Introduction- Breast cancer data is taken from Kaggle, Breast cancer Wisconsin (https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data). Here we are building and training the model to predict breast cancer in any prospective individual and to know whether cancer is malignant (M) or benign (B).

About Dataset

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. n the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server: ftp ftp.cs.wisc.edu cd math-prog/cpo-dataset/machine-learn/WDBC/

Also can be found on UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Attribute Information:

1) ID number

2) Diagnosis (M = malignant, B = benign) 3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area - 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.

Missing attribute values: none

Class distribution: 357 benign, 212 malignant

1) Data loading and creating DataFrame

```
#importing data analysis and visuvalization library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
#loading raw data and creating DataFrame
path ='/content/Breast cancer data.csv'
df=pd.read_csv(path)
df.head() # to read first 5 rows of datasets
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness
0	842302	M	17.99	10.38	122.80	1001.0	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.
2	84300903	M	19.69	21.25	130.00	1203.0	0.
3	84348301	M	11.42	20.38	77.58	386.1	0.
4	84358402	M	20.29	14.34	135.10	1297.0	0.

5 rows × 33 columns

```
df.shape # to know the number of rows and columns
```

```
(569, 33)
```

```
df.info() # to know different data types and any missing values, there is no missing values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
```

```
  1   diagnosis                569 non-null    object 
  2   radius_mean              569 non-null    float64
  3   texture_mean             569 non-null    float64
  4   perimeter_mean           569 non-null    float64
  5   area_mean                569 non-null    float64
  6   smoothness_mean          569 non-null    float64
  7   compactness_mean         569 non-null    float64
  8   concavity_mean           569 non-null    float64
  9   concave points_mean      569 non-null    float64
  10  symmetry_mean            569 non-null    float64
  11  fractal_dimension_mean   569 non-null    float64
  12  radius_se                569 non-null    float64
  13  texture_se               569 non-null    float64
  14  perimeter_se             569 non-null    float64
  15  area_se                  569 non-null    float64
  16  smoothness_se            569 non-null    float64
  17  compactness_se           569 non-null    float64
  18  concavity_se             569 non-null    float64
  19  concave points_se        569 non-null    float64
  20  symmetry_se              569 non-null    float64
  21  fractal_dimension_se     569 non-null    float64
  22  radius_worst             569 non-null    float64
  23  texture_worst            569 non-null    float64
  24  perimeter_worst          569 non-null    float64
  25  area_worst               569 non-null    float64
  26  smoothness_worst         569 non-null    float64
  27  compactness_worst        569 non-null    float64
  28  concavity_worst          569 non-null    float64
  29  concave points_worst     569 non-null    float64
  30  symmetry_worst           569 non-null    float64
  31  fractal_dimension_worst  569 non-null    float64
  32  Unnamed: 32              0 non-null      float64
 dtypes: float64(31), int64(1), object(1)
 memory usage: 146.8+ KB
```

2) Divide data into input and output data

```
x= df.iloc[:,2:32] # input data # last column is blank so its not considered while building
y= df.iloc[:,1] # output data
```

3) Splitting data into Train and Test variables- Here we split data into train and test data. Since we have 569 data entries so we can train model with most of the data and keep some data to test the model and predict the

outcome. Later based on the predicted output we can check accuracy score.

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test= train_test_split(x,y,random_state=42,test_size=0.33) # here t
# by default 70% data goes into train data and 30% data goes into test data.
```

```
# to check the number of data in train and test data types
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(381, 30)
(188, 30)
(381,)
(188,)
```

```
# to check 5 data entries in x_train, this will assign data randomly and the last column whi
x_train.head()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_
172	15.46	11.89	102.50	736.9	0.12570	0.1
407	12.85	21.37	82.63	514.5	0.07551	0.0
56	19.21	18.57	125.50	1152.0	0.10530	0.1
497	12.47	17.31	80.45	480.1	0.08928	0.0
301	12.46	19.89	80.43	471.3	0.08451	0.1

5 rows × 30 columns

4) Model building based on regression, classifier or clustring and training model with train data sets.

```
# model building based on supervised learning (regression and classifier) and unsupervised l
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=19,metric= 'euclidean') # n_neighbors is k value wh
```

```
# training model with train data set
model.fit(x_train,y_train)
```

```
▾                 KNeighborsClassifier
KNeighborsClassifier(metric='euclidean', n_neighbors=19)
```

5) Predicting the output from trained model

```
y_pred= model.predict(x_test) # predicting the output from trained model
y_pred
```

```
array(['B', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'B',
       'M', 'B', 'M', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
       'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'M',
       'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'M', 'B', 'B',
       'B', 'M', 'M', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B',
       'B', 'B', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'M', 'B', 'M', 'M',
       'B', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B',
       'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'M',
       'M', 'M', 'B', 'B', 'M', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'B',
       'B', 'M', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'M',
       'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'M', 'B', 'B'], dtype=object)
```

```
y_test.values
```

```
array(['B', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'B',
       'M', 'B', 'M', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'M',
       'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'M',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'B', 'M', 'M', 'B', 'B',
       'B', 'M', 'M', 'B', 'B', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'M',
       'B', 'B', 'M', 'B', 'M', 'M', 'M', 'M', 'M', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'B', 'M', 'M', 'B', 'M', 'M',
       'B', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B',
       'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'M',
       'M', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'B',
       'B', 'M', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'M',
       'M', 'B', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B',
       'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'M', 'B', 'B'], dtype=object)
```

```
# predicting the output of anyone patient
patient_a= x_test.values[121]
patient_a
```

```
array([1.143e+01, 1.539e+01, 7.306e+01, 3.998e+02, 9.639e-02, 6.889e-02,
       3.503e-02, 2.875e-02, 1.734e-01, 5.865e-02, 1.759e-01, 9.938e-01,
       1.143e+00, 1.267e+01, 5.133e-03, 1.521e-02, 1.434e-02, 8.602e-03,
       1.501e-02, 1.588e-03, 1.232e+01, 2.202e+01, 7.993e+01, 4.620e+02,
       1.190e-01, 1.648e-01, 1.399e-01, 8.476e-02, 2.676e-01, 6.765e-02])
```

```
# predicting the cancer type from trained model
model.predict([patient_a])
# patient_a has benign (B) type of cancer
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does not hav
  warnings.warn(
array(['B'], dtype=object)
```

```
# predicting the output of anyone patient
patient_b= x_test.values[100]
patient_b
```

```
array([1.570e+01, 2.031e+01, 1.012e+02, 7.666e+02, 9.597e-02, 8.799e-02,
       6.593e-02, 5.189e-02, 1.618e-01, 5.549e-02, 3.699e-01, 1.150e+00,
       2.406e+00, 4.098e+01, 4.626e-03, 2.263e-02, 1.954e-02, 9.767e-03,
       1.547e-02, 2.430e-03, 2.011e+01, 3.282e+01, 1.293e+02, 1.269e+03,
       1.414e-01, 3.547e-01, 2.902e-01, 1.541e-01, 3.437e-01, 8.631e-02])
```

```
# predicting the cancer type from trained model
model.predict([patient_b])
# patient_a has malignant (M) type of cancer
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:465: UserWarning: X does not hav
  warnings.warn(
array(['M'], dtype=object)
```

6) Check accuracy of the data predicted by the model

```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)
# the accuracy score is 96.8%, its a good score! the more data we train our model the better
```

```
0.9680851063829787
```

```
a=[1,2,3,'seven',4,5,9,8]
a[-3:-1]
```

```
[5, 9]
```