Amrita Vishwa Vidyapeetham

College Of Engineering,

Bengaluru

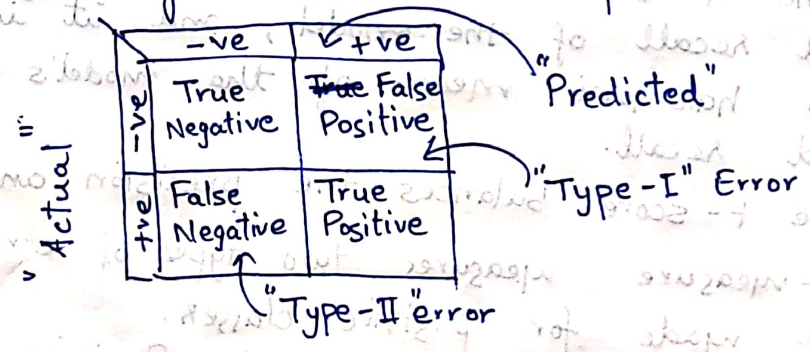
Name: Raman Kumar
Program: Mtech DS
Course: Machine Learning
Registration number: BL.EN.P2DSC22009

Contents:-

Assignment 02

(1) Confusion Matrix for binary-Class Scenario.

- Binary class produces outputs with only 2-label values. example (Yes or No), (1/0) (+ve/-ve) for the given data.
- We use the observed classes to compare with predicted classes for the different evaluation performances after classification.
- Confusion matrix table describe the performance metrices on a set of test data, for which the positive values or true values are known.
- Confusion matrix label will be exactly the same if the performance of classifier will be perfect.
- It has four outcomes
  - True Positive : Correct +ve prediction
  - True Negative : Incorrect +ve prediction
  - False Positive : correct -ve prediction
  - False Negative : Incorrect -ve prediction



- Accuracy: is calculated as the number of all correct predictions divided by total number of dataset

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

- Recall: It is sensitivity (or) true positive rate. When it's actually positive, how often does it predict positive?

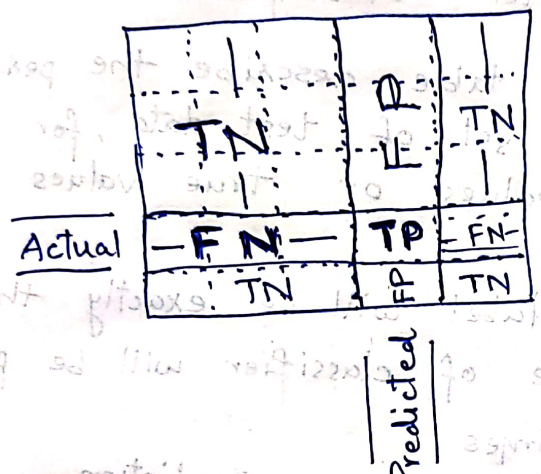
Scanned with OKEN Scanner

Recall = $\frac{TP}{TP+FP}$ Actual Positive $\frac{TP}{TP+FN}$

- **Precision:** is calculated as number of correct positive predictions divided by total no. of positive predictions.

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

Confusion matrix for multiclass-classification:



Q.2). Beta-variation in F-measure:

The F-score is a way of combining the precision and recall of the model, and it is defined as harmonic mean of the model's precision and recall.

- The F-score balances the precision and recall.
- F-measure measures two types of errors that are made for positive classes.
  1) Minimizing FP maximizing Precision
  2) Minimizing FN, maximizing Recall.

F-score provide a way to combine both precision & recall into single measure.

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

- In F-β measure, the coefficient - beta control the balance of precision & recalled

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

Three common values of beta - parameter are -

1) F-0.5 score ($\beta = 0.5$) : More weight on precision, less weight on recall.

2) F-1 score ($\beta = 1$) : Balance the weight on precision & recall.

3). F-2 score ($\beta = 2$) : Less weight on precision, more weight on recall.

(Q.3). **Overfitting in Decision Trees :**

- Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points.
  - h : hypothesis error
  - we let error on training data : $error_{train}(h)$ error over the entire distribution D of data : $error_D(h)$.
  - when hypothesis 'h' overfits the training data, if there is an alternative hypothesis h'

$$error_{train}(h) < error_{train}(h')$$

$$error_D(h) < error_D(h')$$

Causes of Overfitting :

- overfitting due to presence of Noise
- Overfitting due to lack of representative instances
- Overfitting & multiple comparision procedure

Underfitting in Decision Trees :

- Learning algorithm had the opportunity to learn more from training data, but didn't, that comes under underfitting.
  - gts when model is too simple for the data.

Causes: . Less no. of features, or small dataset.

Scanned with OKEN Scanner

(Q.4) OR gate for following question is :
truth table

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

- Initial weights are $(-2, 0.75, 0.63)$
- First row of truth table, the inputs are $(0,0)$ output is 0.

output = 1 if $[(-2 * 0) + (0.75 * 0) + (0.63 * 1)] \geq 0$, else 0

output = 0

- Target output and observed output are difficult,

∴ update the weights :

$\Delta W_0 = \alpha * |\text{predicted values} - \text{observed value}| * \text{input}$

$= 0.1 * |0-0| * 0 = 0$

$W_0 \leftarrow W_0 + \Delta W_0 = -2 + 0 = -2$

- Change in weight of W_1 is :

$\Delta W_1 = 0.1 * |0-0| * 0 = 0$

$W_1 \leftarrow 0.75 + 0 = 0.75$

- Change in weight of W_2 is :

$\Delta W_2 = 0.1 * |0-0| * 1 = 0$

$W_2 \leftarrow W_2 + \Delta W_2 = 0.63 + 0 = 0.63$

- For the 2nd row of truth table,
inputs : (0, 1) , outputs : 1 , weights $(-2, 0.75, 0.63)$

- output = 1 , if $((-2 * 0) + (0.75 * 0) + (0.63 * 1) \geq 0$
output = 0 ; else 0

∴ Since the target value & observed output are different , update the weights.

- change in weight for W_0 is :

$\Delta W_0 = 0.1 * |1-0| * 0 = 0$

$W_0 \leftarrow W_0 + \Delta W_0 = -2 + 0 = -2$

Scanned with OKEN Scanner

- The change in weight W_1 is :

$$\Delta W_1 = 0.1 * |1-0| * 0 = 0$$

$$W_1 \leftarrow W_1 + \Delta W_1 = 0.75 + 0 = 0.75$$

- change in weight W_2 is :

$$\Delta W_2 = 0.1 * |1-0| * 0 = 0.1$$

$$W_2 \leftarrow W_2 + \Delta W_2 = 0.63 + 0.1 = 0.73$$

- Third row of truth table

inputs = (1, 0), target output = (1), weights = (-2, 0.75, 0.73)

- output = 1, if $(-2 * 1) + (0.75 * 0) + (0.73 * 1) \geq 0$; else 0

output = 1

Since the target value & observed output are same. now no need to update the weights.

- For 4th row input (1, 1), output = 1.

Scanned with OKEN Scanner

(Q.5). Trace a decision tree :

- Information gain : The goal is to maximize the information gain at each split, which means we want to select the feature that results in the most homogeneous subgroups.

- Entropy of label (sex) at root node.

Entropy : = $-(p(M) * \log_2(p(M)) + p(F) * \log_2(p(F)))$

$p(M)$: proportions of males

$p(F)$: proportions of females

here, 4 males & 3 females

Entropy = $-(4/7 * \log_2(4/7) + 3/7 * \log_2(3/7))$

= 0.985

- IG on each feature i.e. entropy of subgroups.

- height feature : → three subgroups

High	Medium	Low (Entropy)
$= -(\frac{1}{1} * \log_2(\frac{1}{1}))$	$-(\frac{2}{2} * \log_2(\frac{2}{2}))$	$= -(\frac{1}{2} * \log_2(\frac{1}{2}) + (\frac{1}{2}) * \log_2(\frac{1}{2}))$
$= 0$	$= -(1 * 0)$	$= 1$
	$= 0$	

IG = Entropy (root) − (weighted average of entropies of subgroups)

$= 0.985 - (\frac{1}{7} * 0 + \frac{2}{7} * 0 + \frac{4}{7} * 1)$

$= 0.985 - \frac{4}{7} = 0.169$

- We can repeat the process for the other features to calculate their IG.

* weight : $IG = 0.985 - (\frac{1}{7} * 0 + \frac{3}{7} * 0.918 + \frac{3}{7} * 0)$

$= 0.151$

Hair length = $IG = 0.985 - (\frac{2}{7} * 0.918 + \frac{5}{7} * 0)$

$= \underline{0.043}$.

Scanned with OKEN Scanner

- Based on the calculations, highest IG feature is height.

- **Gini Index:** at root node,

$$\text{Gini} = 1 - (p(M)^2 + p(F)^2)$$

$$= 1 - \left(\frac{4}{7}\right)^2 + \left(\frac{3}{7}\right)^2 = 0.48$$

- For height feature:

High: $\text{Gini} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$

Med: $\text{Gini} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$

low: $\text{Gini} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$

$$\text{Gini} = \left(\frac{1}{7}\right) * 0 + \left(\frac{2}{7} * 0.5\right) + \left(\frac{4}{7} * 0.5\right) = 0.429$$

- Weight feature:

$$\text{Gini} = \left(\frac{1}{7} * 0\right) + \left(\frac{3}{7} * 0.5\right) + \left(\frac{3}{7} * 0\right) = 0.286$$

$$\text{Hair length} = \left(\frac{2}{7} * 0.5\right) + \left(\frac{5}{7} * 0\right) = 0.143$$

Gini index of ~~weight~~ 'height' is 'lowest'.

Scanned with OKEN Scanner

(6). To calculate the parameters 'm' & 'c' for a line of the form $Y = mX + c$ using linear regression, you can use the following steps:

1. Calculate the mean of the X values & the mean of Y values.
2. Calculate the difference between each X value and the mean of the X values, and the difference between each Y value and the mean of the Y values.
3. Calculate the sum of the products of the differences calculated in step 2.
4. ~~Divide~~ Calculate the sum of the squares of the differences in the X values.
5. Divide the sum from step 3 by the sum from step 4. The result is the value of m.
6. Calculate the value of 'c' by substituting the mean of the X values and the mean of Y values into the equation $Y = mX + c$ & solving for c.

Scanned with OKEN Scanner

• data = [(1, 1.75), (2, 2.14), (2.5, 2.47), (3, 2.38), (3.5, 3.55), (4, 2.72), (5, 2.91), (4.5, 4.2)]

Step 1:

```
x_mean = sum(x for x, y in data)/len(data)
y_mean = sum(y for x, y in data)/len(data)
```

Step 2:

```
xy_diff = [(x - x_mean) * (y - y_mean) for x, y in data]
x_sq_diff = [(x - x_mean) ** 2 for x, y in data]
```

Step 3:

```
xy_diff_sum = sum(xy_diff)
```

Step 4:

```
x_sq_diff_sum = sum(x_sq_diff)
```

Step 5:

```
m = xy_diff_sum / x_sq_diff_sum
```

Step 6:

```
c = y_mean - m * x_mean
printf(f"m = {m:.1f}")
printf(f"c = {c:.1f}")
```

⇒ m = 0.4 c = 1.3

Scanned with OKEN Scanner

(8). In SVM, the kernal function plays a crucial role in mapping the input data into a higher-dimensional space, where it becomes possible to perform linear classification. This is important because many datasets are not linearly seperable in their original feature space, but they might be linearly seperable in a higher-dimensional space.

- e.g., suppose we have a dataset with two classes, A and B, and we want to build an SVM classifier to seperate the two classes. If the data is not linearly seperable in the original feature space, we can apply a kernal function to map the data into a higher dimensional space where it becomes linearly seperable.
- Types of kernal with SVM:
  - linear kernal
  - polynomial kernal
  - radial basis kernal (RBF)
- For multi-class classification, there are several ways in which SVM's can be used. One approach is to build a one-versus-all (OVA) classifier, in which a seperate SVM is trained for each class, with that class as the positive class and all other classes as the negative class.
- Another approach is to use a one-versus-one (OVO) classifier, in which a seperate SVM is trained for each pair of classes.

Scanned with OKEN Scanner

(7.)

Support Vectors	Y_i * α_i	Class Values
[1.7 0.5]	-0.4485	-1
[1.9 0.2]	-1	-1
[1.9 0.4]	-1	-1
[3.3 1]	0.436651	1
[5.1 1.6]	0.160646	1
[3 1.1]	1	1

Intercept (b) = -6.8

Vector, A = [7 3]

Class Value for test vector X

= Sign(Sum (y_i * alpha_i * DOT_PRODUCT (Supportvectors, X)) + b)

[Support vectors, X] ⇒ Dot product of support vector and X.

$$= \begin{bmatrix} 1.7 & 0.5 \\ 1.9 & 0.2 \\ 1.9 & 0.4 \\ 3.3 & 1 \\ 5.1 & 1.6 \\ 3 & 1.1 \end{bmatrix}_{6\times2} \cdot \begin{bmatrix} 7 \\ 3 \end{bmatrix}_{2\times1} = \begin{bmatrix} 13.4 \\ 13.9 \\ 14.5 \\ 26.1 \\ 40.5 \\ 24.3 \end{bmatrix}_{6\times1}$$

Applying the formula, sum(y_i * alpha_i * Dot_Product(SupportVector, X))

$$= \begin{bmatrix} -0.4485 & -1 & -1 & 0.436651 & 0.160646 & 1 \end{bmatrix} \begin{bmatrix} 13.4 \\ 13.9 \\ 14.5 \\ 26.1 \\ 40.5 \\ 24.3 \end{bmatrix}$$

$$= (0.4485 \times 13.4) + (-1 \times 13.9) + (-1 \times 14.5) + (0.43665 \times 26.1) + (0.160646 \times 40.5) + (1 \times 24.3)$$

$$= 7.792828$$

X = 7.792828 + intercept = 7.792828 - 6.8 = 0.992828

Sign is +ve.

∴ Class value is +1

Scanned with OKEN Scanner