Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

Assignment No. 1

Course Outcomes (CO):

CO1	Apply models for reasoning with uncertainty using unreliable information.
CO2	Apply the key principles and techniques in building a cognitive application
CO3	Apply fuzzy logic in knowledge-based systems
CO4	Apply deep learning concepts and architectures to build and train models
CO5	Apply metrics to measure the performance of various learning algorithms
CO6	Summarize the trends in Data Science for Multimodal application

Problem Statement:
Discuss the Evaluation Metric of Receiver Operating Characteristic (ROC). Illustrate a use case where specifically ROC should be used and why ? Use a dataset to Implement and Evaluate multiple classifiers for using python. You can use any dataset for its implementation. **(BL5, CO5)**

Solution:

Assessing and Comparing Classifier Performance with ROC Curves

The most commonly reported measure of classifier performance is accuracy: the percent of correct classifications obtained.

This metric has the advantage of being easy to understand and makes comparison of the performance of different classifiers trivial, but it ignores many of the factors which should be taken into account when honestly assessing the performance of a classifier.

What Is Meant By Classifier Performance?

Classifier performance is more than just a count of correct classifications.

Consider, for interest, the problem of screening for a relatively rare condition such as cervical cancer, which has a prevalence of about 10% (actual stats). If a lazy Pap smear screener was to classify every slide they see as “normal”, they would have a 90% accuracy. But that figure completely ignores the fact that the 10% of women who do have the disease have not been diagnosed at all.

Some Performance Metrics

Most classifiers produce a score, which is checked against threshold to decide the classification. If a classifier produces a score between 0.0 (definitely negative) and 1.0 (definitely positive), it

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

is common to consider anything over 0.5 as positive.

However, any threshold applied to a dataset (in which PP is the positive population and NP is the negative population) is going to produce true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) (Figure 1). We need a method which will take into account all of these numbers.

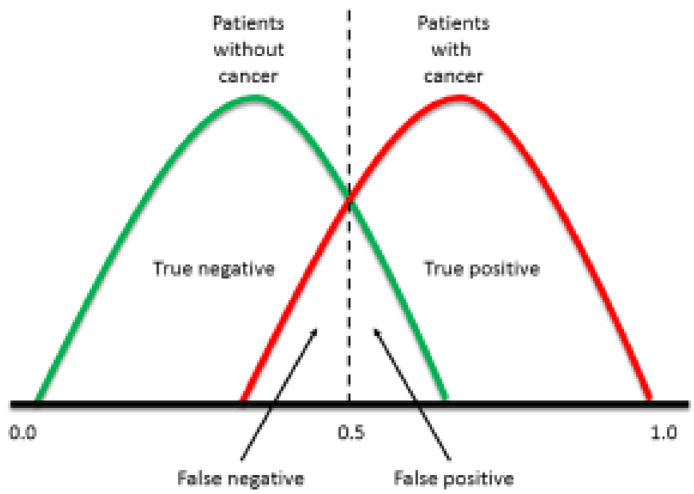


Figure 1. Overlapping datasets will always generate false positives and negatives as well as true positives and negatives

Once you have numbers for all of these measures, some useful metrics can be calculated.

- **Accuracy** = (1 – Error) = (TP + TN)/(PP + NP) = Pr(C), the probability of a correct classification.
- **Sensitivity** = TP/(TP + FN) = TP/PP = the ability of the test to detect disease in a population of diseased individuals.
- **Specificity** = TN/(TN + FP) = TN / NP = the ability of the test to correctly rule out the disease in a disease-free population.

Let's calculate these metrics for some reasonable real-world numbers. If we have 100,000 patients, of which 200 (20%) actually have cancer, we might see the following test results (Table 1):

	Test Positive	Test Negative	Total
Patient Diseased	160	40	200
Patient Healthy	29940	69860	99800
Total	30100	69900	100000

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

For this data:

- **Sensitivity** = TP/(TP + FN) = 160 / (160 + 40) = 80.0%
  Specificity = TN/(TN + FP) = 69,860 / (69,860 + 29,940) = 70.0%

In other words, our test will correctly identify 80% of people with the disease, but 30% of healthy people will incorrectly test positive. By only considering the sensitivity (or accuracy) of the test, potentially important information is lost.

By considering our wrong results as well as our correct ones we get much greater insight into the performance of the classifier.

One way to overcome the problem of having to choose a cutoff is to start with a threshold of 0.0, so that every case is considered as positive. We correctly classify all of the positive cases, and incorrectly classify all of the negative cases. We then move the threshold over every value between 0.0 and 1.0, progressively decreasing the number of false positives and increasing the number of true positives.

TP (sensitivity) can then be plotted against FP (1 – specificity) for each threshold used. The resulting graph is called a Receiver Operating Characteristic (ROC) curve (Figure 2). ROC curves were developed for use in signal detection in radar returns in the 1950's, and have since been applied to a wide range of problems.

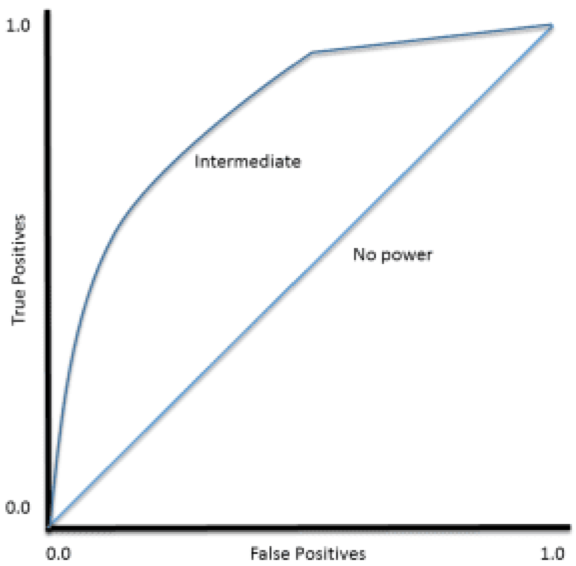


Figure 2. Examples of ROC curves

For a perfect classifier the ROC curve will go straight up the Y axis and then along the X axis. A classifier with no power will sit on the diagonal, whilst most classifiers fall somewhere in

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

between.

Using ROC Curves

Threshold Selection

It is immediately apparent that a ROC curve can be used to select a threshold for a classifier which maximises the true positives, while minimising the false positives.

However, different types of problems have different optimal classifier thresholds. For a cancer screening test, for example, we may be prepared to put up with a relatively high false positive rate in order to get a high true positive, it is most important to identify possible cancer sufferers.

For a follow-up test after treatment, however, a different threshold might be more desirable, since we want to minimise false negatives, we don't want to tell a patient they're clear if this is not actually the case.

Performance Assessment

ROC curves also give us the ability to assess the performance of the classifier over its entire operating range. The most widely-used measure is the area under the curve (AUC). As you can see from Figure 2, the AUC for a classifier with no power, essentially random guessing, is 0.5, because the curve follows the diagonal. The AUC for that mythical being, the perfect classifier, is 1.0. Most classifiers have AUCs that fall somewhere between these two values.

An AUC of less than 0.5 might indicate that something interesting is happening. A very low AUC might indicate that the problem has been set up wrongly, the classifier is finding a relationship in the data which is, essentially, the opposite of that expected. In such a case, inspection of the entire ROC curve might give some clues as to what is going on: have the positives and negatives been mislabelled?

Classifier Comparison

The AUC can be used to compare the performance of two or more classifiers. A single threshold can be selected and the classifiers' performance at that point compared, or the overall performance can be compared by considering the AUC.

Implementation Gaussian Naïve Bayes, Support Vector Machine and Decision Tree for Breast Cancer Detection for evaluating its performance using metric of ROC-AUC Plot

Code:

```
# Implementation Gaussian Naïve Bayes, Support Vector Machine and Decision Tree for Breast Cancer Detection for evaluating its performance using metric of ROC-AUC Plot

# Importing the required libraries

import numpy as np
```

Academic Year: 2025-26
Class / Branch/ Div: BE- IT-C
Semester: VII
Subject: AI & DS-II

Name: Zahra Surve
Student ID: 23204002

```
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

# Load the datasets
cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

# splitting X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
  X, y, test_size=0.3, random_state=1)

# GAUSSIAN NAIVE BAYES
gnb = GaussianNB()
# train the model
gnb.fit(X_train, y_train)
# make predictions
```

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

```
gnb_pred = gnb.predict(X_test)
# print the accuracy
print("Accuracy of Gaussian Naive Bayes: ",
  accuracy_score(y_test, gnb_pred))
# print other performance metrics
print("Precision of Gaussian Naive Bayes: ",
  precision_score(y_test, gnb_pred, average='weighted'))
print("Recall of Gaussian Naive Bayes: ",
  recall_score(y_test, gnb_pred, average='weighted'))
print("F1-Score of Gaussian Naive Bayes: ",
  f1_score(y_test, gnb_pred, average='weighted'))

# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test, gnb_pred)
roc_auc = auc(fpr, tpr)

# Plot ROC curve for GAUSSIAN NAIVE BAYES
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve for GAUSSIAN NAIVE BAYES')
plt.legend(loc='lower right')
```

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

```
plt.show()

# DECISION TREE CLASSIFIER
dt = DecisionTreeClassifier(random_state=0)
# train the model
dt.fit(X_train, y_train)
# make predictions
dt_pred = dt.predict(X_test)
# print the accuracy
print("Accuracy of Decision Tree Classifier: ",
  accuracy_score(y_test, dt_pred))
# print other performance metrics
print("Precision of Decision Tree Classifier: ",
  precision_score(y_test, dt_pred, average='weighted'))
print("Recall of Decision Tree Classifier: ",
  recall_score(y_test, dt_pred, average='weighted'))
print("F1-Score of Decision Tree Classifier: ",
  f1_score(y_test, dt_pred, average='weighted'))

# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test, dt_pred)
roc_auc = auc(fpr, tpr)

# Plot ROC curve for DECISION TREE CLASSIFIER
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
```

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

```
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve for DECISION TREE CLASSIFIER')
plt.legend(loc='lower right')
plt.show()


# SUPPORT VECTOR MACHINE
svm_clf = svm.SVC(kernel='linear') # Linear Kernel
# train the model
svm_clf.fit(X_train, y_train)
# make predictions
svm_clf_pred = svm_clf.predict(X_test)
# print the accuracy
print("Accuracy of Support Vector Machine: ",
  accuracy_score(y_test, svm_clf_pred))
# print other performance metrics
print("Precision of Support Vector Machine: ",
  precision_score(y_test, svm_clf_pred, average='weighted'))
print("Recall of Support Vector Machine: ",
  recall_score(y_test, svm_clf_pred, average='weighted'))
print("F1-Score of Support Vector Machine: ",
  f1_score(y_test, svm_clf_pred, average='weighted'))
```

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

```
# Compute ROC curve
fpr, tpr, _ = roc_curve(y_test, svm_clf_pred)
roc_auc = auc(fpr, tpr)

# Plot ROC curve for SUPPORT VECTOR MACHINE
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve for SUPPORT VECTOR MACHINE')
plt.legend(loc='lower right')
plt.show()
```

Output:

Accuracy of Gaussian Naive Bayes: 0.8596491228070176

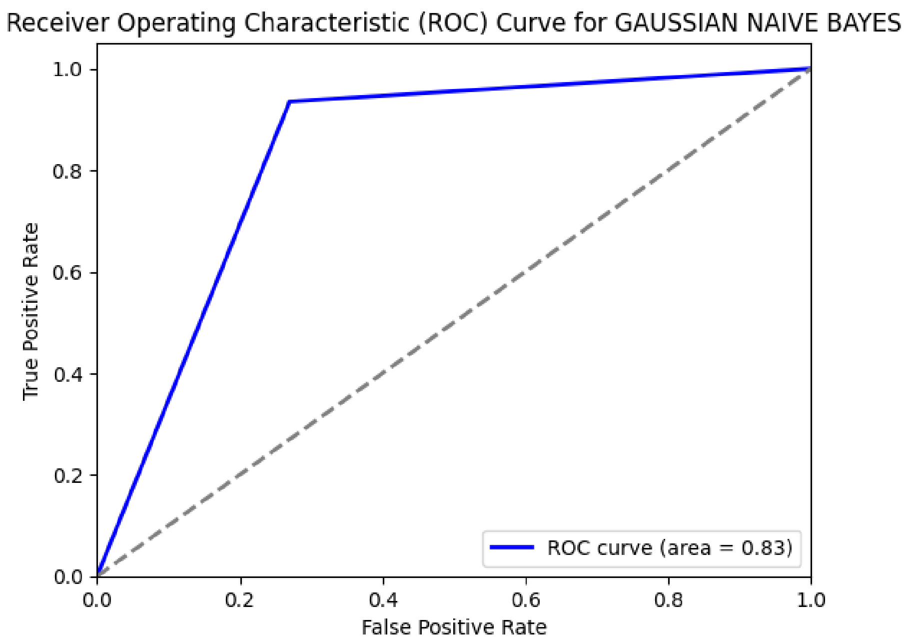
Precision of Gaussian Naive Bayes: 0.8603504283574303

Recall of Gaussian Naive Bayes: 0.8596491228070176

F1-Score of Gaussian Naive Bayes: 0.8567046239339575

Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

Name: Zahra Surve **Student ID: 23204002**

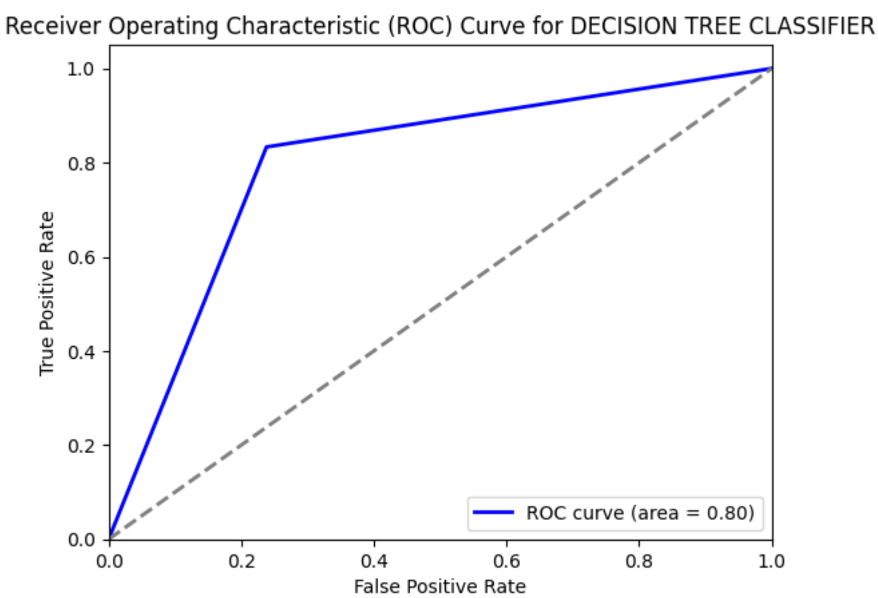


Accuracy of Decision Tree Classifier: 0.8070175438596491

Precision of Decision Tree Classifier: 0.8092959671907041

Recall of Decision Tree Classifier: 0.8070175438596491

F1-Score of Decision Tree Classifier: 0.8079024945265226



Academic Year: 2025-26 **Semester: VII**
Class / Branch/ Div: BE- IT-C **Subject: AI & DS-II**

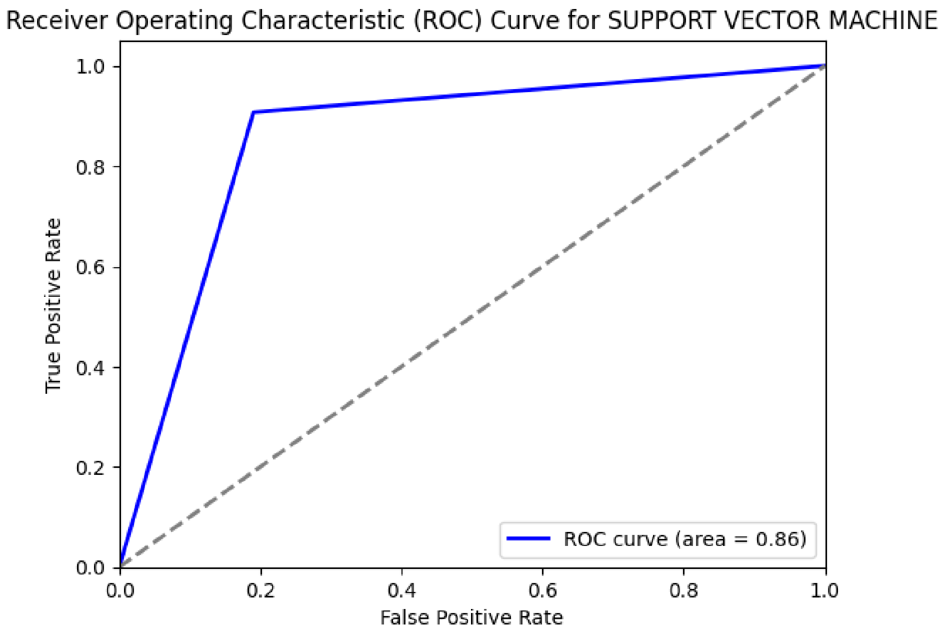
Name: Zahra Surve **Student ID: 23204002**

Accuracy of Support Vector Machine: 0.8713450292397661

Precision of Support Vector Machine: 0.8707035845948703

Recall of Support Vector Machine: 0.8713450292397661

F1-Score of Support Vector Machine: 0.8708976495693214



Evaluation:
As per the plot we can clearly see that SVM outperforms the other two classifiers i.e. Gaussian Naïve Bayes and Decision Tree Classifier with ROC of 0.86 in terms of ROC- AUC curve.