Name: Mariam Badure

Roll no: 04

Subject: ML

Expt.-5: Disease Prediction using Naive Bayes and Neural Network with Comparison of Classifiers

Date: 11\02\2026

In [1]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

In [2]:
```
data = load_breast_cancer()

x = data.data
y = data.target

print("Classes:", data.target_names)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

```
Classes: ['malignant' 'benign']
```

In [3]:
```
# Logistic Regression

lr = LogisticRegression(max_iter=5000)
lr.fit(x_train, y_train)
pred_lr = lr.predict(x_test)
print("Logistic Regression Test Accuracy:", accuracy_score(y_test, pred_lr))
pred_lr_train = lr.predict(x_train)
print("Logistic Train Accuracy:", accuracy_score(y_train, pred_lr_train))
```

```
Logistic Regression Test Accuracy: 0.9473684210526315
Logistic Train Accuracy: 0.9673366834170855
```

In [4]:
```
# Decision Tree

dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
pred_dt = dt.predict(x_test)
```

```
print("Decision Tree Test Accuracy:",accuracy_score(y_test, pred_dt))
pred_dt_train = dt.predict(x_train)
print("Decision Train Accuracy:", accuracy_score(y_train, pred_dt_train))
```

```
Decision Tree Test Accuracy: 0.9473684210526315
Decision Train Accuracy: 1.0
```

In [5]:
```
# KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(x_train, y_train)
pred_knn = knn.predict(x_test)
print("KNeighbors Classifier Test Accuracy:", accuracy_score(y_test, pred_knn))
pred_knn_train = knn.predict(x_train)
print("KNeighbors Classifier Train Accuracy:", accuracy_score(y_train, pred_knn_tra
```

```
KNeighbors Classifier Test Accuracy: 0.9239766081871345
KNeighbors Classifier Train Accuracy: 0.949748743718593
```

In [6]:
```
# GaussianNB

nb =  GaussianNB()
nb.fit(x_train, y_train)
pred_nb = nb.predict(x_test)
print("GaussianNB TestAccuracy:", accuracy_score(y_test, pred_nb))
pred_nb_train = nb.predict(x_train)
print("GaussianNB TrainAccuracy:", accuracy_score(y_train, pred_nb_train))
```

```
GaussianNB TestAccuracy: 0.9473684210526315
GaussianNB TrainAccuracy: 0.9447236180904522
```

In [7]:
```
# MLPClassifier

nn =  MLPClassifier()
nn.fit(x_train, y_train)
pred_nn = nn.predict(x_test)
print("MLP Classifier Test Accuracy:", accuracy_score(y_test, pred_nn))
pred_nn_train = nn.predict(x_train)
print("MLP Classifier Train Accuracy:", accuracy_score(y_train, pred_nn_train))
```

```
MLP Classifier Test Accuracy: 0.9122807017543859
MLP Classifier Train Accuracy: 0.9522613065326633
c:\users\hp\ai-pkj\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.p
y:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached an
d the optimization hasn't converged yet.
  warnings.warn(
```

Expt-5b

Logistic Regression

In [8]:
```
precision = precision_score(y_test, pred_lr)
recall = recall_score(y_test, pred_lr)
```

```
f1 = f1_score(y_test, pred_lr)
roc = roc_auc_score(y_test, pred_lr)
```

In [9]:
```
print("Precision:", precision)
print("Recall:", recall)
print("F-1:", f1)
print("Roc:", roc)
```

```
Precision: 0.9642857142857143
Recall: 0.9557522123893806
F-1: 0.96
Roc: 0.9433933475740006
```

In [10]:
```
# Example confusion matrix, Type-I and Type-II error
cm_lr = confusion_matrix(y_test, pred_lr)
print("\nLogisticRegression Confusion Matrix:")
print(cm_lr)
TN1, FP1, FN1, TP1 = cm_lr.ravel()
type1_lr = FP1 / (FP1 + TN1)
type2_lr = FN1 / (FN1 + TP1)

print("FP (Type-I):", FP1)
print("FP (Type-II):", FN1)
print("Type-I Error Rate in %:", type1_lr*100)
print("Type-II Error Rate in %:", type2_lr*100)
```

```
LogisticRegression Confusion Matrix:
[[ 54   4]
 [  5 108]]
FP (Type-I): 4
FP (Type-II): 5
Type-I Error Rate in %: 6.896551724137931
Type-II Error Rate in %: 4.424778761061947
```

Decision Tree

In [11]:
```
precision = precision_score(y_test, pred_dt)
recall = recall_score(y_test, pred_dt)
f1 = f1_score(y_test, pred_dt)
roc = roc_auc_score(y_test, pred_dt)
```

In [12]:
```
print("Precision:", precision)
print("Recall:", recall)
print("F-1:", f1)
print("Roc:", roc)
```

```
Precision: 0.9482758620689655
Recall: 0.9734513274336283
F-1: 0.9606986899563319
Roc: 0.9350015257857797
```

In [13]:
```
 # Example confusion matrix, Type-I and Type-II error
cm_dt = confusion_matrix(y_test, pred_dt)
print("\nDecision Tree Confusion Matrix:")
print(cm_dt)
```

```
TN2, FP2, FN2, TP2 = cm_dt.ravel()
type1_dt = FP2 / (FP2 + TN2)
type2_dt = FN2 / (FN2 + TP2)

print("FP (Type-I):", FP2)
print("FP (Type-II):", FN2)
print("Type-I Error Rate in %:", type1_dt*100)
print("Type-II Error Rate in %:", type2_dt*100)
```

```
Decision Tree Confusion Matrix:
[[ 52   6]
 [  3 110]]
FP (Type-I): 6
FP (Type-II): 3
Type-I Error Rate in %: 10.344827586206897
Type-II Error Rate in %: 2.6548672566371683
```

KNN

In [14]:
```
precision = precision_score(y_test, pred_knn)
recall = recall_score(y_test, pred_knn)
f1 = f1_score(y_test, pred_knn)
roc = roc_auc_score(y_test, pred_knn)
```

In [15]:
```
print("Precision:", precision)
print("Recall:", recall)
print("F-1:", f1)
print("Roc:", roc)
```

```
Precision: 0.9098360655737705
Recall: 0.9823008849557522
F-1: 0.9446808510638298
Roc: 0.8963228562709795
```

In [16]:
```
#KNeighborsClassifier

cm_knn = confusion_matrix(y_test, pred_knn)
print("\nKNN Confusion Matrix:")
print(cm_knn)

TN3, FP3, FN3, TP3 = cm_knn.ravel()

type1_knn = FP3 / (FP3 + TN3)
type2_knn = FN3 / (FN3 + TP3)

print("FP (Type-I):", FP3)
print("FN (Type-II):", FN3)
print("Type-I Error Rate in %:", type1_knn * 100)
print("Type-II Error Rate in %:", type2_knn * 100)
```

```
KNN Confusion Matrix:
[[ 47  11]
 [  2 111]]
FP (Type-I): 11
FN (Type-II): 2
Type-I Error Rate in %: 18.96551724137931
Type-II Error Rate in %: 1.7699115044247788
```

GaussianNB

In [17]:
```
precision = precision_score(y_test, pred_nb)
recall = recall_score(y_test, pred_nb)
f1 = f1_score(y_test, pred_nb)
roc = roc_auc_score(y_test, pred_nb)
```

In [18]:
```
print("Precision:", precision)
print("Recall:", recall)
print("F-1:", f1)
print("Roc:", roc)
```

```
Precision: 0.940677966101695
Recall: 0.9823008849557522
F-1: 0.961038961038961
Roc: 0.9308056148916692
```

In [19]:
```
# GaussianNB
cm_nb = confusion_matrix(y_test, pred_nb)
print("\nNaive Bayes Confusion Matrix:")
print(cm_nb)

TN4, FP4, FN4, TP4 = cm_nb.ravel()

type1_nb = FP4 / (FP4 + TN4)
type2_nb = FN4 / (FN4 + TP4)

print("FP (Type-I):", FP4)
print("FN (Type-II):", FN4)
print("Type-I Error Rate in %:", type1_nb * 100)
print("Type-II Error Rate in %:", type2_nb * 100)
```

```
Naive Bayes Confusion Matrix:
[[ 51   7]
 [  2 111]]
FP (Type-I): 7
FN (Type-II): 2
Type-I Error Rate in %: 12.068965517241379
Type-II Error Rate in %: 1.7699115044247788
```

MLPClassifier

In [20]:
```
precision = precision_score(y_test, pred_nn)
recall = recall_score(y_test, pred_nn)
```

```
f1 = f1_score(y_test, pred_nn)
roc = roc_auc_score(y_test, pred_nn)
```

In [21]:
```
print("Precision:", precision)
print("Recall:", recall)
print("F-1:", f1)
print("Roc:", roc)
```

```
Precision: 0.8888888888888888
Recall: 0.9911504424778761
F-1: 0.9372384937238494
Roc: 0.8748855660665242
```

In [22]:
```
# MLPClassifier

cm_nn = confusion_matrix(y_test, pred_nn)
print("\nNeural Network Confusion Matrix:")
print(cm_nn)

TN5, FP5, FN5, TP5 = cm_nn.ravel()

type1_nn = FP5 / (FP5 + TN5)
type2_nn = FN5 / (FN5 + TP5)

print("FP (Type-I):", FP5)
print("FN (Type-II):", FN5)
print("Type-I Error Rate in %:", type1_nn * 100)
print("Type-II Error Rate in %:", type2_nn * 100)
```

```
Neural Network Confusion Matrix:
[[ 44  14]
 [  1 112]]
FP (Type-I): 14
FN (Type-II): 1
Type-I Error Rate in %: 24.137931034482758
Type-II Error Rate in %: 0.8849557522123894
```

In [ ]:
```

```