Breast Cancer Wisconsin (Diagnostic) Data Set

Breast Cancer Classification

Data Set Characteristics:	Multivariate
Attribute Characteristics:	Real
Associated Tasks:	Classification

Number of Instances:	569
Number of Attributes:	32
Missing Values?	No

some values were deleted to handel missing data

Logistic Regression & K Nearest Neighbors

Process

1- Preprocessing

2- Data Mining

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	co
0	842302	M	17.990	10.38	122.80	1001.0	NAN	
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	
2	84300903	M	19.690	21.25	130.00	1203.0	0.1096	
3	84348301	M	11.420	20.38	77.58	386.1	0.1425	
4	84358402	M	20.290	14.34	135.10	1297.0	0.1003	
5	843786	M	12.450	15.70	82.57	477.1	0.1278	
6	844359	M	18.250	19.98	119.60	1040.0	0.09463	
7	84458202	M	13.710	20.83	90.20	577.9	0.1189	
8	844981	M	13.000	21.82	87.50	519.8	0.1273	
9	84501001	M	12.460	24.04	83.97	475.9	0.1186	

The Steps Of Preprocessing

1. Importing the dataset
2. Taking care of missing data
3. Encoding the Class Variable
4. Splitting the dataset into the Training set and Test set
5. Feature Scaling

01

Importing The Data

importing the data set from **UCI** as **CSV** file from **.data** file enabling us to manulate the data and handel the missing values
then assign names to the columns

```
import pandas as pd
import numpy as np

breastDataFrame = pd.read_csv("breastCancer.data" ,sep="," , header=None)

breastDataFrame.columns = ["id","diagnosis","radius_mean","texture_mean",
"perimeter_mean","area_mean","smoothness_mean","compactness_mean","
concavity_mean","concave points_mean","symmetry_mean","
fractal_dimension_mean","radius_se","texture_se","perimeter_se","area_se"
,"smoothness_se","compactness_se","concavity_se","concave points_se","
symmetry_se","fractal_dimension_se","radius_worst","texture_worst","
perimeter_worst","area_worst","smoothness_worst","compactness_worst","
concavity_worst","concave points_worst","symmetry_worst","
fractal_dimension_worst"]
```

Selection data X are the independent variables and y will be the dependent variable

```
X = breastDataFrame.iloc[:,1:]
y = breastDataFrame.iloc[:,[0]]
```

01

Output

Drop ID Column

Class

+ 10 Features →

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean	radius_se	texture
0	842302	M	17.990	10.38	122.80	1001.0	NAN	0.27760	0.300100	0.147100	0.2419	0.07871	1.095	0.9
1	842517	M	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.1812	0.05667	0.5435	0.7
2	84300903	M	19.690	21.25	130.00	1203.0	0.1096	0.15990	0.197400	0.127900	0.2069	0.05999	0.7456	0.7
3	84348301	M	11.420	20.38	77.58	386.1	0.1425	0.28390	0.241400	0.105200	0.2597	0.09744	0.4956	1.1
4	84358402	M	20.290	14.34	135.10	1297.0	0.1003	0.13280	0.198000	0.104300	0.1809	0.05883	NAN	0.7
5	843786	M	12.450	15.70	82.57	477.1	0.1278	0.17000	0.157800	0.080890	0.2087	0.07613	0.3345	0.8
6	844359	M	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.1794	0.05742	0.4467	0.7
								0.16450	0.093660	0.059850	0.2196	0.07451	0.5835	1.3
								0.19320	0.185900	0.093530	0.2350	0.07389	0.3063	1.0
								0.23960	0.227300	0.085430	0.2030	0.08243	0.2976	1.5

```
breastDataFrame = breastDataFrame.drop(['id'],axis=1)
```

02

```python
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')

imputer = imputer.fit(X.iloc[:, :])

X.iloc[:, :] = imputer.transform(X.iloc[:, :])


X = np.array(X)

```

Handel Missing Data

Each NAN value replaced with the mean of the column

03

Encoding The Class Variable

M => 1

B => 0

```python
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
y = le.fit_transform(y)
```

```
In [53]: y

Out[53]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
                0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
                0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1,
                1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0,
                0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
                1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
                1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
                1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
                0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
```

```python
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size
= 0.25, random_state = 39)
```

04

Splitting The Dataset Into The Training Set And Test Set

75% of the data for training the model

25% of the data for testing the model

random state selection of values is 39

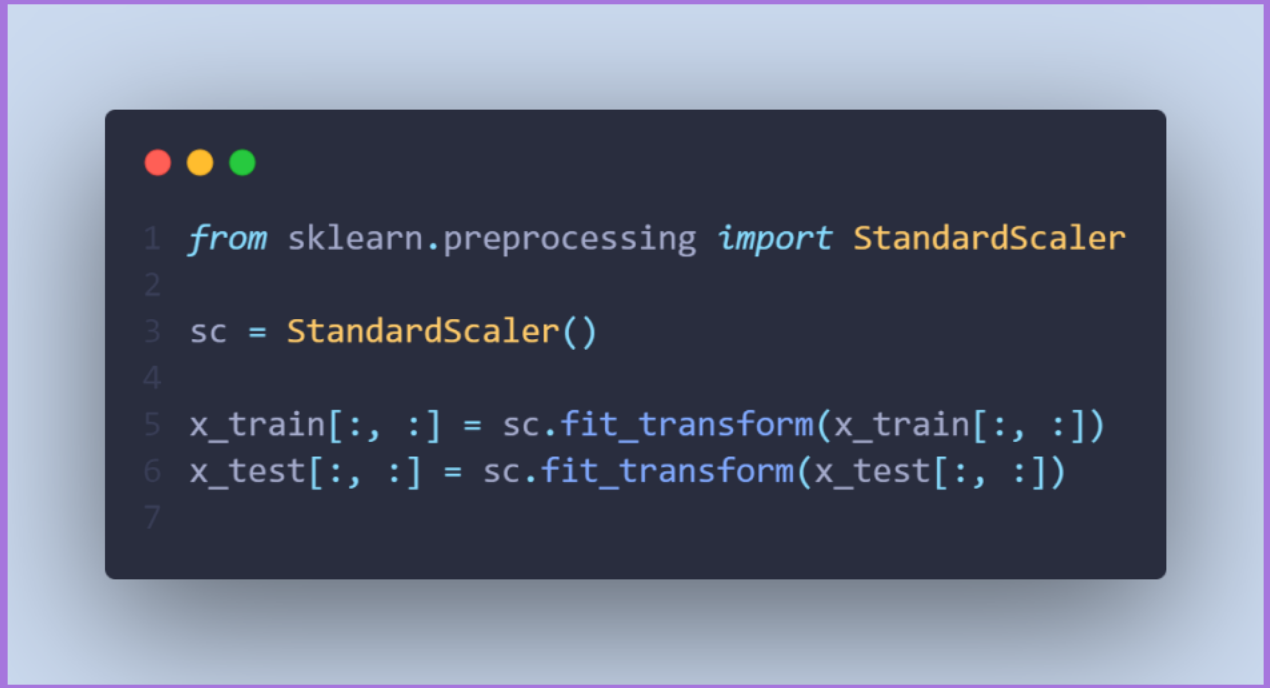
05

Feature Scaling

Standardisation

$$= \frac{X - \text{mean}(X)}{\text{Standard deviation}(X)}$$

after Standardisation all values will be around between -3 and +3



```python
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train[:, :] = sc.fit_transform(x_train[:, :])
x_test[:, :] = sc.fit_transform(x_test[:, :])
```

Done with Preprocessing

Time For Data Mining

1- Preprocessing

2- Data Mining

Two Machine Learining Models Were Used

1- Logisitic Regression (95%)

2- KNN (94%)

```python
from sklearn.linear_model import LogisticRegression
Log_BC_Model = LogisticRegression()

Log_BC_Model.fit(x_train, y_train)
yDash = Log_BC_Model.predict(x_test)
```

01 Logistic Regression

```
In [35]: from sklearn.metrics import accuracy_score
         print("The score of model is : {} ".format(accuracy_score(y_test,yDash)))

         The score of model is : 0.951048951048951
```

02
K Nearest Neighbors

```python
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5,
metric = "minkowski", p = 2)
classifier = classifier.fit(np.array(x_train), y_train)
yDash = classifier.predict(x_test)

```

In [34]:
```python
from sklearn.metrics import accuracy_score

print("The score of model is : {} ".format(accuracy_score(y_test,yDash)))
```

```
The score of model is : 0.9440559440559441
```

Thank You!