WEEK- 7: **Evaluation Metrics for Classification.**

Session No.6

Evaluation Metrics for Classification

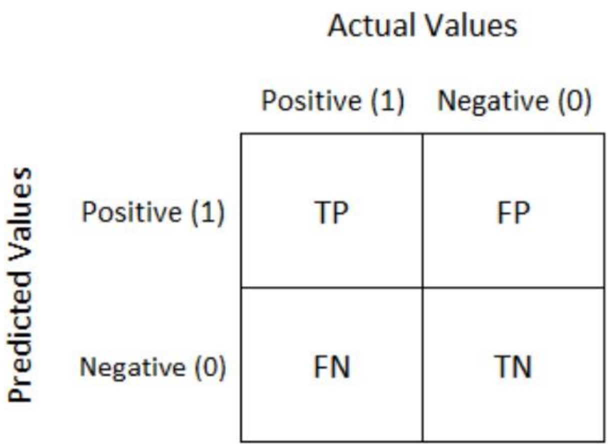
- The most important task in building any machine learning model is to evaluate itsperformance.
- Evaluation metrics are tied to machine learning tasks.

- Using different metrics for performance evaluation, we should be able to improve our model's overall predictive power before we roll it out for production on unseen data.
- Without doing a proper evaluation of the Machine Learning model by using different evaluation metrics, and only depending on accuracy, can lead to a problem when the respective model is deployed on unseen data and may end in poor predictions.

Confusion Matrix

Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values.

A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known.



It is extremely useful for measuring the Recall, Precision, Accuracy, and AUC-ROC curves. **True Positive:** We predicted positive and it's true. In the image, we predicted that a woman ispregnant and she actually is.

True Negative: We predicted negative and it's true. In the image, we predicted that a man isnot pregnant and he actually is not.

False Positive (Type 1 Error)- We predicted positive and it's false. In the image, we predictedthat a man is pregnant but he actually is not.

False Negative (Type 2 Error)- We predicted negative and it's false. In the image, wepredicted that a woman is not pregnant but she actually is.

Accuracy

Accuracy simply measures how often the classifier correctly predicts. We can define accuracyas the ratio of the number of correct predictions and the total number of predictions.

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} = \frac{\boldsymbol{No\ of\ Correct\ Predictions}}{\boldsymbol{Total\ no\ of\ predictions}}$$

- When any model gives an accuracy rate of 99%, you might think that model is performing very good but this is not always true and can be misleading in some situations.
- Accuracy is useful when the target class is *well balanced* but is not a good choice for the unbalanced classes.gine the scenario where we had 99 images of the dog and only 1 image of a cat present in our training data. Then our model would always predict the dog, and therefore we got 99% accuracy.
- In reality, Data is always imbalanced for example Spam email, credit card fraud, and medical diagnosis.
- Hence, if we want to do a better model evaluation and have a full picture of the model evaluation, other metrics such as recall and precision should also be considered

Precision

- Precision explains how many of the correctly predicted cases actually turned out to bepositive.
- Precision is useful in the cases where False Positive is a higher concern than False Negatives.
- The importance of Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn and this could be harmful to the business.
- **Precision for a label is defined as the number of true positives divided by the number of predicted positives.**

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- Precision is very useful when you have a model that starts some kind of business workflow(e.g. marketing campaigns) when it predicts 1.
- So, you want your model to be as correct as possible when it says 1 and don't care too muchwhen it predicts 0.
- Precision is very much used in marketing campaigns, because a marketing automation campaign is supposed to start an activity on a user when it predicts that they will respond successfully. That's why we need high precision, which is the probability that our model is correct when it predicts 1.
- Low values for precision will make our business lose money, because we are contacting customers that are not interested in our commercial offer.

Recall (Sensitivity)

- Recall explains how many of the actual positive cases we were able to predict correctly with our model.
- It is a useful metric in cases where False Negative is of higher concern than False Positive. It is important in medical cases where it doesn't matter whether we raise a false alarm butthe actual positive cases should not go undetected.
- **Recall for a label is defined as the number of true positives divided by the total number of actual positives.**

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- Recall is used when you have to correctly classify some event that has already occurred.
- For example, fraud detection models must have a high recall in order to detect fraudsproperly.
- In such situations, we don't care about the real 0s, because we are interested only inspotting the real 1s as often as possible.
- So, we're working with the second row of the confusion matrix.
- Common uses of recall are, as said, fraud detection models or even disease detection ona patient. If somebody is ill, we need to spot their illness avoiding the false negatives. A

false negative patient may become contagious and it's not safe. That's why, when we have to spot an event that already occurred, we need to work with recall.

Specificity (True negative rate)

- Specificity (SP) is calculated as the number of correct negative predictions divided by thetotal number of negatives. It is also called true negative rate (TNR).

$$Specificity = \frac{TN}{TN + FP}$$

- This metric is of interest if you are concerned about the accuracy of your negative rate andthere is a high cost to a positive outcome.
- Let's say we wanted to send a handwritten note to the family of each passenger who diedas identified by our model for titanic dataset.
- Since the Titanic sunk in 1912, we feel the families have had time to heal from their lossand so would not be distraught by receiving a note.
- However, we feel it would be incredibly insensitive to send a note to a family of a survivor, as their death would have been more recent (the last Titanic survivor died in 2009 at age97) and the family would still be grieving.

F1 Score

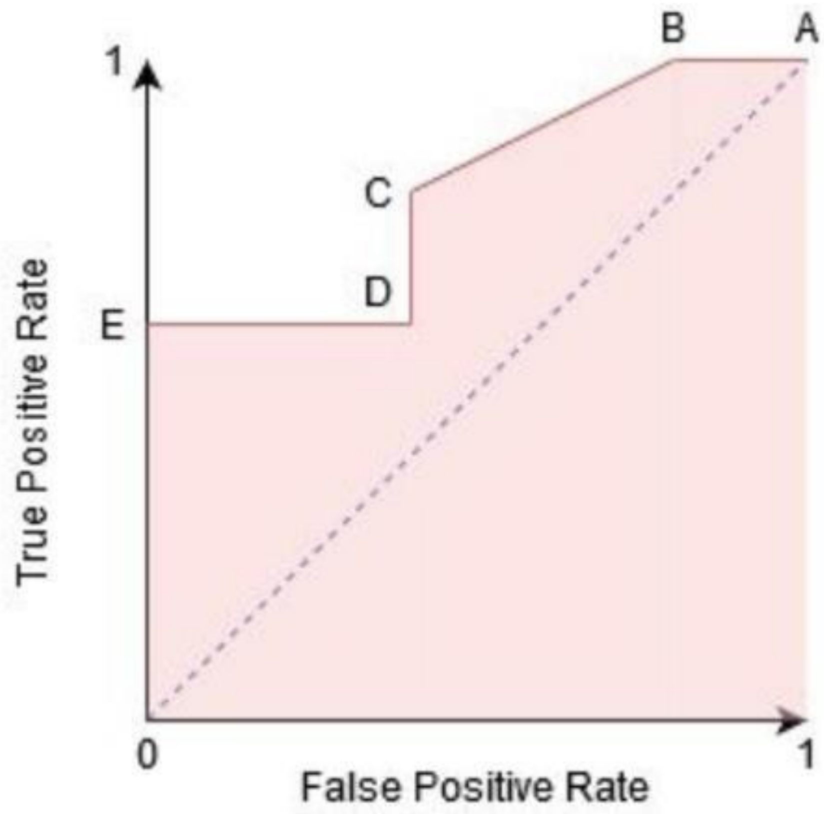
- It gives a combined idea about Precision and Recall metrics. It is maximum whenPrecision is equal to Recall.
- **F1 Score is the harmonic mean of precision and recall.**

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

- The F1 score punishes extreme values more. F1 Score could be an effective evaluationmetric in the following cases:
  - When FP and FN are equally costly.
  - Adding more data doesn't effectively change the outcome
  - True Negative is high

AUC-ROC

- The Receiver Operator Characteristic (ROC) is a probability curve that plots the TPR(True Positive Rate) against the FPR(False Positive Rate) at various threshold values and separatesthe 'signal' from the 'noise'.
- The **Area Under the Curve (AUC)** is the measure of the ability of a classifier to distinguishbetween classes. From the graph, we simply say the area of the curve ABDE and the X andY-axis.
- From the graph shown below, the greater the AUC, the better is the performance of the model at different threshold points between positive and negative classes.
- This simply means that When AUC is equal to 1, the classifier is able to perfectly distinguishbetween all Positive and Negative class points.
- When AUC is equal to 0, the classifier would be predicting all Negatives as Positives and vice versa.
- When AUC is 0.5, the classifier is not able to distinguish between the Positive and Negativeclasses.



In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df = pd.read_csv("C:\\Users\\maths\\aiml\\Train.csv")
df.head()
```

Out[2]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel78
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	

5 rows × 785 columns

In [6]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.iloc[:,1:],df.iloc[:,0],test_size=0.2,random_state=2)
```

In [7]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [8]:

```
clf1 = LogisticRegression()
clf2 = DecisionTreeClassifier()
```

In [9]:

```
clf1.fit(X_train,y_train)
clf2.fit(X_train,y_train)
```

```
C:\Users\maths\AppData\Roaming\Python\Python39\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbf gs failed to converge (status=1): STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
 https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[9]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [10]:

```
y_pred1 = clf1.predict(X_test)
y_pred2 = clf2.predict(X_test)
```

In [11]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy of Logistic Regression",accuracy_score(y_test,y_pred1))
print("Accuracy of Decision Trees",accuracy_score(y_test,y_pred2))
```

```
Accuracy of Logistic Regression 0.9145238095238095
Accuracy of Decision Trees 0.8552380952380952
```

In [12]:

```
print("Logistic Regression Confusion Matrix\n")
pd.DataFrame(confusion_matrix(y_test,y_pred1),columns=list(range(0,10)))
```

Logistic Regression

Confusion Matrix Out[12]:

	0	1	2	3	4	5	6	7	8	9
0	791	0	1	0	1	9	11	1	6	1
1	0	938	2	4	0	3	0	3	12	0
2	2	9	745	16	9	4	13	13	15	3
3	1	4	19	763	0	36	5	5	20	11
4	2	5	4	0	804	1	7	3	9	21
5	9	3	4	23	10	610	15	10	34	11
6	12	4	12	1	7	8	789	0	5	1
7	3	2	13	6	4	2	0	812	2	29
8	11	12	13	18	1	21	6	6	694	11
9	3	6	2	14	25	4	0	37	7	736

In [13]:

```
print("Decision Tree Confusion Matrix\n")

pd.DataFrame(confusion_matrix(y_test,y_pred2),columns=list(range(0,10)))
```

Decision Tree Confusion

Matrix Out[13]:

	0	1	2	3	4	5	6	7	8	9
0	755	1	10	8	11	12	7	3	8	6
1	2	907	12	9	6	6	2	6	10	2
2	9	18	669	35	13	10	19	20	27	9

3	9	14	30	690	7	44	9	16	26	19
4	5	4	12	3	742	8	13	7	14	48
5	14	4	10	35	9	584	22	9	23	19
6	8	1	11	5	16	19	756	2	15	6
7	4	9	24	12	6	5	1	774	10	28

8 11 10 24 32 17 35 11 9 622 22 **9** 7 5 8 13 41 19 5 31 20 685

In [14]:

```
from sklearn.metrics import precision_score,recall_score,f1_score
```

In [15]:

```
precision_score(y_test,y_pred1,average='weighted')
```

Out[15]:

0.91421507

13630827

In [16]:

```
recall_score(y_test,y_pred1,average='weighted')
```

Out[16]:

0.91452380

95238095

In [17]:

```
f1_score(y_test,y_pred1,average='weighted')
```

Out[17]:

0.9142794994052751

In [18]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred1))
```

```
              precision    recall  f1-score    support

0       0.95       0.96       0.96        821
1       0.95       0.98       0.96        962
2       0.91       0.90       0.91        829
3       0.90       0.88       0.89        864
4       0.93       0.94       0.94        856
5       0.87       0.84       0.85        729
6       0.93       0.94       0.94        839
7       0.91       0.93       0.92        873
8       0.86       0.88       0.87        793             9        0.89        0.88        0.89
834

    accuracy
0.91       8400    macro avg        0.91
0.91       0.91        8400 weighted avg
0.91        0.91        0.91        8400
```

```
In [ ]:
```