataset into our code. After that, we need to extract the dependent and independent variables from the given dataset. The independent variable is years of experience, and the dependent variable is salary.

```
# Importing the dataset
```

```
dataset = pd.read_csv('Salary_Data.csv')
```

```
X = dataset.iloc[:, :-1].values
```

```
y = dataset.iloc[:, 1].values
```

In the above lines of code, for x variable, we have taken -1 value since we want to remove the last column from the dataset. For y variable, we have taken 1 value as a parameter, since we want to extract the second column and indexing starts from the zero.

c) Next, we will split both variables into the test set and training set. We have 30 observations, so we will take 20 observations for the training set and 10 observations for the test set. We are splitting our dataset so that we can train our model using a training dataset and then test the model using a test dataset.

```
# Splitting the dataset into the Training set and Test set
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state =
```

0) Step-2: Fitting the Simple Linear Regression to the Training Set:

Now the second step is to fit our model to the training dataset. To do so, we will import the LinearRegression class of the linear_model library from the scikit learn. After importing the class, we are going to create an object of the class named as a regressor.

```
# Fitting Simple Linear Regression to the Training set
```

```
from sklearn.linear_model import LinearRegression
```

```
regressor = LinearRegression()
```

```
regressor.fit(X_train, y_train)
```

In the above code, we have used a fit() method to fit our Simple Linear Regression object to the training set. In the fit() function, we have passed the x_train and y_train, which is our training dataset for the dependent and an independent variable. We have fitted our regressor object to the training set so that the model can easily learn the correlations between the predictor and target variables.

Step: 3. Prediction of test set result:

dependent (salary) and an independent variable (Experience). So, now, our model is ready to predict the output for the new observations. In this step, we will provide the test dataset (new observations) to the model to check whether it can predict the correct output or not.

We will create a prediction vector y_pred, which will contain predictions of test dataset, and prediction of training set respectively.

```
# Predicting the Test set results
```

```
y_pred = regressor.predict(X_test)
```

Step: 4. visualizing the Training set results:

Now in this step, we will visualize the training set result. To do so, we will use the `scatter()` function of the `pyplot` library, which we have already imported in the pre-processing step. The `scatter()` function will create a scatter plot of observations.

In the x-axis, we will plot the Years of Experience of employees and on the y-axis, salary of employees. In the function, we will pass the real values of training set, which means a year of experience `x_train`, training set of Salaries `y_train`, and color of the observations. Here we are taking a green color for the observation, but it can be any color as per the choice.

Now, we need to plot the regression line, so for this, we will use the `plot()` function of the `pyplot` library. In this function, we will pass the years of experience for training set, predicted salary for training set `x_pred`, and color of the line.

Next, we will give the title for the plot. So here, we will use the `title()` function of the `pyplot` library and pass the name ("Salary vs Experience (Training Dataset)").

After that, we will assign labels for x-axis and y-axis using `xlabel()` and `ylabel()`

function. # Visualising the Training set results

```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

In the above plot, we can see the real values observations in green dots and predicted values are covered by the red regression line. The regression line shows a correlation between the dependent and independent variable.

The good fit of the line can be observed by calculating the difference between actual values and predicted values. But as we can see in the above plot, most of the observations are close to the regression line, hence our model is good for the training set.

Step: 5. visualizing the Test set results:

In the previous step, we have visualized the performance of our model on the training set. Now, we will do the same for the Test set. The complete code will remain the same as the above code, except in this, we will use `x_test`, and `y_test` instead of `x_train` and `y_train`

Visualising the Test set results

```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
```



```
plt.title('Salary vs Experience (Test set)')  
plt.xlabel('Years of Experience')
```

```
plt.ylabel('Salary')
```

```
plt.show()
```

2. Multiple linear regression

Multiple Linear Regression is one of the important regression algorithms which models the linear relationship between a single dependent continuous variable and more than one independent variable.

Example:

Prediction of CO2 emission based on engine size and number of cylinders in a car.

Some key points about MLR:

- ▢ For MLR, the dependent or target variable(Y) must be the continuous/real, but the predictor or independent variable may be of continuous or categorical form.
- ▢ Each feature variable must model the linear relationship with the dependent variable.
- ▢ MLR tries to fit a regression line through a multidimensional space of data-points.

The multiple regression equation explained above takes the following form:

$$y = b_1x_1 + b_2x_2 + \dots + b_nx_n +$$

c. Where,

Y= Output/Response variable

$b_0, b_1, b_2, b_3, b_n, \dots$ = Coefficients of the model.

$x_1, x_2, x_3, x_4, \dots$ = Various Independent/feature variable

Assumptions for Multiple Linear Regression:

- ▢ A linear relationship should exist between the Target and predictor variables.
- ▢ The regression residuals must be normally distributed.
- ▢ MLR assumes little or no multicollinearity (correlation between the independent variable) in

data. Implementation of Multiple Linear Regression model using Python:

To implement MLR using Python, we have below problem:

Problem Description:

We have a dataset of 50 start-up companies. This dataset contains five main information: R&D Spend, Administration Spend, Marketing Spend, State, and Profit for a financial year. Our goal is to create a model that can easily determine which company has a maximum profit, and which is the most affecting factor for the profit of a company.

Since we need to find the Profit, so it is the dependent variable, and the other four variables are independent variables. Below are the main steps of deploying the MLR model:

1. Data Pre-processing Steps
2. Fitting the MLR model to the training set
3. Predicting the result of the test

set Step-1: Data Pre-processing Step:

The very first step is data pre-processing

, which we have already discussed in this tutorial. This process contains the below steps:

- ▢ Importing libraries: Firstly, we will import the library which will help in building the model. Below is the code for it:

Importing the

libraries import numpy

as np

import matplotlib.pyplot as plt

import pandas as pd

- ▢ Importing dataset: Now we will import the dataset(50_CompList), which contains all the variables. Extracting dependent and independent variables from it.

Importing the dataset

dataset = pd.read_csv('50_Startups.csv')

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, 4]

- ▢ Convert the column into categorical

columns

states=pd.get_dummies(X['State'],drop_first=True)

- ▢ Drop the state

column

X=X.drop('State',axis=1)

- ▢ concat the dummy

variables

X=pd.concat([X,states],axis=1)

- ▢ Now we will split the dataset into training and test

set. # Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

Step: 2- Fitting our MLR model to the Training set:

Now, we have well prepared our dataset in order to provide training, which means we will fit our regression model to the training set. It will be similar to as we did in Simple Linear Regression model.

Fitting Multiple Linear Regression to the Training

set from sklearn.linear_model import

LinearRegression regressor = LinearRegression()

regressor.fit(X_train, y_train)

Step: 3- Prediction of Test set results:

The last step for our model is checking the performance of the model. We will do it by predicting the test set result. For prediction, we will create a `y_pred` vector.

```
# Predicting the Test set results
```

```
y_pred = regressor.predict(X_test)
```

Now, checking the final results

```
In [43]: print('Train score:', regressor.score(X_train, y_train))
print('Test score', regressor.score(X_test, y_test))
```

```
Train score: 0.9499572530324031
Test score 0.9393955917820569
```

The above score tells that our model is 95% accurate with the training dataset and 93% accurate with the test dataset.

Applications of Multiple Linear Regression:

There are mainly two applications of Multiple Linear Regression:

- Effectiveness of Independent variable on prediction:
- Predicting the impact of changes:

3. Logistic linear regression

Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.

In the simplest case there are two outcomes, which is called binomial, an example of which is predicting if a tumor is malignant or benign. Other cases have more than two outcomes to classify, in this case it is called multinomial. A common example for multinomial logistic regression would be predicting the class of an iris flower between 3 different species.

Here we will be using basic logistic regression to predict a binomial variable. This means it has only two possible outcomes.

Example:

```
import numpy
from sklearn import linear_model
#Reshaped for Logistic function.
X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1,1)
y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
logr = linear_model.LogisticRegression()
logr.fit(X,y)
#predict if tumor is cancerous where the size is 3.46mm:
predicted = logr.predict(numpy.array([3.46]).reshape(-1,1))
print(predicted)
```

Q) Generalized Linear Models

Answer: Multiple linear regression and simple linear regression

Q) Support Vector Machines

Answer: Already covered in previous units

Chapter-III

Binary Classification: Multiclass/Structured outputs, MNIST, Ranking.






Binary Classification

It is a process or task of classification, in which a given data is being classified into two classes. It's basically a kind of prediction about which of two groups the thing belongs to.

Let us suppose, two emails are sent to you, one is sent by an insurance company that keeps sending their ads, and the other is from your bank regarding your credit card bill. The email service provider will classify the two emails, the first one will be sent to the spam folder and the second one will be kept in the primary one.

This process is known as binary classification, as there are two discrete classes, one is spam and the other is primary. So, this is a problem of binary classification.

Binary classification uses some algorithms to do the task, some of the most common algorithms used by binary classification are .




-  Logistic Regression
-  k-Nearest Neighbors
-  Decision Trees
-  Support Vector Machine
-  Naive Bayes

Multiclass

Classification

Multi-class classification is the task of classifying elements into different classes. Unlike binary, it doesn't restrict itself to any number of classes.

Examples of multi-class classification are

-  classification of news in different categories,
-  classifying books according to the subject,
-  classifying students according to their streams etc.

In these, there are different classes for the response variable to be classified in and thus according to the name, it is a Multi-class classification.

Can a classification possess both binary or multi-class?

Let us suppose we have to do sentiment analysis of a person, if the classes are just "positive" and "negative", then it will be a problem of binary class. But if the classes are "sadness", "happiness", "disgusting", "depressed", then it will be called a problem of Multi-class classification.

Binary vs Multiclass Classification

Parameters	Binary classification	Multi-class classification
No. of classes	It is a classification of two groups, i.e. classifies objects in at most two classes. The most popular algorithms used by the binary classification are-	There can be any number of classes in it, i.e., classifies the object into more than two classes. Popular algorithms that can be used for multi-class classification include:
Algorithms used	<ul style="list-style-type: none">• Logistic Regression• k-Nearest Neighbors• Decision Trees• Support Vector Machine• Naive Bayes	<ul style="list-style-type: none">• k-Nearest Neighbors• Decision Trees• Naive Bayes• Random Forest.• Gradient Boosting
Examples	Examples of binary classification include- <ul style="list-style-type: none">• Email spam detection (spam or not).• Churn prediction (churn or not).• Conversion prediction (buy or not).	Examples of multi-class classification include: <ul style="list-style-type: none">• Face classification.• Plant species classification.• Optical character recognition.

Q) MNIST

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems.

The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets.

The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%.

Extended MNIST (EMNIST) is a newer dataset developed and released by NIST to be the (final) successor to MNIST.[11][12] MNIST included images only of handwritten digits. EMNIST includes all the images from

NIST Special Database 19, which is a large database of handwritten uppercase and lower case letters as well as digits. The images in EMNIST were converted into the same 28x28 pixel format, by the same process, as were the MNIST images. Accordingly, tools which work with the older, smaller, MNIST dataset will likely work unmodified with EMNIST.

Q) Ranking

A binary classification system involves a system that generates ratings for each occurrence, which, by ordering them, are turned into rankings, which are then compared to a threshold. Occurrences with rankings above the threshold are declared positive, and occurrences below the threshold are declared negative.