Performance Measures for Classification

Classification and Regression

- Two main types of machine learning models:
- Classification and regression
- Both Regression and Classification algorithms are known as Supervised Learning algorithms
- Both models are used to predict outcomes based on input variables and work with labeled datasets
- They differ in their objectives and output formats.

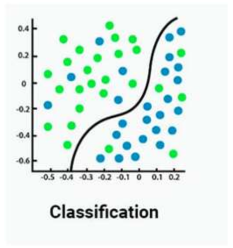
Classification Models

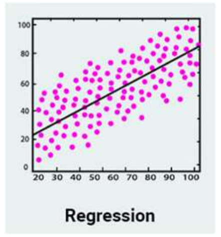
- Classification models are used when the outcome variable is categorical or binary
- The objective of a classification model is to assign new observations to one of the predefined categories based on the input variables.
- Ex: classification problems include spam detection, image recognition, and sentiment analysis.
- The output of a classification model is a probability or score indicating the likelihood of a new observation belonging to each category.

Regression Models

- Regression models are used when the outcome variable is continuous or numeric, meaning it can take any value within a certain range.
- The objective of a regression model is to predict the value of the outcome variable based on the input variables.
- Ex: Regression problems include stock price prediction, sales forecasting, and weather forecasting.
- The output of a regression model is a numerical value indicating the predicted outcome variable.

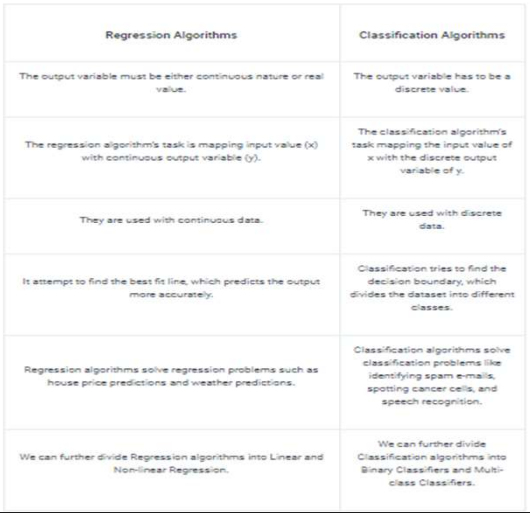
Classification vs Regression





Classification vs Regression

- Regression finds correlations between dependent and independent variables.
- The Regression algorithm’s task is finding the mapping function so we can map the input variable of “x” to the continuous output variable of “y.”
- So regression algorithms help predict continuous variables such as house prices, market trends, weather patterns, oil and gas prices
- Classification is an algorithm that finds functions that help divide the dataset into classes based on various parameters.
- Classification algorithms find the mapping function to map the “x” input to “y” discrete output.
- Classification algorithms are used for email and spam classification, predicting the willingness of bank customers to pay their loans, and identifying cancer tumor cells.

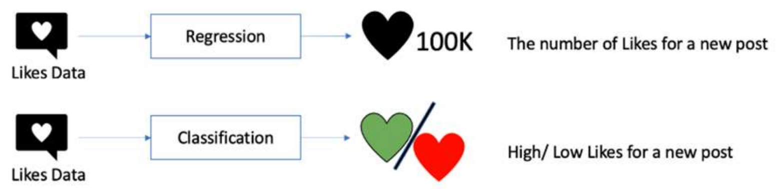


Regression Algorithms	Classification Algorithms
The output variable must be either continuous nature or real value.	The output variable has to be a discrete value.
The regression algorithm's task is mapping input value (x) with continuous output variable (y).	The classification algorithm's task mapping the input value of x with the discrete output variable of y.
They are used with continuous data.	They are used with discrete data.
It attempt to find the best fit line, which predicts the output more accurately.	Classification tries to find the decision boundary, which divides the dataset into different classes.
Regression algorithms solve regression problems such as house price predictions and weather predictions.	Classification algorithms solve classification problems like identifying spam e-mails, spotting cancer cells, and speech recognition.
We can further divide Regression algorithms into Linear and Non-linear Regression.	We can further divide Classification algorithms into Binary Classifiers and Multi-class Classifiers.

Types of Regression and classification

- Types of Regression
- Simple linear, Multiple linear, Polynomial, Support vector machine, Decision tree
- Types of Classification
- Logistic regression, Naïve Bayes, KNN, Support vector machine Decision tree

A common Application for both Classification and Regression Models


Likes Data
Regression
100K
The number of Likes for a new post
Likes Data
Classification
High/ Low Likes for a new post

A common Application for both Classification and Regression Models

- A problem that can be solved using both regression and classification models is predicting the number of likes on a tweet.
- The regression model can be used to predict the exact number of likes a tweet will receive based on input variables such as the length of the tweet, the use of certain keywords or hashtags, and the time of day it was posted.
- The classification model can be used to categorize tweets into high or low engagement categories based on the number of likes they are likely to receive.
- Where is it used?
- Classification model can be useful for social media marketers who want to quickly identify the most engaging tweets and focus their efforts on those tweets that are likely to receive a high number of likes.
- The regression model can provide more precise predictions for specific tweets that are tailored to a particular audience or marketing campaign.

Evaluation metrics to evaluate the performance of classification and regression models

- In classification models, we can use metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to evaluate the performance of the model.
- In regression models, we can use metrics such as mean squared error (MSE), root mean squared error (RMSE), and R-squared to evaluate the performance of the model.

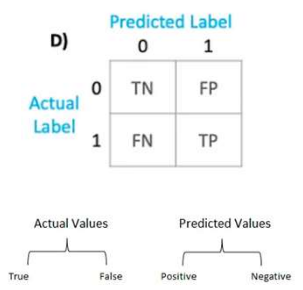
Confusion matrix

- In machine Learning, Classification is the process of categorizing a given set of data into different categories.
- A confusion matrix is a performance evaluation tool representing the accuracy of a classification model.
- It displays the number of true positives, true negatives, false positives, and false negatives.
- A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes.
- The matrix compares the actual target values with those predicted by the machine learning model.
- This gives a holistic view of how well our classification model is performing and what kinds of errors it is making.

Confusion matrix

- For a binary classification problem, we would have a 2 x 2 matrix with 4 values
- The matrix displays the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) produced by the model on the test data.
- For multi-class classification, the matrix shape will be equal to the number of classes i.e for N classes it will be N x N.

Scikit-learn Confusion matrix with 1(true), 0(false) labels in data



True Positive: Interpretation: You predicted positive and it's true.

True Negative: Interpretation: You predicted negative and it's true.

False Positive: (Type 1 Error): Interpretation: You predicted positive and it is false.

False Negative: (Type 2 Error): Interpretation: You predicted negative and it is false.

Note: We describe predicted values as Positive and Negative and actual values as True and False

Confusion matrix

Actual	Predicted	Outcome
1	1	TP
0	0	TN
0	0	TN
1	1	TP
0	0	TN
0	0	TN
1	0	FN
0	1	FP
0	0	TN
1	0	FN

10 instances
TP = 2
TN = 5
FP = 1
FN = 2
Usage of confusion matrix in scikit-learn
confusion_matrix(actual,predicted, labels=[0, 1, 2])
labels is intended for multiclass use

confusion_matrix(actual,predicted) is same as confusion_matrix(actual,predicted, labels=[0, 1])
Setting labels=[0, 1] implies that negative class is labelled 0 and positive class is called 1

Setting labels=[1, 0] implies that positive class is labelled 0 and negative class is called 1

Classification Report

- actual = [1,0,0,1,0,0,1,0,0,1] # 1 is positive , 0 is negative, Support: Actual positive 4, actual negative 6, total = 10
- predicted = [1,0,0,1,0,0,0,1,0,0]
- Accuracy = TP+TN/(TP+FP+TN+FN)= 2+5/(2+1+5+2)=7/10=0.7
- Precision = TP/(TP+FP) = 2/(2+1) = 2/3 = 0.67
- Recall = TP/(TP+FN) = 2/(2+2)=2/4 = 0.5
- F1-Score = 2*(Recall * Precision)/(Recall+Precision) = 2* (0.5*0.67)/(0.5+0.67) = 0.67/1.16 = 0.57
- Specificity = TN/(TN+FP)= 5/(5+1) = 0.833
- Scikit-learn Output of classification report: Consider blue highlighted values – 1 is positive , 0 is negative as per our sample input
- Support: Actual positive 4, actual negative 6, total = 10

```
Confusion matrix :
 [[5 1]
 [2 2]]
Outcome values :
 5 1 2 2
Classification report :
              precision    recall  f1-score   support

           1       0.67      0.50      0.57         4
           0       0.71      0.83      0.77         6

    accuracy                           0.70        10
   macro avg       0.69      0.67      0.67        10
weighted avg       0.70      0.70      0.69        10
```

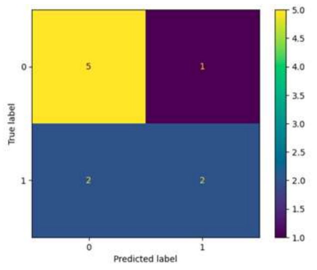
Confusion Matrix Using Scikit-learn

```
# confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# actual values
actual = [1,0,0,1,0,0,1,0,0,1]
# predicted values
predicted = [1,0,0,1,0,0,0,1,0,0]

# confusion matrix
matrix = confusion_matrix(actual,predicted)
print('Confusion matrix : \n',matrix)

# outcome values order in sklearn
tn, fp, fn, tp = confusion_matrix(actual,predicted).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
```

```
# classification report for precision, recall f1-score and accuracy
matrix = classification_report(actual,predicted)
print('Classification report : \n',matrix)

#individual reports
Accuracy = metrics.accuracy_score(actual, predicted)
Precision = metrics.precision_score(actual,predicted)
Sensitivity_recall = metrics.recall_score(actual, predicted)
Specificity = metrics.recall_score(actual, predicted, pos_label=0)
F1_score = metrics.f1_score(actual, predicted)

#metrics:
print({"Accuracy":Accuracy,"Precision":Precision,"Sensitivity_recall":Sensitivity_recall,"Specificity":Specificity,"F1_score":F1_score})
```

Confusion Matrix Using Scikit-learn

```
Confusion matrix :
 [[5 1]
 [2 2]]
Outcome values :
 5 1 2 2
Classification report :
              precision    recall  f1-score   support

           1       0.67      0.50      0.57         4
           0       0.71      0.83      0.77         6

    accuracy                           0.70        10
   macro avg       0.69      0.67      0.67        10
weighted avg       0.70      0.70      0.69        10
```



```
{'Accuracy': 0.7, 'Precision': 0.6666666666666666,
'Sensitivity_recall': 0.5, 'Specificity': 0.8333333333333334,
'F1_score': 0.5714285714285715}
```

Confusion Matrix Using Scikit-learn

- Sklearn confusion_matrix() returns the values of the Confusion matrix. The output is, however, slightly different. It takes the rows as Actual values and the columns as Predicted values.
- Sklearn classification_report() outputs precision, recall, and f1-score, for each target class. In addition to this, it also has some extra values: micro avg, macro avg, and weighted avg
  - Support: The support is the number of occurrences of each class in actual value (ie y_test)

Confusion Matrix Using Scikit-learn

Mirco average is the precision/recall/f1-score calculated for all the classes.

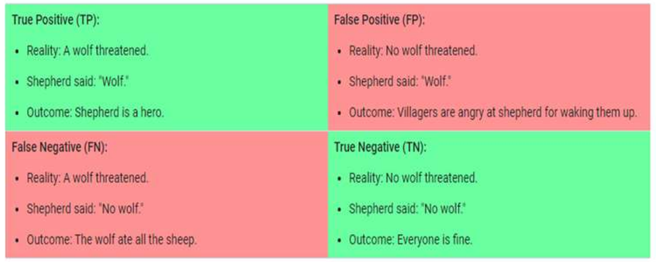
$$\text{Micro avg Precision} = \frac{TP1 + TP2}{TP1 + TP2 + FP1 + FP2}$$

Macro average is the average of precision/recall/f1-score.

$$\text{Macro avg Precision} = \frac{P1 + P2}{2}$$

Weighted average is just the weighted average of precision/recall/f1-score.

Confusion Matrix - Summary



True Positive (TP): • Reality: A wolf threatened. • Shepherd said: "Wolf." • Outcome: Shepherd is a hero.	**False Positive (FP):** • Reality: No wolf threatened. • Shepherd said: "Wolf." • Outcome: Villagers are angry at shepherd for waking them up.
False Negative (FN): • Reality: A wolf threatened. • Shepherd said: "No wolf." • Outcome: The wolf ate all the sheep.	**True Negative (TN):** • Reality: No wolf threatened. • Shepherd said: "No wolf." • Outcome: Everyone is fine.

A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class.

A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class.

Implementation of TN, FP, FN , TP

```
def find_TP(y, y_hat):
   # counts the number of true positives (y = 1, y_hat = 1)
   return sum((y == 1) & (y_hat == 1))
def find_FN(y, y_hat):
   # counts the number of false negatives (y = 1, y_hat = 0) Type-II
error
   return sum((y == 1) & (y_hat == 0))
def find_FP(y, y_hat):
   # counts the number of false positives (y = 0, y_hat = 1) Type-I error
   return sum((y == 0) & (y_hat == 1))
def find_TN(y, y_hat):
   # counts the number of true negatives (y = 0, y_hat = 0)
   return sum((y == 0) & (y_hat == 0))
```

```
TP = find_TP(y, y_hat)
FN = find_FN(y, y_hat)
FP = find_FP(y, y_hat)
TN = find_TN(y, y_hat)
print('TP:',TP)
print('FN:',FN)
print('FP:',FP)
print('TN:',TN)
precision = TP/(TP+FP)
print('Precision:',precision)
recall = recall_score(y, y_hat)
print('Recall: %f' % recall)
# F1_score =
2*Precision*Recall/Precision+Recall
f1_score =
2*((precision*recall)/(precision+rec
all))
print('F1 score: %f' % f1_score)
```

Problem Statement -1

- A classification dataset with 1000 data points is fed to a classifier (say logistic regression or decision tree) and the below confusion matrix is obtained:
- True Positive (TP) = 560
- True Negative (TN) = 330
- False Positive (FP) = 60
- False Negative (FN) = 50

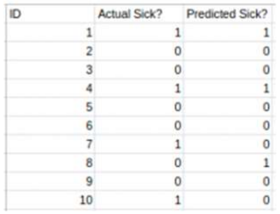
Give the interpretation

Problem Statement -1

- A classification dataset with 1000 data points is fed to a classifier (say logistic regression or decision tree) and get the below confusion matrix:
- Give the interpretation
- True Positive (TP) = 560, meaning the model correctly classified 560 positive class data points.
- True Negative (TN) = 330, meaning the model correctly classified 330 negative class data points.
- False Positive (FP) = 60, meaning the model incorrectly classified 60 negative class data points as belonging to the positive class.
- False Negative (FN) = 50, meaning the model incorrectly classified 50 positive class data points as belonging to the negative class.

Problem Statement -2

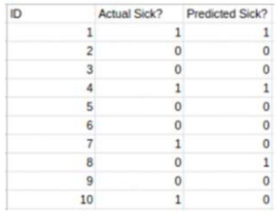
We need to predict how many people are infected with a contagious virus before they show the symptoms and isolate them from the healthy population . The two values for our target variable would be Sick and Not Sick
Assign Confusion matrix



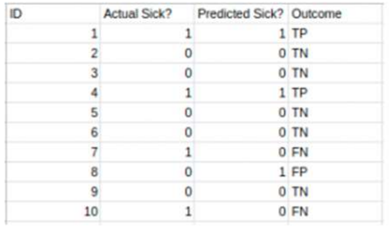
ID	Actual Sick?	Predicted Sick?
1	1	1
2	0	0
3	0	0
4	1	1
5	0	0
6	0	0
7	1	0
8	0	1
9	0	0
10	1	0

Problem Statement -2

We need to predict how many people are infected with a contagious virus before they show the symptoms and isolate them from the healthy population . The two values for our target variable would be Sick and Not Sick
Assign Confusion matrix



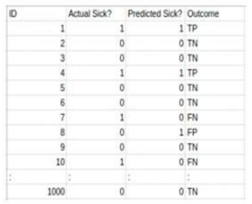
ID	Actual Sick?	Predicted Sick?
1	1	1
2	0	0
3	0	0
4	1	1
5	0	0
6	0	0
7	1	0
8	0	1
9	0	0
10	1	0



ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FN
8	0	1	FP
9	0	0	TN
10	1	0	FN

Problem Statement -3

- We need to predict how many people are infected with a contagious virus before they show the symptoms and isolate them from the healthy population. The two values for our target variable would be Sick and Not Sick
- Show that accuracy is not a proper indicator
- The total outcome values are:
- TP = 30, TN = 930, FP = 30, FN = 10



ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	TP
2	0	0	TN
3	0	0	TN
4	1	1	TP
5	0	0	TN
6	0	0	TN
7	1	0	FN
8	0	1	FP
9	0	0	TN
10	1	0	FN
⋮	⋮	⋮	⋮
1000	0	0	TN

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

Problem Statement -3

- We need to predict how many people are infected with a contagious virus before they show the symptoms and isolate them from the healthy population . The two values for our target variable would be Sick and Not Sick
- Show that accuracy is not a proper indicator
- The total outcome values are:
- TP = 30, TN = 930, FP = 30, FN = 10

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

$$Accuracy = \frac{30+930}{30+30+930+10} = 0.96$$

Problem Statement -3

- Interpretation of accuracy:
- Our model is saying, accuracy = 96%, means "It can predict sick people 96% of the time".
- However, it is doing the opposite. It predicts the people who will not get sick with 96% accuracy while the sick are spreading the virus!

- This is not a correct metric for our model. The data set is imbalanced
- Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e one class label has a very high number of observations and the other has a very low number of observations (30 & 930)
- When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop down to 60%
- Should measure how many positive cases we can predict correctly to arrest the spread of the contagious virus
- Should measure, out of the correct predictions, how many are positive cases to check the reliability of our model

- This is where we need the dual concept of Precision and Recall

Balanced vs Imbalanced data sets

- Balanced datasets are those which have approximately the same number of positive and negative values.

- Imbalanced datasets or Skewed datasets are those which have a very high difference in the number of positive and negative values.

Precision vs Recall

- Precision:
- The confusion matrix can be used to calculate a variety of metrics, such as accuracy, precision, recall, and F1 score.
- TP = 30, TN = 930, FP = 30, FN = 10
- Precision tells us how many of the correctly predicted cases actually turned out to be positive.
- Precision = TP / {TP+FP}
- Precision = 30/(30+30) = 0.5
- 50% percent of the correctly predicted cases turned out to be positive cases.
- Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.
- Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.

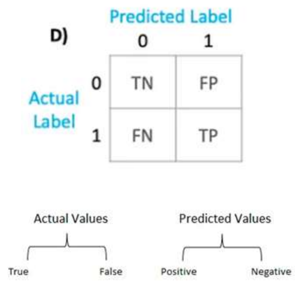
Precision vs Recall

- Recall:
- Recall tells us how many of the actual positive cases we were able to predict correctly with our model.
- Recall = TP / {TP+FN}
- Recall = 30/(30+10) = 75%
- 75% of the positives were successfully predicted by our model.
- Recall is a useful metric in cases where False Negative trumps False Positive.
- Recall is important in medical cases where it does not matter whether we raise a false alarm, but the actual positive cases should not go undetected
- In our example, Recall would be a better metric
- Should not accidentally discharge an infected person
- So accuracy was a bad metric for our model.

F1-Score

- F1-Score:
- What if there is no clear distinction between whether Precision is more important or Recall.
- Combine them- The F1-score captures both the trends in a single value
- Increase the precision of our model, the recall goes down, and vice-versa.
- F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.
- F1-Score = 2*(Recall * Precision) / (Recall + Precision)
- The interpretability of the F1-score is poor.
- This means that we do not know what our classifier is maximizing – precision or recall.
- So, use it in combination with other evaluation metrics, giving a complete picture of the result.

Scikit-learn Confusion matrix with 1(true), 0(false) labels in data



True Positive: Interpretation: You predicted positive and it's true.

True Negative: Interpretation: You predicted negative and it's true.

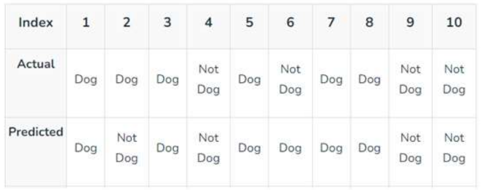
False Positive: (Type 1 Error): Interpretation: You predicted positive and it's false.

False Negative: (Type 2 Error): Interpretation: You predicted negative and it's false.

Note: We describe predicted values as Positive and Negative and actual values as True and False

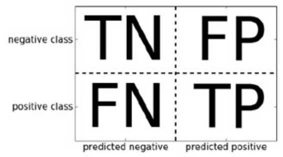
Problem Statement -4

Problem: For the image recognition having a Dog image or Not Dog image, construct confusion matrix and derive all classification metrics.



Index	1	2	3	4	5	6	7	8	9	10
Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog

Confusion Matrix in Machine Learning



True Positive (TP): It is the total counts having both predicted and actual values are Dog (1-1 case)

True Negative (TN): It is the total counts having both predicted and actual values are Not Dog (0-0 case).

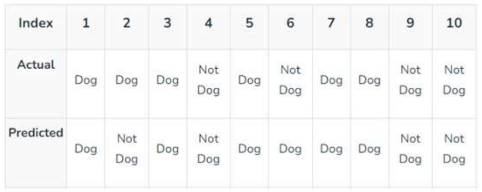
False Positive (FP): It is the total counts having prediction is Dog while actually Not Dog (0-1 case).

False Negative (FN): It is the total counts having prediction is Not Dog while actually, it is Dog (1-0 case).

if the model incorrectly (or falsely) predicts positive class, it is a false positive (FP).
If the model incorrectly (or falsely) predicts negative class, it is a false negative (FN)
The diagonal elements represent the true positive (TP) and true negative (TN) cases, while the off-diagonal elements represent the false positive (FP) and false negative (FN) cases

Problem Statement -4

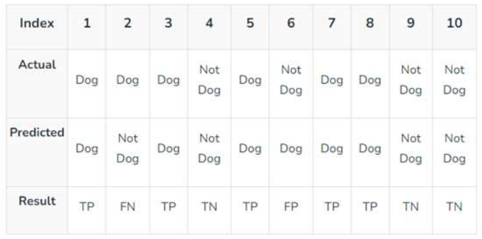
Problem: For the image recognition having a Dog image or Not Dog image, mark result as TP, TN, FP, FN under each column and get their total counts to construct confusion matrix



Index	1	2	3	4	5	6	7	8	9	10
Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog

Problem Statement -4

For the following data set,



Index	1	2	3	4	5	6	7	8	9	10
Actual	Dog	Dog	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Not Dog	Not Dog
Predicted	Dog	Not Dog	Dog	Not Dog	Dog	Dog	Dog	Dog	Not Dog	Not Dog
Result	TP	FN	TP	TN	TP	FP	TP	TP	TN	TN

Actual Dog Counts = 6

Actual Not Dog Counts = 4

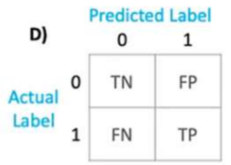
True Positive Counts = 5

False Positive Counts = 1

True Negative Counts = 3

False Negative Counts = 1

D)



		Predicted Label	
		0	1
Actual Label	0	TN	FP
	1	FN	TP

Confusion Matrix in Machine Learning

```
#Import the necessary libraries
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

#Create the NumPy array for actual and predicted labels.
actual    = np.array(
  ['Dog','Dog','Dog','Not Dog','Dog','Not
Dog','Dog','Dog','Not Dog','Not Dog'])
predicted = np.array(
  ['Dog','Not Dog','Dog','Not
Dog','Dog','Dog','Dog','Dog','Not Dog','Not Dog'])
```

```
#compute the confusion matrix.
cm = confusion_matrix(actual,predicted)
 tn, fp, fn, tp = cm.ravel()
print (tn, fp, fn, tp)
#Plot the confusion matrix.
sns.heatmap(cm,
        annot=True,
        fmt='g',
        xticklabels=['Dog','Not Dog'],
        yticklabels=['Dog','Not Dog'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```

Problem Statement 4 - Accuracy

- From the confusion matrix, we can find the following metrics
- Accuracy: Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.
- From all the classes (positive and negative), how many of them we have predicted correctly? Accuracy should be high as possible.
- Accuracy = {TP+TN} / {TP+TN+FP+FN}
- For the above case:
- Accuracy = (5+3)/(5+3+1+1) = 8/10 = 0.8

Problem Statement 4 - Precision

- Precision: Precision is a measure of how accurate a model's positive predictions are.
- It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model
- From all the classes we have predicted as positive, how many are actually positive? Precision should be high as possible
- Precision = TP / {TP+FP}
- For the above case:
- Precision = 5/(5+1) =5/6 = 0.8333

Problem Statement 4 - Recall

- Recall: Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset.
- It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.
- From all the positive classes, how many we predicted correctly? Recall should be high as possible.
- Recall = TP / {TP+FN}
- For the above case:
- Recall = 5/(5+1) =5/6 = 0.8333

Problem Statement 4 - F1-Score

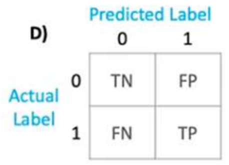
- It is difficult to compare two models with low precision and high recall or vice versa.
- So to make them comparable, we use F-Score.
- F-score helps to measure Recall and Precision at the same time.
- It uses Harmonic Mean in place of Arithmetic Mean by punishing the extreme values more.
- F1-Score: F1-score is used to evaluate the overall performance of a classification model.
- It is the harmonic mean of precision and recall
- F1-Score = {2 * Precision * Recall} / {Precision + Recall}
- For the above case:
- F1-Score: = (2* 0.8333* 0.8333)/(0.8333+ 0.8333) = 0.8333

Summary

- **Accuracy:** (TP + TN) / (TP + TN + FP + FN)
- **Precision:** TP / (TP + FP)
- **Recall (Sensitivity or True Positive Rate):** TP / (TP + FN)
- **Specificity (True Negative Rate):** TN / (TN + FP)
- **F1-Score:** 2 * (Precision * Recall) / (Precision + Recall)

Problem Statement 5

- cm= confusion_matrix([1, 1, 1, 0, 1, 0, 1, 1, 0, 0], [1,0, 1, 0, 1, 1, 1, 1, 0, 0])
- print (cm)
- tn, fp, fn, tp = confusion_matrix([1, 1, 1, 0, 1, 0, 1, 1, 0, 0], [1,0, 1, 0, 1, 1, 1, 1, 0, 0]).ravel()
- print (tn, fp, fn, tp)
- Output:
- [[3 1]
- [1 5]]
- 3 1 1 5



Problem Statement 6

- Draw the confusion matrix

- actual = [1, 0, 1, 0, 1, 1, 0, 0, 1, 0]
- # Predicted outcomes
- predicted = [1, 0, 1, 0, 1, 0, 1, 1, 1, 0]
- We assign Result=[TP, TN, TP, TN, TP, FN,FP, FP, TP, TN]
- We get count TN =3, FP = 2, FN= 1, TP = 4

Problem Statement 6- Python

```
from sklearn.metrics import confusion_matrix
# Actual outcomes
actual = [1, 0, 1, 0, 1, 1, 0, 0, 1, 0]
# Predicted outcomes
predicted = [1, 0, 1, 0, 1, 0, 1, 1, 1, 0]
# Create a confusion matrix
conf_matrix = confusion_matrix(actual, predicted)
print("Confusion Matrix:")
print(conf_matrix)
```

Problem Statement 6 - Output

```
Confusion Matrix:
[[3 2]
 [1 4]]
```

AUC-ROC Curve

- **AUC-ROC"Area Under the Curve" (AUC) of the "Receiver Operating Characteristic" (ROC)**
- **AUROC (Area under Receiver operating characteristics curve).**

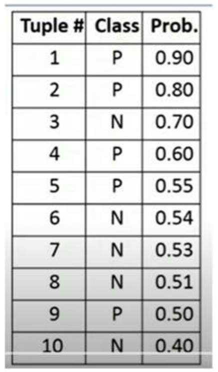
$$TPR = \frac{TP}{TP+FN} \qquad FPR = \frac{FP}{FP+TN}$$

- TPR/Recall/Sensitivity = TP/TP +FN   Specificity = TN/TN+FP
- FPR = 1-Specificity= FP/TN+FP
- Intuitively TPR/recall corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points. In other words, the higher the TPR, the fewer positive data points we will miss.
- Intuitively FPR/fallout corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points. In other words, the higher the FPR, the more negative data points we will misclassify.
- AUC-ROC Curve is plotted with TPR against the FPR where TPR is on y-axis and FPR is on the x-axis.

AUC-ROC Curve

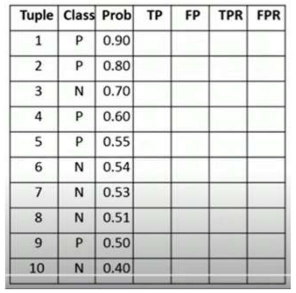
- AUC-ROC curve is a performance measurement for the classification problem at various thersholds settings.
- ROC is a probability curve and AUC represents the degree or measure of separability.
- ROC (Receiver Operating Characteristic) curve tells us about how good the model can distinguish between two things(e.g. If a patient has a disease or no).
- Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

Draw ROC curve for the following problem definition



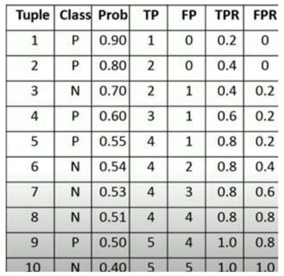
Tuple #	Class	Prob.
1	P	0.90
2	P	0.80
3	N	0.70
4	P	0.60
5	P	0.55
6	N	0.54
7	N	0.53
8	N	0.51
9	P	0.50
10	N	0.40

Draw ROC curve for the following problem definition

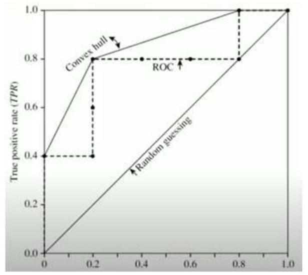


Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90				
2	P	0.80				
3	N	0.70				
4	P	0.60				
5	P	0.55				
6	N	0.54				
7	N	0.53				
8	N	0.51				
9	P	0.50				
10	N	0.40				

Draw ROC curve for the following problem definition



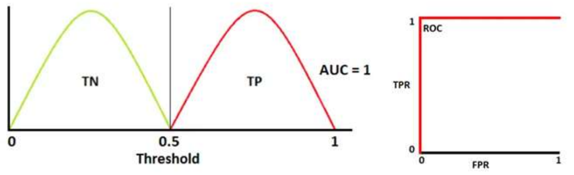
Tuple	Class	Prob	TP	FP	TPR	FPR
1	P	0.90	1	0	0.2	0
2	P	0.80	2	0	0.4	0
3	N	0.70	2	1	0.4	0.2
4	P	0.60	3	1	0.6	0.2
5	P	0.55	4	1	0.8	0.2
6	N	0.54	4	2	0.8	0.4
7	N	0.53	4	3	0.8	0.6
8	N	0.51	4	4	0.8	0.8
9	P	0.50	5	4	1.0	0.8
10	N	0.40	5	5	1.0	1.0



FPR on X axis and TPR on Y axis

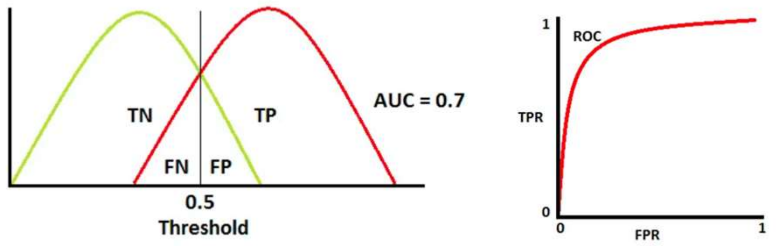
AUC-ROC Curve

- Red distribution curve is of the positive class (patients with disease) and the green distribution curve is of the negative class(patients with no disease)
- When two curves do not overlap, model has an ideal measure of separability. It is perfectly able to distinguish between positive class and negative class



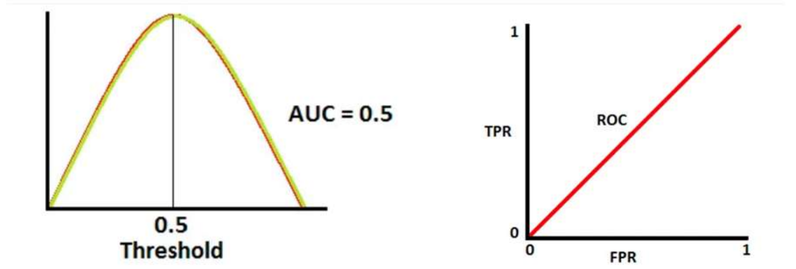
AUC-ROC Curve

When two distributions overlap, we introduce type 1 and type 2 errors.
When AUC is 0.7, it means there is a 70% chance that the model will be able to distinguish between positive class and negative class



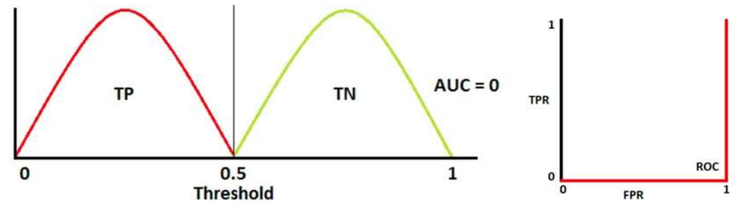
AUC-ROC Curve

This is the worst situation.
When AUC is approximately 0.5, the model has no discrimination capacity to distinguish between positive class and negative class.



AUC-ROC Curve

When AUC is approximately 0, the model is actually reciprocating the classes.
It means the model is predicting a negative class as a positive class and vice versa.



AUC-ROC Curve

- When AUC is equal to 1, the classifier is able to perfectly distinguish between all Positive and Negative class points.
- When AUC is equal to 0, the classifier would be predicting all Negatives as Positives and vice versa.
- When AUC is 0.5, the classifier is not able to distinguish between the Positive and Negative classes