Name : Susmita Rani Saha , ID : B-180305047

Name : Tanvir ahammed hridoy , ID : B-180305020

Step 1 : Import dataset and print

first of all we have to import dataset which contain null or missing values

In [37]:
```
# load the dataset and review rows
from pandas import read_csv
# load the dataset
dataset = read_csv('pima-indians-diabetes.csv', header=None)
# print the first 20 rows of data
print(dataset.head(20))
```

```
     0    1   2   3    4     5      6   7  8
0    6  148  72  35    0  33.6  0.627  50  1
1    1   85  66  29    0  26.6  0.351  31  0
2    8  183  64   0    0  23.3  0.672  32  1
3    1   89  66  23   94  28.1  0.167  21  0
4    0  137  40  35  168  43.1  2.288  33  1
5    5  116  74   0    0  25.6  0.201  30  0
6    3   78  50  32   88  31.0  0.248  26  1
7   10  115   0   0    0  35.3  0.134  29  0
8    2  197  70  45  543  30.5  0.158  53  1
9    8  125  96   0    0   0.0  0.232  54  1
10   4  110  92   0    0  37.6  0.191  30  0
11  10  168  74   0    0  38.0  0.537  34  1
12  10  139  80   0    0  27.1  1.441  57  0
13   1  189  60  23  846  30.1  0.398  59  1
14   5  166  72  19  175  25.8  0.587  51  1
15   7  100   0   0    0  30.0  0.484  32  1
16   0  118  84  47  230  45.8  0.551  31  1
17   7  107  74   0    0  29.6  0.254  31  1
18   1  103  30  38   83  43.3  0.183  33  0
19   1  115  70  30   96  34.6  0.529  32  1
```

Show statistics values

We can see that there are columns that have a minimum value of zero (0). On some columns, a value of zero does not make sense and indicates an invalid or missing value.

In [38]:
```
# summarize the dataset
print(dataset.describe())
```

```
                0           1           2           3           4           5
\
count  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000
mean     3.845052  120.894531   69.105469   20.536458   79.799479   31.992578
std      3.369578   31.972618   19.355807   15.952218  115.244002    7.884160
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%      1.000000   99.000000   62.000000    0.000000    0.000000   27.300000
50%      3.000000  117.000000   72.000000   23.000000   30.500000   32.000000
75%      6.000000  140.250000   80.000000   32.000000  127.250000   36.600000
max     17.000000  199.000000  122.000000   99.000000  846.000000   67.100000

                6           7           8
count  768.000000  768.000000  768.000000
mean     0.471876   33.240885    0.348958
std      0.331329   11.760232    0.476951
min      0.078000   21.000000    0.000000
25%      0.243750   24.000000    0.000000
50%      0.372500   29.000000    0.000000
75%      0.626250   41.000000    1.000000
max      2.420000   81.000000    1.000000
```

Check dimensions

Looking at the dataset's dimensions as a measure of its size:

In [39]:
```
print(dataset.shape)
```

```
(768, 9)
```

Step 2 : Check Missing value

There are 2 ways for checking missing values.

Way 1 : compute the value = 0 in each column then print

In [40]:
```
num_missing = (dataset[[0,1,2,3,4,5,6,7,8]] == 0).sum()
# report the results
print(num_missing)
```

```
0    111
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8    500
dtype: int64
```

Way 2 : Replace 0 with nan, then call isnull() functions to mark all of the NaN values in the dataset as True and get a count of the missing values for each column.

In [41]:
```
import numpy as np
dataset[[0,1,2,3,4,5,6,7,8]] = dataset[[0,1,2,3,4,5,6,7,8]].replace(0, np.nan)
# count the number of nan values in each column
print(dataset.isnull().sum())
```

```
0    111
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8    500
dtype: int64
```

Then check dataset

In [33]:
```
print(dataset.head(20))
```

```
      0      1     2     3      4     5      6   7    8
0   6.0  148.0  72.0  35.0    NaN  33.6  0.627  50  1.0
1   1.0   85.0  66.0  29.0    NaN  26.6  0.351  31  NaN
2   8.0  183.0  64.0   NaN    NaN  23.3  0.672  32  1.0
3   1.0   89.0  66.0  23.0   94.0  28.1  0.167  21  NaN
4   NaN  137.0  40.0  35.0  168.0  43.1  2.288  33  1.0
5   5.0  116.0  74.0   NaN    NaN  25.6  0.201  30  NaN
6   3.0   78.0  50.0  32.0   88.0  31.0  0.248  26  1.0
7  10.0  115.0   NaN   NaN    NaN  35.3  0.134  29  NaN
8   2.0  197.0  70.0  45.0  543.0  30.5  0.158  53  1.0
9   8.0  125.0  96.0   NaN    NaN   NaN  0.232  54  1.0
10  4.0  110.0  92.0   NaN    NaN  37.6  0.191  30  NaN
11 10.0  168.0  74.0   NaN    NaN  38.0  0.537  34  1.0
12 10.0  139.0  80.0   NaN    NaN  27.1  1.441  57  NaN
13  1.0  189.0  60.0  23.0  846.0  30.1  0.398  59  1.0
14  5.0  166.0  72.0  19.0  175.0  25.8  0.587  51  1.0
15  7.0  100.0   NaN   NaN    NaN  30.0  0.484  32  1.0
16  NaN  118.0  84.0  47.0  230.0  45.8  0.551  31  1.0
17  7.0  107.0  74.0   NaN    NaN  29.6  0.254  31  1.0
18  1.0  103.0  30.0  38.0   83.0  43.3  0.183  33  NaN
19  1.0  115.0  70.0  30.0   96.0  34.6  0.529  32  1.0
```

Step 3 : Missing Values Causes Problems checking

apply a model on the dataset which contain missing values. It causes problem.

In [42]:
```
# example where missing values cause errors
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
# split dataset into inputs and outputs
values = dataset.values
X = values[:,0:8]
y = values[:,8]
# define the model
model = LinearDiscriminantAnalysis()
# define the model evaluation procedure
cv = KFold(n_splits=3, shuffle=True, random_state=1)
# evaluate the model
result = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
# report the mean performance
print('Accuracy: %.3f' % result.mean())
```

```
Accuracy: nan

E:\ana\lib\site-packages\sklearn\model_selection\_validation.py:372: FitFaile
dWarning:
3 fits failed out of a total of 3.
The score on these train-test partitions for these parameters will be set to
nan.
If these failures are not expected, you can try to debug them by setting erro
r_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
---
3 fits failed with the following error:
Traceback (most recent call last):
  File "E:\ana\lib\site-packages\sklearn\model_selection\_validation.py", lin
e 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "E:\ana\lib\site-packages\sklearn\discriminant_analysis.py", line 544,
in fit
    X, y = self._validate_data(
  File "E:\ana\lib\site-packages\sklearn\base.py", line 581, in _validate_dat
a
    X, y = check_X_y(X, y, **check_params)
  File "E:\ana\lib\site-packages\sklearn\utils\validation.py", line 964, in c
heck_X_y
    X = check_array(
  File "E:\ana\lib\site-packages\sklearn\utils\validation.py", line 800, in c
heck_array
    _assert_all_finite(array, allow_nan=force_all_finite == "allow-nan")
  File "E:\ana\lib\site-packages\sklearn\utils\validation.py", line 114, in _
assert_all_finite
    raise ValueError(
ValueError: Input contains NaN, infinity or a value too large for dtype('floa
t64').

  warnings.warn(some_fits_failed_message, FitFailedWarning)
```

Step 4 : Handeling missing values

way 1: Drop missing values

We can drop missing values in two ways. We can use one of them. It use dropna() function.

i) Drop Rows with any missing values

In [43]:
```
# Drop rows with any missing value
print(dataset.shape)
dataset.dropna(axis=0,inplace=True)
# summarize the shape of the data with missing rows removed
print(dataset.shape)
```

```
(768, 9)
(111, 9)
```

ii) Drops Columns with any missing values

In [45]:
```
# Drop columns with any missing value
dataset.dropna(axis=1,inplace=True)
print(dataset.shape)
```

```
(111, 9)
```

Again apply model

After removing missing values the model can perform perfectly

In [44]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
# split dataset into inputs and outputs
values = dataset.values
X = values[:,0:8]
y = values[:,8]
# define the model
model = LinearDiscriminantAnalysis()
# define the model evaluation procedure
cv = KFold(n_splits=3, shuffle=True, random_state=1)
# evaluate the model
result = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
# report the mean performance
print('Accuracy: %.3f' % result.mean())
```

```
Accuracy: 1.000
```

Way 2 : Fill another values

We can also replace missing values with mean, mode, median values, instead of droping missing values. For this we use fillna() function

In [46]:
```
dataset = read_csv('pima-indians-diabetes.csv', header=None)
dataset[[0,1,2,3,4,5,6,7,8]] = dataset[[0,1,2,3,4,5,6,7,8]].replace(0, np.nan)
# count the number of nan values in each column
print(dataset.isnull().sum())
dataset.fillna(dataset.mean(), inplace=True)
print(dataset.isnull().sum())
```

```
0    111
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8    500
dtype: int64
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
dtype: int64
```

way 3 : Replace another values using simpleimpute class

We can also uses the SimpleImputer class to replace missing values with the mean of each column then prints the number of NaN values in the transformed matrix.

In [51]:
```
from sklearn.impute import SimpleImputer
dataset = read_csv('pima-indians-diabetes.csv', header=None)
dataset[[0,1,2,3,4,5,6,7,8]] = dataset[[0,1,2,3,4,5,6,7,8]].replace(0, np.nan)
# count the number of nan values in each column
print(dataset.isnull().sum())
# retrieve the numpy array
values = dataset.values
# define the imputer
imputer = SimpleImputer(missing_values=nan, strategy='mean')
# transform the dataset
transformed_values = imputer.fit_transform(values)
# count the number of NaN values in each column
print('Missing: %d' % np.isnan(transformed_values).sum())
```

```
0    111
1      5
2     35
3    227
4    374
5     11
6      0
7      0
8    500
dtype: int64
Missing: 0
```