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

## Student Name: Prewitt Gomes

Roll No.: 22 Batch: 1 Date: 11-02-2026

```
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 mean_squared_error,r2_score,precision_score,accurac
import pandas as pd
```

```
data=load_breast_cancer()

x=data.data
y=data.target
print("Classes:", data.target_names)

df=pd.DataFrame(x, columns=data.feature_names)
df['target']=y

df.head()
df.tail()
df.info()
df.columns
df.describe()
```

```
Classes: ['malignant' 'benign']
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
```

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

```
lr = LogisticRegression(max_iter=5000)
lr.fit(x_train, y_train)
pred_lr = lr.predict(x_test)
print("Logistic TestAccuracy:", accuracy_score(y_test, pred_lr))
pred_lr_train = lr.predict(x_train)
print("Logistic TrainAccuracy:", accuracy_score(y_train, pred_lr_train))
```

```
Logistic TestAccuracy: 0.9590643274853801
Logistic TrainAccuracy: 0.9597989949748744
```

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)
pred_dt = dt.predict(x_test)
print("DecisionTreeClassifier TestAccuracy:", accuracy_score(y_test, pred_dt))
pred_dt_train = dt.predict(x_train)
print("DecisionTreeClassifier TrainAccuracy:", accuracy_score(y_train, pred_dt_
```

```
DecisionTreeClassifier TestAccuracy: 0.8888888888888888
DecisionTreeClassifier TrainAccuracy: 1.0
```

```
kn = KNeighborsClassifier()
kn.fit(x_train, y_train)
pred_kn = kn.predict(x_test)
print("KNeighborsClassifier TestAccuracy:", accuracy_score(y_test, pred_kn))
pred_kn_train = kn.predict(x_train)
print("KNeighborsClassifier TrainAccuracy:", accuracy_score(y_train, pred_kn_tr
```

```
KNeighborsClassifier TestAccuracy: 0.9239766081871345
KNeighborsClassifier TrainAccuracy: 0.949748743718593
```

```
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: 0.9415204678362573
GaussianNB TrainAccuracy: 0.9371859296482412
```

| | mean | mean | mean | mean area | mean | mean |
|---|---|---|---|---|---|---|
| **min** | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 |
| **25%** | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 |

```
mlp = MLPClassifier()
mlp.fit(x_train, y_train)
```

```
pred_mlp = mlp.predict(x_test)
print("MLPClassifier TestAccuracy:", accuracy_score(y_test, pred_mlp))
pred_mlp_train = mlp.predict(x_train)
print("MLPClassifier TrainAccuracy:", accuracy_score(y_train, pred_mlp_train))
```

```
MLPClassifier TestAccuracy: 0.935672514619883
MLPClassifier TrainAccuracy: 0.9547738693467337
/usr/local/lib/python3.12/dist-packages/sklearn/neural_network/_multilayer_perce
  warnings.warn(
```

```
cm_lr = confusion_matrix(y_test, pred_lr)
print("\nLogistic Regression 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("FN (Type-II):", FN1)
print("Type-I Error Rate in %:", type1_lr*100)
print("Type-II Error Rate in %:", type2_lr*100)
```

```
Logistic Regression Confusion Matrix:
[[ 57   4]
 [  3 107]]
FP (Type-I): 4
FN (Type-II): 3
Type-I Error Rate in %: 6.557377049180328
Type-II Error Rate in %: 2.727272727272727
```

```
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("FN (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:
[[53  8]
 [11 99]]
FP (Type-I): 8
FN (Type-II): 11
Type-I Error Rate in %: 13.114754098360656
Type-II Error Rate in %: 10.0
```

```
cm_kn = confusion_matrix(y_test, pred_kn)
print("\n KNeigbors Confusion Matrix:")
print(cm_kn)
TN3, FP3, FN3, TP3 = cm_kn.ravel()
type1_kn = FP3 / (FP3 + TN3)
type2_kn = FN3 / (FN3 + TP3)

print("FP (Type-I):", FP3)
print("FN (Type-II):", FN3)
print("Type-I Error Rate in %:", type1_kn*100)
print("Type-II Error Rate in %:", type2_kn*100)
```

```
 KNeigbors Confusion Matrix:
[[ 51  10]
 [  3 107]]
FP (Type-I): 10
FN (Type-II): 3
Type-I Error Rate in %: 16.39344262295082
Type-II Error Rate in %: 2.727272727272727
```

```
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:
[[ 55   6]
 [  4 106]]
FP (Type-I): 6
FN (Type-II): 4
Type-I Error Rate in %: 9.836065573770492
Type-II Error Rate in %: 3.6363636363636362
```

```
cm_mlp = confusion_matrix(y_test, pred_mlp)
print("\nMLP Confusion Matrix:")
print(cm_mlp)
TN5, FP5, FN5, TP5 = cm_mlp.ravel()
type1_mlp = FP5 / (FP5 + TN5)
type2_mlp = FN5 / (FN5 + TP5)

print("FP (Type-I):", FP5)
print("FN (Type-II):", FN5)
```

```
print("Type-I Error Rate in %:", type1_mlp*100)
print("Type-II Error Rate in %:", type2_mlp*100)
```

```
MLP Confusion Matrix:
[[ 52   9]
 [  2 108]]
FP (Type-I): 9
FN (Type-II): 2
Type-I Error Rate in %: 14.754098360655737
Type-II Error Rate in %: 1.8181818181818181
```

```
train_acc_lr = accuracy_score(y_train, lr.predict(x_train))
test_acc_lr = accuracy_score(y_test, pred_lr)
precision_lr = precision_score(y_test, pred_lr)
recall_lr = recall_score(y_test, pred_lr)
f1_lr = f1_score(y_test, pred_lr)
roc_lr = roc_auc_score(y_test, pred_lr)

print("\n===== LOGISTIC REGRESSION RESULTS =====")
print("Training Accuracy :", train_acc_lr)
print("Test Accuracy     :", test_acc_lr)
print("Precision         :", precision_lr)
print("Recall            :", recall_lr)
print("F1-score          :", f1_lr)
print("ROC-AUC           :", roc_lr)
```

```
===== LOGISTIC REGRESSION RESULTS =====
Training Accuracy : 0.9597989949748744
Test Accuracy     : 0.9590643274853801
Precision         : 0.963963963963964
Recall            : 0.9727272727272728
F1-score          : 0.9683257918552036
ROC-AUC           : 0.9535767511177348
```

```
train_acc_dt = accuracy_score(y_train, dt.predict(x_train))
test_acc_dt = accuracy_score(y_test, pred_dt)
precision_dt = precision_score(y_test, pred_dt)
recall_dt = recall_score(y_test, pred_dt)
f1_dt = f1_score(y_test, pred_dt)
roc_dt = roc_auc_score(y_test, pred_dt)

print("\n===== DECISION TREE RESULTS =====")
print("Training Accuracy :", train_acc_dt)
print("Test Accuracy     :", test_acc_dt)
print("Precision         :", precision_dt)
print("Recall            :", recall_dt)
print("F1-score          :", f1_dt)
print("ROC-AUC           :", roc_dt)
```

```
===== DECISION TREE RESULTS =====
Training Accuracy : 1.0
Test Accuracy     : 0.8888888888888888
Precision         : 0.9252336448598131
Recall            : 0.9
F1-score          : 0.9124423963133641
ROC-AUC           : 0.8844262295081967
```

```
pred_kn = kn.predict(x_test)
train_acc_kn = accuracy_score(y_train, kn.predict(x_train))
test_acc_kn = accuracy_score(y_test, pred_kn)
precision_kn = precision_score(y_test, pred_kn)
recall_kn = recall_score(y_test, pred_kn)
f1_kn = f1_score(y_test, pred_kn)
roc_kn = roc_auc_score(y_test, pred_kn)

print("\n===== KNN RESULTS =====")
print("Training Accuracy :", train_acc_kn)
print("Test Accuracy     :", test_acc_kn)
print("Precision         :", precision_kn)
print("Recall            :", recall_kn)
print("F1-score          :", f1_kn)
print("ROC-AUC           :", roc_kn)
```

```
===== KNN RESULTS =====
Training Accuracy : 0.949748743718593
Test Accuracy     : 0.9239766081871345
Precision         : 0.9145299145299145
Recall            : 0.9727272727272728
F1-score          : 0.9427312775330396
ROC-AUC           : 0.9043964232488823
```

```
pred_nb = nb.predict(x_test)
train_acc_nb = accuracy_score(y_train, nb.predict(x_train))
test_acc_nb = accuracy_score(y_test, pred_nb)
precision_nb = precision_score(y_test, pred_nb)
recall_nb = recall_score(y_test, pred_nb)
f1_nb = f1_score(y_test, pred_nb)
roc_nb = roc_auc_score(y_test, pred_nb)

print("\n===== NAIVE BAYES RESULTS =====")
print("Training Accuracy :", train_acc_nb)
print("Test Accuracy     :", test_acc_nb)
print("Precision         :", precision_nb)
print("Recall            :", recall_nb)
print("F1-score          :", f1_nb)
print("ROC-AUC           :", roc_nb)
```

```
===== NAIVE BAYES RESULTS =====
Training Accuracy : 0.9371859296482412
Test Accuracy     : 0.9415204678362573
Precision         : 0.9464285714285714
Recall            : 0.9636363636363636
F1-score          : 0.954954954954955
```

```
pred_mlp = mlp.predict(x_test)
train_acc_mlp = accuracy_score(y_train, mlp.predict(x_train))
test_acc_mlp = accuracy_score(y_test, pred_mlp)
precision_mlp = precision_score(y_test, pred_mlp)
recall_mlp = recall_score(y_test, pred_mlp)
f1_mlp = f1_score(y_test, pred_mlp)
roc_mlp = roc_auc_score(y_test, pred_mlp)

print("\n===== MLP RESULTS =====")
print("Training Accuracy :", train_acc_mlp)
print("Test Accuracy     :", test_acc_mlp)
print("Precision         :", precision_mlp)
print("Recall            :", recall_mlp)
print("F1-score          :", f1_mlp)
print("ROC-AUC           :", roc_mlp)
```

```
===== MLP RESULTS =====
Training Accuracy : 0.9547738693467337
Test Accuracy     : 0.935672514619883
Precision         : 0.9230769230769231
Recall            : 0.9818181818181818
F1-score          : 0.9515418502202643
ROC-AUC           : 0.9171385991058123
```