# 3.NAIVE BAYES CLASSIFIER

- JUNE2024
- PRACTICAL
- PROJECT1 :NAIVE BAYES 1

## Naive bayes classifeir

- It is probalistic supervised machinelearning algorithm to predict class
- mostly used for text data ,multiclass classifications
- Types of NB:
    - Multinomial N.B :DATA having classes
    - Bernoulli N.B: class in 0&1 boolean format
    - Gaussian N.B : Numerical dataset : datashould be normallly distributed

# PROJECT 1:

- Breast cancer prediction :
- The goal of this project is to develop a predictive model using the Naive Bayes algorithm to classify whether a tumor is benign or malignant based on certain features.
- Breast cancer is one of the most common cancers among women worldwide. Early detection is crucial for effective treatment and improved survival rates. This project aims to create a machine learning model using the Naive Bayes algorithm to predict whether a

breast tumor is benign (non-cancerous) or malignant (cancerous) based on features obtained from medical imaging.

```
# import libraries
import numpy as np
import pandas as pd
```

```
#load dataset
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
```

```
data.data
```

Show hidden output

```
data.feature_names
```

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean
area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal
dimension',
       'radius error', 'texture error', 'perimeter error',
'area error',
       'smoothness error', 'compactness error', 'concavity
error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst
texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave
points',
       'worst symmetry', 'worst fractal dimension'],
dtype='<U23')
```

```
data.target
```

```
data.target_names #malignant: cancer=0, benign= nocancer=1
```

```
array(['malignant', 'benign'], dtype='<U9')
```

```
#Create dataframaame using above data
df=pd.DataFrame(np.c_[data.data, data.target], columns=[list(data.fe
df.head()
```

| | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | m c |
|---|---|---|---|---|---|---|---|
| **0** | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | |
| **1** | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | |
| **2** | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | |
| **3** | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | |
| **4** | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | |

5 rows × 31 columns

```
df.shape
```

```
(569, 31)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   (mean radius,)              569 non-null    float64
 1   (mean texture,)             569 non-null    float64
 2   (mean perimeter,)           569 non-null    float64
 3   (mean area,)                569 non-null    float64
 4   (mean smoothness,)          569 non-null    float64
 5   (mean compactness,)         569 non-null    float64
 6   (mean concavity,)           569 non-null    float64
 7   (mean concave points,)      569 non-null    float64
```

```
 8   (mean symmetry,)            569 non-null    float64
 9   (mean fractal dimension,)   569 non-null    float64
 10  (radius error,)             569 non-null    float64
 11  (texture error,)            569 non-null    float64
 12  (perimeter error,)          569 non-null    float64
 13  (area error,)               569 non-null    float64
 14  (smoothness error,)         569 non-null    float64
 15  (compactness error,)        569 non-null    float64
 16  (concavity error,)          569 non-null    float64
 17  (concave points error,)     569 non-null    float64
 18  (symmetry error,)           569 non-null    float64
 19  (fractal dimension error,)  569 non-null    float64
 20  (worst radius,)             569 non-null    float64
 21  (worst texture,)            569 non-null    float64
 22  (worst perimeter,)          569 non-null    float64
 23  (worst area,)               569 non-null    float64
 24  (worst smoothness,)         569 non-null    float64
 25  (worst compactness,)        569 non-null    float64
 26  (worst concavity,)          569 non-null    float64
 27  (worst concave points,)     569 non-null    float64
 28  (worst symmetry,)           569 non-null    float64
 29  (worst fractal dimension,)  569 non-null    float64
 30  (target,)                   569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

### Seperate indepdent anddepdent features:

- X&y

```
#indepdeent and depdent features are seperate out :
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

### Train_test_split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size

print('Shape of X_train = ', X_train.shape)
print('Shape of y_train = ', y_train.shape)
print('Shape of X_test = ', X_test.shape)
```

```
print('Shape of y_test = ', y_test.shape)
```

```
Shape of X_train =  (455, 30)
Shape of y_train =  (455,)
Shape of X_test =  (114, 30)
Shape of y_test =  (114,)
```

# Train Naive Bayes Classifier Model

```
#I.Use GAUSSIAN N.B:Numerical features
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(X_train, y_train)#train model
```

```
GaussianNB()
```

```
#score
classifier.score(X_test, y_test)
```

```
0.9736842105263158
```

```
#II.MultinomialNB :mostly usedfor categorical data
from sklearn.naive_bayes import MultinomialNB
classifier_m = MultinomialNB()
classifier_m.fit(X_train, y_train)
```

```
MultinomialNB()
```

```
classifier_m.score(X_test,y_test)
```

```
0.8947368421052632
```

```
#III.BernoulliNB:binary class 0&1
from sklearn.naive_bayes import BernoulliNB
classifier_b = BernoulliNB()
classifier_b.fit(X_train, y_train)

classifier_b.score(X_test, y_test)
```

```
0.5789473684210527
```

```
#BernoulliNB givesbadscore soit ismostly used on textdata
```

```
#I.Use GAUSSIAN N.B:givesgoodscore fornumericalfeaturechoosefor predic
#Preditct feature from unseen/ new data
```

```
patient1 = [17.99,
 10.38,
 122.8,
 1001.0,
 0.1184,
 0.2776,
 0.3001,
 0.1471,
 0.2419,
 0.07871,
 1.095,
 0.9053,
 8.589,
 153.4,
 0.006399,
 0.04904,
 0.05373,
 0.01587,
 0.03003,
 0.006193,
 25.38,
 17.33,
 184.6,
 2019.0,
 0.1622,
 0.6656,
 0.7119,
```

```
 0.2654,
 0.4601,
 0.1189]
```

```
patient1 = np.array([patient1])
patient1
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01,
    2.776e-01,
            3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00,
    9.053e-01,
            8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02,
    1.587e-02,
            3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02,
    2.019e+03,
            1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01,
    1.189e-01]])
```

```
#PREDICT FEATURE:
classifier.predict(patient1)#cancer
```

```
array([0.])
```

```
pred = classifier.predict(patient1)
```

```
if pred[0] == 0:
  print('Patient has Cancer (malignant tumor)')
else:
  print('Patient has no Cancer (malignant benign)')
```

```
Patient has Cancer (malignant tumor)
```

## Evaluation metrics:

```
from sklearn.metrics import accuracy_score, precision_score, recall_sc

# Predictions on test set useony Gaussiann.b model
y_pred = classifier.predict(X_test)
```

```
# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.97
```

```
# Precision, Recall, F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-score: {f1:.2f}")
```

```
Precision: 0.96
Recall: 1.00
F1-score: 0.98
```

```
# Classification Report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.94      0.97        48
         1.0       0.96      1.00      0.98        66

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

Confusion Matrix:
[[45  3]
 [ 0 66]]
```

## CONCLUSIONS

- The model tells that robust performance across both classes, with high precision and recall scores indicating reliable predictions.
- The confusion matrix confirms minimal misclassifications, with only a few instances of false positives.
- Both accuracy and balanced dataset it goodfor classification taskfor thisbreast cancer project.

**Save Model:**

```
import joblib
```

```
joblib.dump(classifier, 'naive_bayes_Breastcancermodel.pkl')
```

```
['naive_bayes_Breastcancermodel.pkl']
```

```
#download.pkl filerun it on localcomputer for deployement using Flas
```