MACHINE LEARNING MAJOR PROJECT

Name : M Girish

Email : mgirish2221@gmail.com

Every machine learning project begins by understanding what the data and drawing the objectives. While applying machine learning algorithms to your data set, you are understanding, building and analyzing the data as to get the end result.

Following are the steps involved in creating a well-defined ML project:

1] Understand and define the problem

2] Prepare the data

3] Explore and Analyse the data

4] Apply the algorithms

5] Reduce the errors

6] Predict the result

PROBLEM STATEMENT :-

In this **Diabetes Prediction using Machine Learning Project Code**, the objective is to predict whether the person has Diabetes or not based on various features like Number of Pregnancies, Insulin Level, Age, BMI. The data set that has used in this project is taken from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage."

TOOLS AND LIBRARIES :-

- **Jupyter Notebook** : The Jupyter Notebook is an open-source web application that allows us to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.
- **Python3** : Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.
- **Pandas** : Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **Scikit-Learn Library** : Scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.
- **NumPy** : NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
-

DAIBETES DATASET DESCRIPTION :-

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. It is provided courtesy of the Pima Indians Diabetes Database and is available on Kaggle. Here is the link to the dataset. It consists of several medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. The dataset has 9 columns as shown below;

• Pregnancies	–	Number of times pregnant
• Glucose	–	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
• Blood Pressure	–	Diastolic blood pressure (mm Hg)

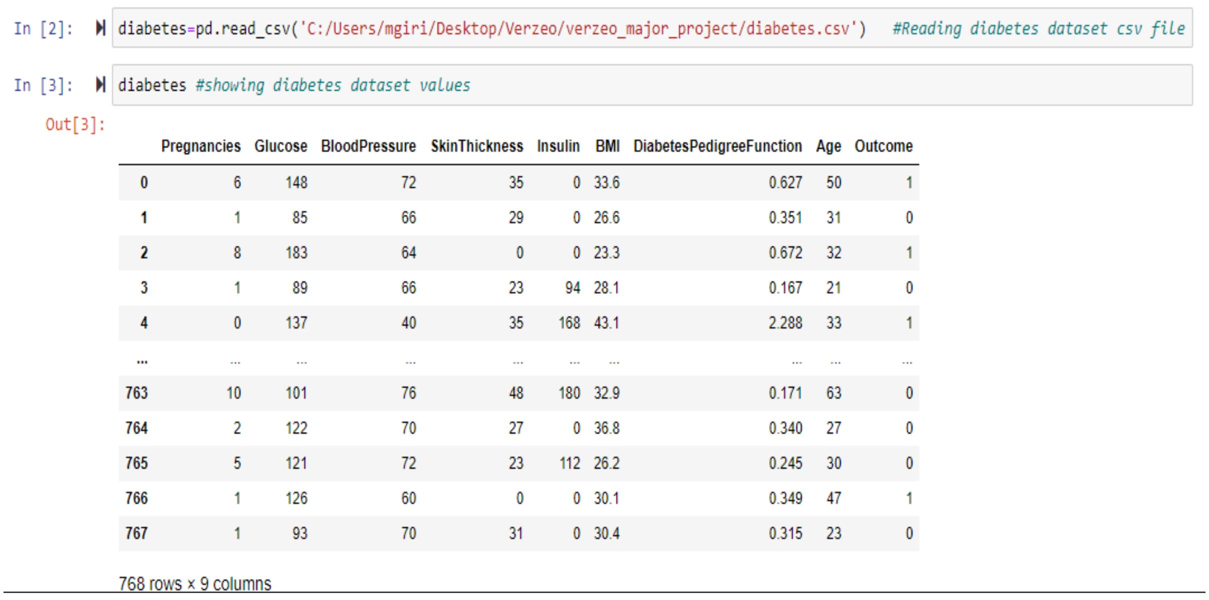
• Skin Thickness	–	Triceps skinfold thickness (mm)
• Insulin	–	2-Hour serum insulin (mu U/ml)
• BMI	–	Body mass index (weight in kg/(height in m)^2)
• Diabetes Pedigree Function	–	Diabetes pedigree function
• Age	–	Age (years)
• Outcome	–	Class variable (0 or 1) 268 of 768 are 1, the others are 0

DATA ANALYSIS AND PREPROCESSING :-

Data analysis is a big task in machine learning most of the time in building a model is taken by this process. Because, for machine learning models training and testing of data used should be clean and filtered according to model criteria.

Generally, cleaning and filtering of data is done by using Numpy , Pandas and Matplotlib libraries.

1. READING DATA :

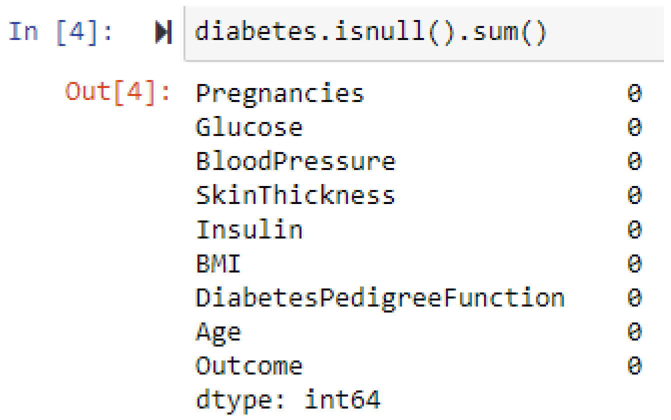


```
In [2]:  diabetes=pd.read_csv('C:/Users/mgiri/Desktop/Verzeo/verzeo_major_project/diabetes.csv')   #Reading diabetes dataset csv file

In [3]:  diabetes #showing diabetes dataset values
Out[3]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

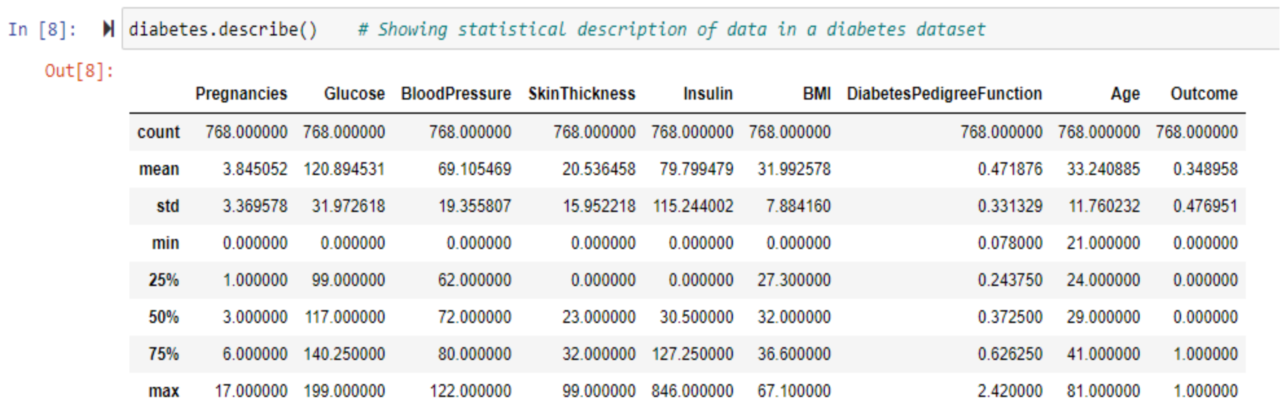
768 rows × 9 columns

2. CHECKING NULL VALUES IN FEATURES :



```
In [4]:  diabetes.isnull().sum()

Out[4]:  Pregnancies                 0
         Glucose                     0
         BloodPressure               0
         SkinThickness               0
         Insulin                     0
         BMI                         0
         DiabetesPedigreeFunction    0
         Age                         0
         Outcome                     0
         dtype: int64
```

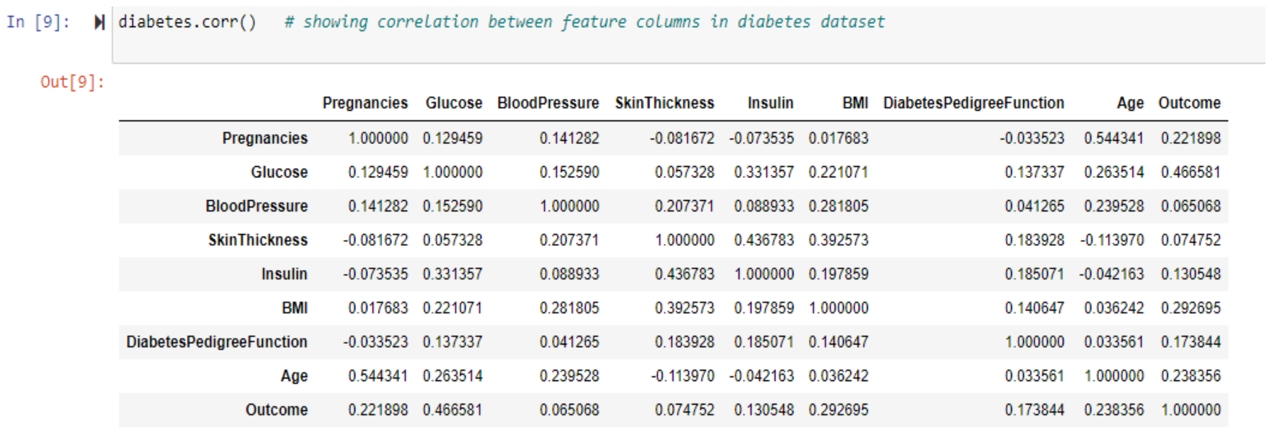
3. STATISTICAL DESCRIPTION OF DATA :



```
In [8]:  diabetes.describe()    # Showing statistical description of data in a diabetes dataset

Out[8]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	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	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

4. CORRELATION BETWEEN THE FEATURES :

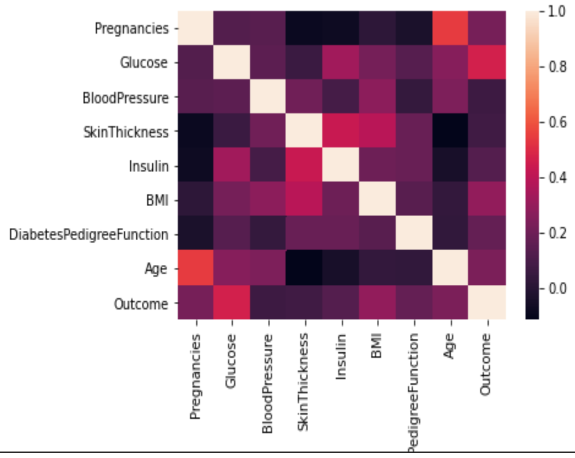


```
In [9]:  diabetes.corr()    # showing correlation between feature columns in diabetes dataset

Out[9]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

5. HEATMAP OF FEATUTES:

```
In [10]:  sns.heatmap(diabetes.corr(),xticklabels=diabetes.corr().columns,yticklabels=diabetes.corr().columns)
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2a050648c40>
```



6. FINDING MISSING VALUES IN FEATURES :

```
In [14]:  print('Number of rows missing Glucose: {0}'.format(len(diabetes.loc[diabetes['Glucose'] == 0])))
          print('Number of rows missing Blood Pressure: {0}'.format(len(diabetes.loc[diabetes['BloodPressure'] == 0])))
          print('Number of rows missing Insulin: {0}'.format(len(diabetes.loc[diabetes['Insulin'] == 0])))
          print('Number of rows missing BMI: {0}'.format(len(diabetes.loc[diabetes['BMI'] == 0])))
          print('Number of rows missing Skin Thickness: {0}'.format(len(diabetes.loc[diabetes['SkinThickness'] == 0])))
          print('Number of rows missing Age: {0}'.format(len(diabetes.loc[diabetes['Age'] == 0])))
          print('Number of rows missing Diabetes Pedigree Function: {0}'.format(len(diabetes.loc[diabetes['DiabetesPedigreeFunction']
```

```
Number of rows missing Glucose: 5
Number of rows missing Blood Pressure: 35
Number of rows missing Insulin: 374
Number of rows missing BMI: 11
Number of rows missing Skin Thickness: 227
Number of rows missing Age: 0
Number of rows missing Diabetes Pedigree Function: 0
```

7. FILLING MISSING VALUES:

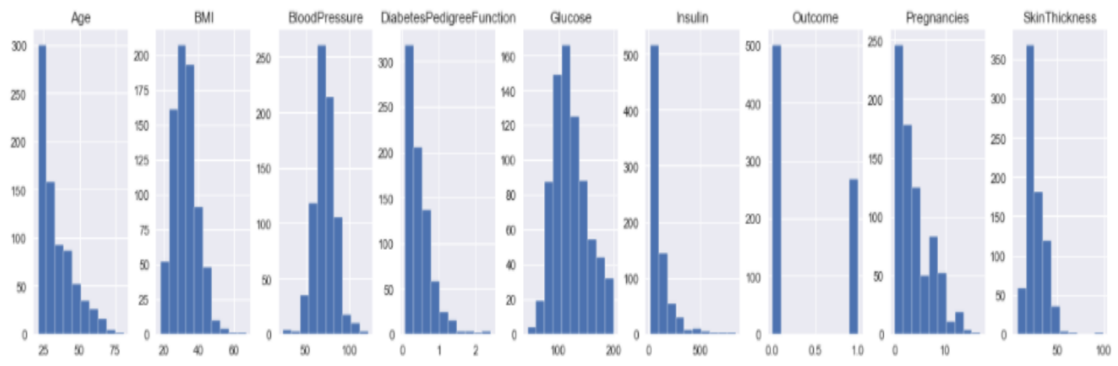
```
In [15]:  x = diabetes['Glucose'].mean()
          diabetes['Glucose'].replace(0,x,inplace=True)
          x = diabetes['BloodPressure'].mean()
          diabetes['BloodPressure'].replace(0,x,inplace=True)
          x = diabetes['Insulin'].mean()
          diabetes['Insulin'].replace(0,x,inplace=True)
          x = diabetes['BMI'].mean()
          diabetes['BMI'].replace(0,x,inplace=True)
          x = diabetes['SkinThickness'].mean()
          diabetes['SkinThickness'].replace(0,x,inplace=True)
          diabetes.head()
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

8. HISTOGRAM OF FEATURE VALUES :

```
In [21]:  diabetes.hist(layout=(1,9), figsize=(20,4))    # histogram representation of features in diabetes dataset
          pyplot.show()
```



MODEL SELECTION :-

For given Diabetes dataset the class/label values are discrete and characterised so its better to use classification models to achieve better performance of a model. I have selected a KNN model and Decision Tree model.

1. KNN Model :-

- ➢ K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on supervised Learning technique.
- ➢ K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- ➢ K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- ➢ K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- ➢ K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **How it works:-**

  The K-NN working can be explained on the basis of the below algorithm:

  Step-1: Select the number K of the neighbours

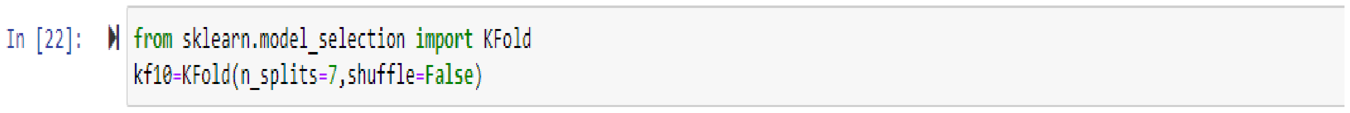
  Step-2: Calculate the Euclidean distance of K number of neighbours

  Step-3: Take the K nearest neighbours as per the calculated Euclidean distance.

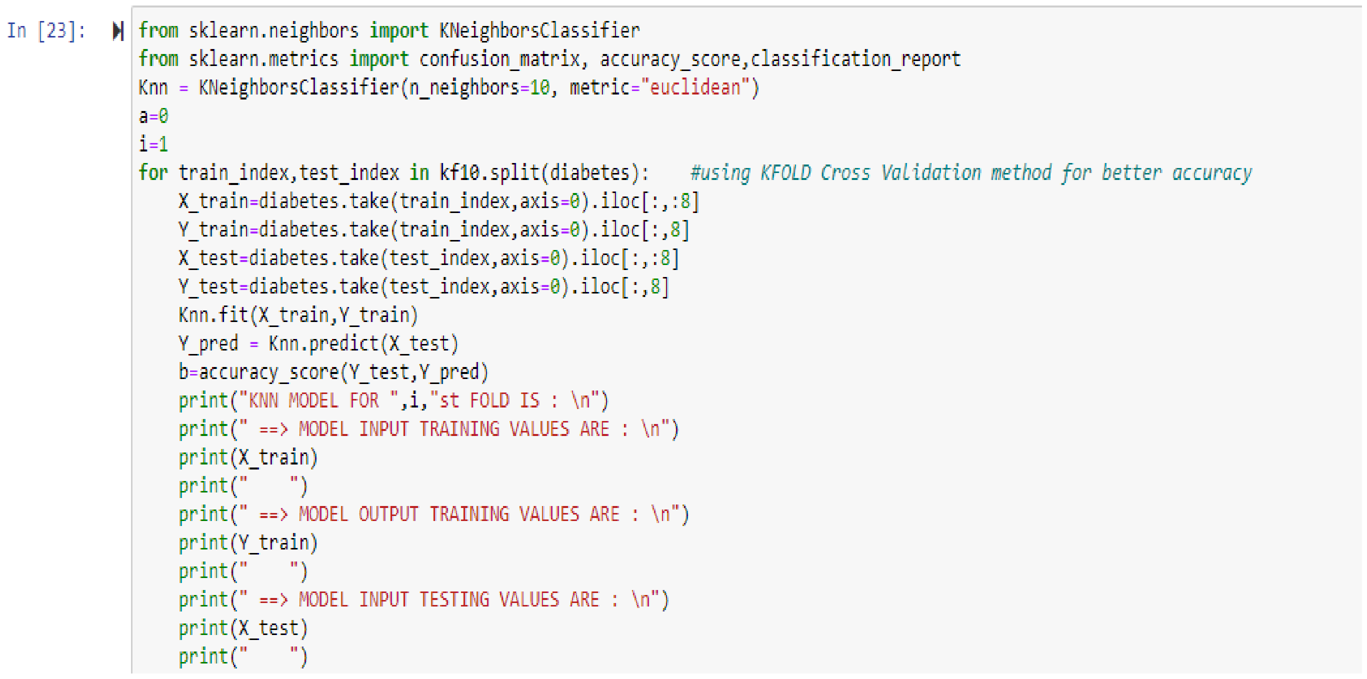
  Step-4: Among these k neighbours, count the number of the data points in each category.

  Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.

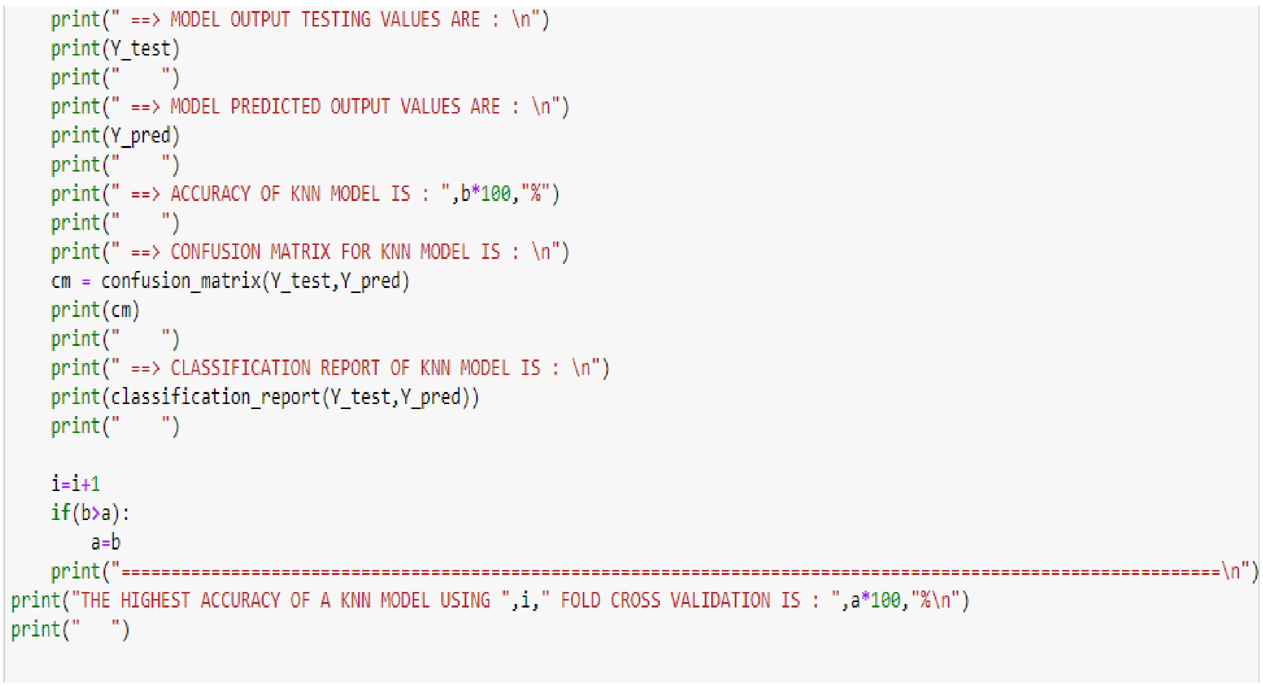
  Step-6: Our model is ready.



```
In [22]:  from sklearn.model_selection import KFold
          kf10=KFold(n_splits=7,shuffle=False)
```

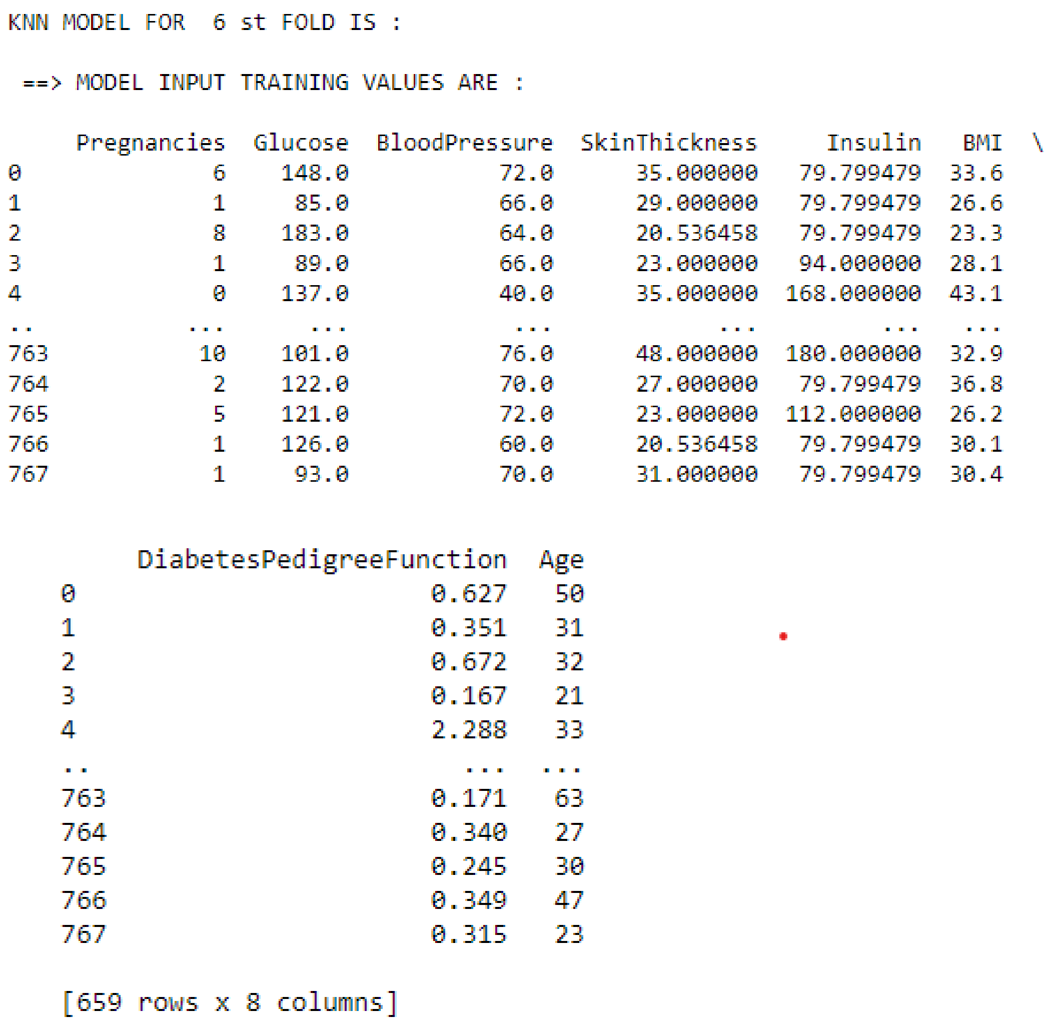


```
In [23]:  from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
          Knn = KNeighborsClassifier(n_neighbors=10, metric="euclidean")
          a=0
          i=1
          for train_index,test_index in kf10.split(diabetes):    #using KFOLD Cross Validation method for better accuracy
              X_train=diabetes.take(train_index,axis=0).iloc[:,:8]
              Y_train=diabetes.take(train_index,axis=0).iloc[:,8]
              X_test=diabetes.take(test_index,axis=0).iloc[:,:8]
              Y_test=diabetes.take(test_index,axis=0).iloc[:,8]
              Knn.fit(X_train,Y_train)
              Y_pred = Knn.predict(X_test)
              b=accuracy_score(Y_test,Y_pred)
              print("KNN MODEL FOR ",i,"st FOLD IS : \n")
              print(" ==> MODEL INPUT TRAINING VALUES ARE : \n")
              print(X_train)
              print("    ")
              print(" ==> MODEL OUTPUT TRAINING VALUES ARE : \n")
              print(Y_train)
              print("    ")
              print(" ==> MODEL INPUT TESTING VALUES ARE : \n")
              print(X_test)
              print("    ")
```

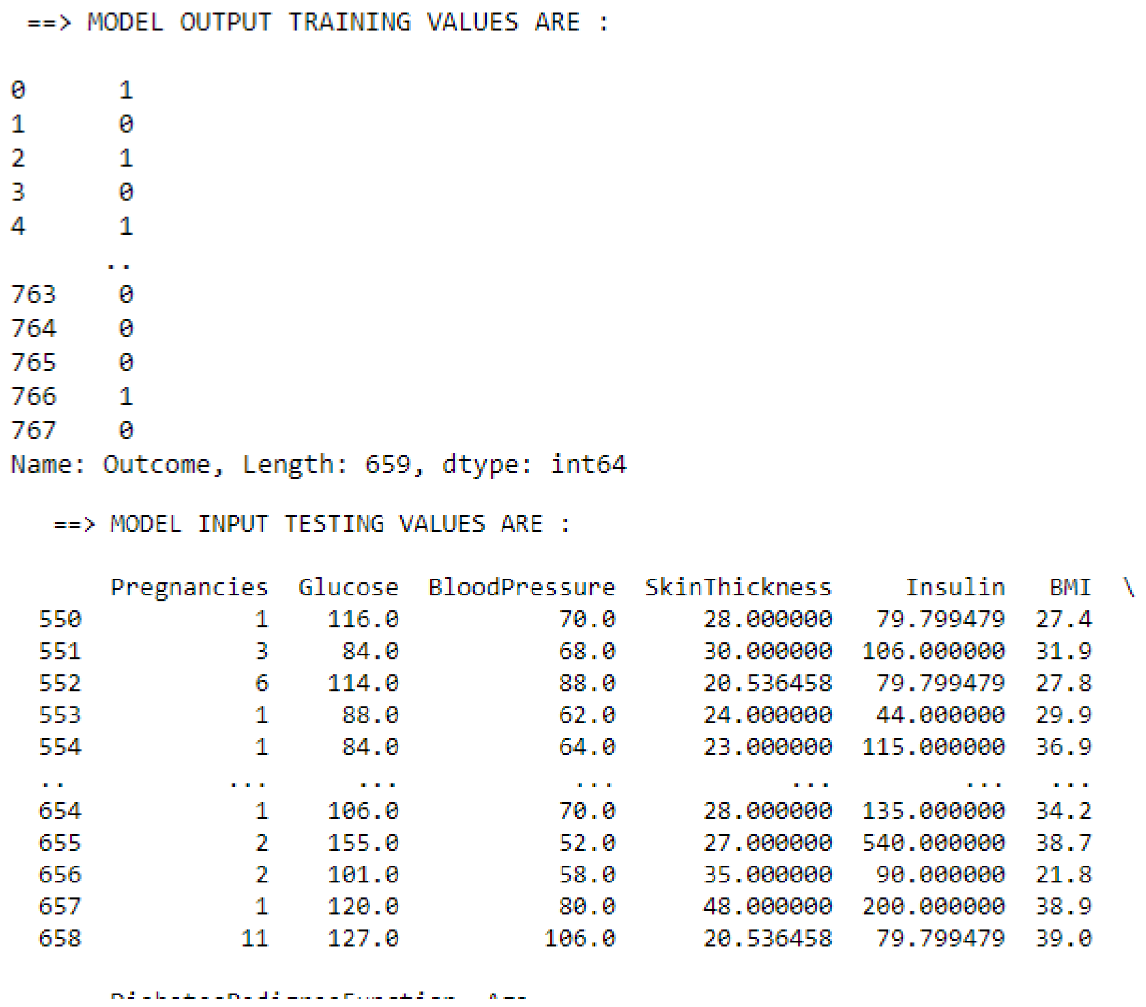


```
    print(" ==> MODEL OUTPUT TESTING VALUES ARE : \n")
    print(Y_test)
    print("    ")
    print(" ==> MODEL PREDICTED OUTPUT VALUES ARE : \n")
    print(Y_pred)
    print("    ")
    print(" ==> ACCURACY OF KNN MODEL IS : ",b*100,"%")
    print("    ")
    print(" ==> CONFUSION MATRIX FOR KNN MODEL IS : \n")
    cm = confusion_matrix(Y_test,Y_pred)
    print(cm)
    print("    ")
    print(" ==> CLASSIFICATION REPORT OF KNN MODEL IS : \n")
    print(classification_report(Y_test,Y_pred))
    print("    ")

    i=i+1
    if(b>a):
        a=b
    print("=====================================================================================================================\n")
print("THE HIGHEST ACCURACY OF A KNN MODEL USING ",i," FOLD CROSS VALIDATION IS : ",a*100,"%\n")
print("   ")
```

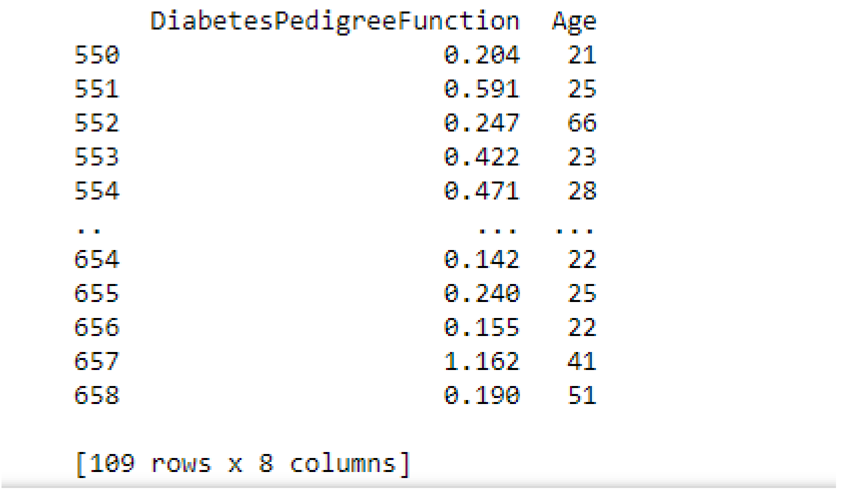
OUTPUT:



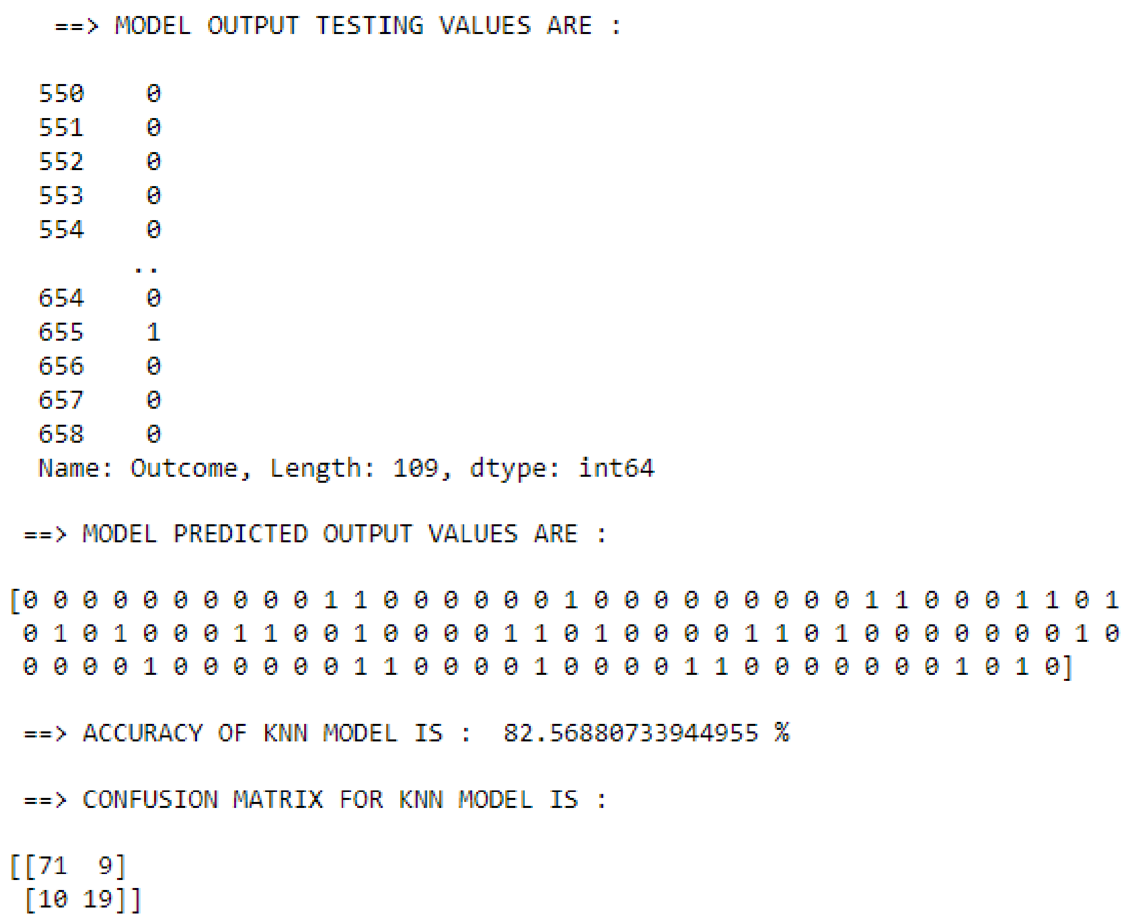
```
KNN MODEL FOR  6 st FOLD IS :

 ==> MODEL INPUT TRAINING VALUES ARE :

     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0              6    148.0           72.0      35.000000   79.799479  33.6
1              1     85.0           66.0      29.000000   79.799479  26.6
2              8    183.0           64.0      20.536458   79.799479  23.3
3              1     89.0           66.0      23.000000   94.000000  28.1
4              0    137.0           40.0      35.000000  168.000000  43.1
..           ...      ...            ...            ...         ...   ...
763           10    101.0           76.0      48.000000  180.000000  32.9
764            2    122.0           70.0      27.000000   79.799479  36.8
765            5    121.0           72.0      23.000000  112.000000  26.2
766            1    126.0           60.0      20.536458   79.799479  30.1
767            1     93.0           70.0      31.000000   79.799479  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[659 rows x 8 columns]
```



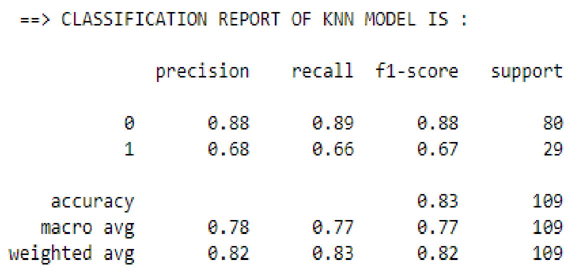
```
 ==> MODEL OUTPUT TRAINING VALUES ARE :

0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 659, dtype: int64

   ==> MODEL INPUT TESTING VALUES ARE :

     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
550            1    116.0           70.0      28.000000   79.799479  27.4
551            3     84.0           68.0      30.000000  106.000000  31.9
552            6    114.0           88.0      20.536458   79.799479  27.8
553            1     88.0           62.0      24.000000   44.000000  29.9
554            1     84.0           64.0      23.000000  115.000000  36.9
..           ...      ...            ...            ...         ...   ...
654            1    106.0           70.0      28.000000  135.000000  34.2
655            2    155.0           52.0      27.000000  540.000000  38.7
656            2    101.0           58.0      35.000000   90.000000  21.8
657            1    120.0           80.0      48.000000  200.000000  38.9
658           11    127.0          106.0      20.536458   79.799479  39.0

     DiabetesPedigreeFunction  Age
```



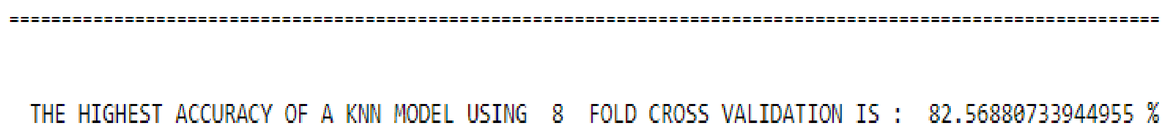
```
     DiabetesPedigreeFunction  Age
550                     0.204   21
551                     0.591   25
552                     0.247   66
553                     0.422   23
554                     0.471   28
..                        ...  ...
654                     0.142   22
655                     0.240   25
656                     0.155   22
657                     1.162   41
658                     0.190   51

[109 rows x 8 columns]
```



```
 ==> MODEL OUTPUT TESTING VALUES ARE :

550    0
551    0
552    0
553    0
554    0
      ..
654    0
655    1
656    0
657    0
658    0
Name: Outcome, Length: 109, dtype: int64

 ==> MODEL PREDICTED OUTPUT VALUES ARE :

[0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 1
 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0
 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0]

 ==> ACCURACY OF KNN MODEL IS :  82.56880733944955 %

 ==> CONFUSION MATRIX FOR KNN MODEL IS :

[[71  9]
 [10 19]]
```



```
 ==> CLASSIFICATION REPORT OF KNN MODEL IS :

              precision    recall  f1-score   support

           0       0.88      0.89      0.88        80
           1       0.68      0.66      0.67        29

    accuracy                           0.83       109
   macro avg       0.78      0.77      0.77       109
weighted avg       0.82      0.83      0.82       109
```



```
========================================================================================================

  THE HIGHEST ACCURACY OF A KNN MODEL USING  8  FOLD CROSS VALIDATION IS :  82.56880733944955 %
```

2. Decision Tree Classification Model :-

- ➢ Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- The decisions or the test are performed on the basis of features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.
- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.
- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

How it works :-

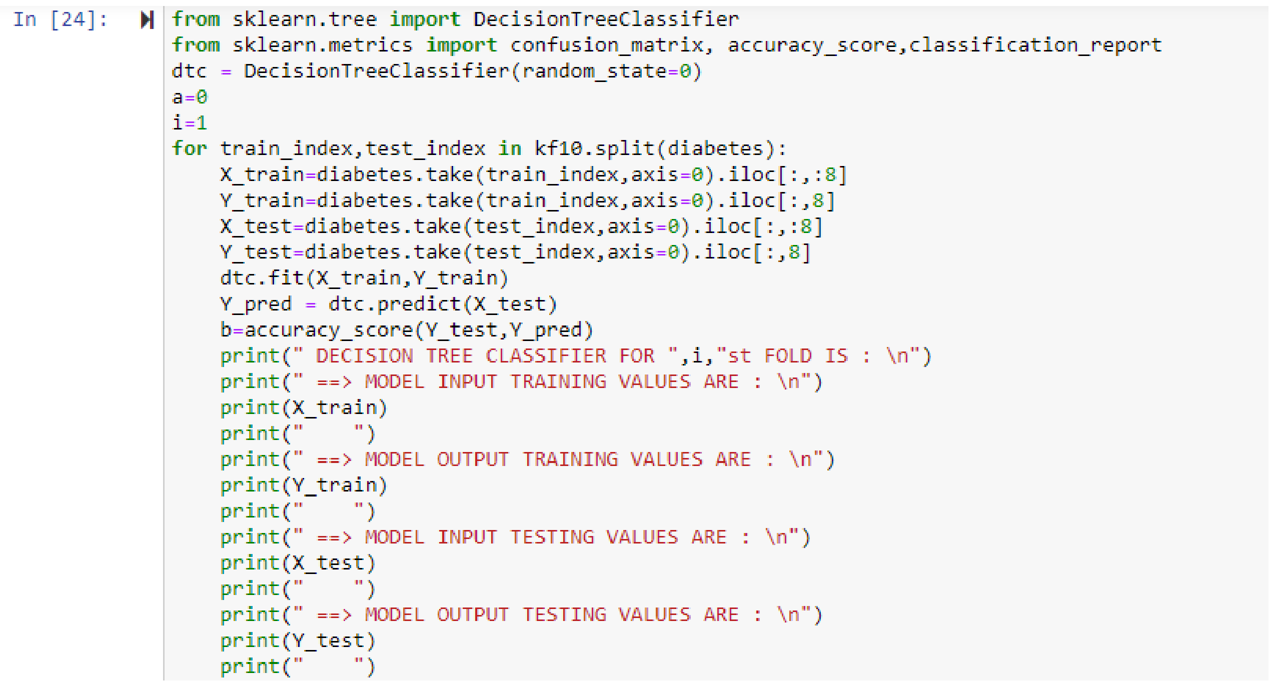
Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

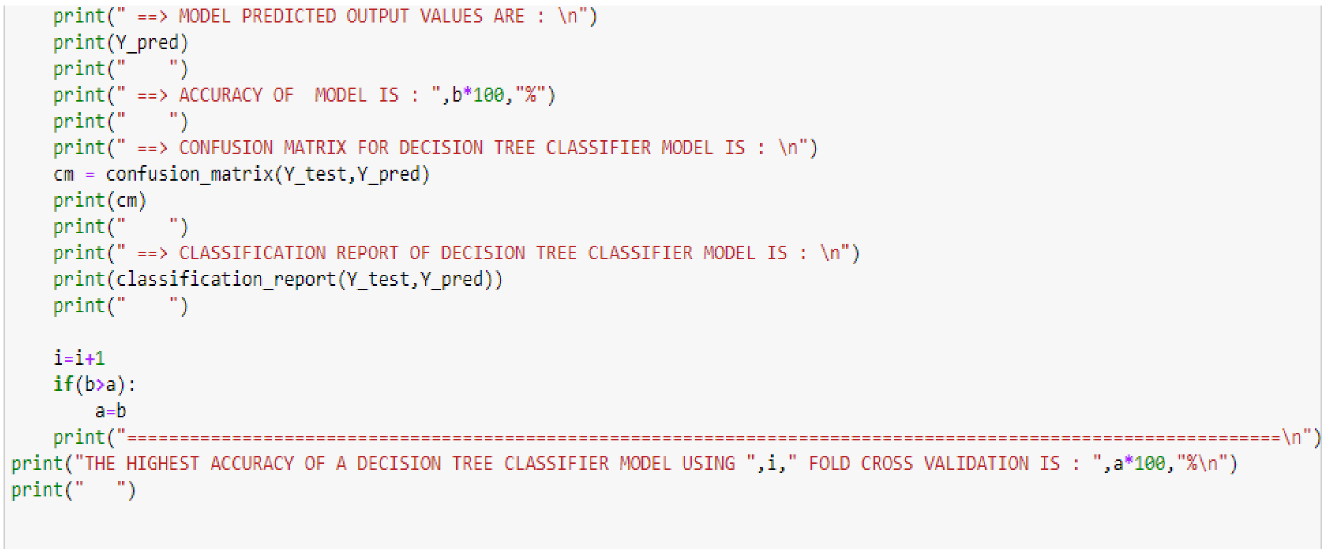
Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute

Step-5: Recursively make new decision trees using the subsets of the dataset created in step-3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.



```
In [24]:    from sklearn.tree import DecisionTreeClassifier
            from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
            dtc = DecisionTreeClassifier(random_state=0)
            a=0
            i=1
            for train_index,test_index in kf10.split(diabetes):
                X_train=diabetes.take(train_index,axis=0).iloc[:,:8]
                Y_train=diabetes.take(train_index,axis=0).iloc[:,8]
                X_test=diabetes.take(test_index,axis=0).iloc[:,:8]
                Y_test=diabetes.take(test_index,axis=0).iloc[:,8]
                dtc.fit(X_train,Y_train)
                Y_pred = dtc.predict(X_test)
                b=accuracy_score(Y_test,Y_pred)
                print(" DECISION TREE CLASSIFIER FOR ",i,"st FOLD IS : \n")
                print(" ==> MODEL INPUT TRAINING VALUES ARE : \n")
                print(X_train)
                print("    ")
                print(" ==> MODEL OUTPUT TRAINING VALUES ARE : \n")
                print(Y_train)
                print("    ")
                print(" ==> MODEL INPUT TESTING VALUES ARE : \n")
                print(X_test)
                print("    ")
                print(" ==> MODEL OUTPUT TESTING VALUES ARE : \n")
                print(Y_test)
                print("    ")
```

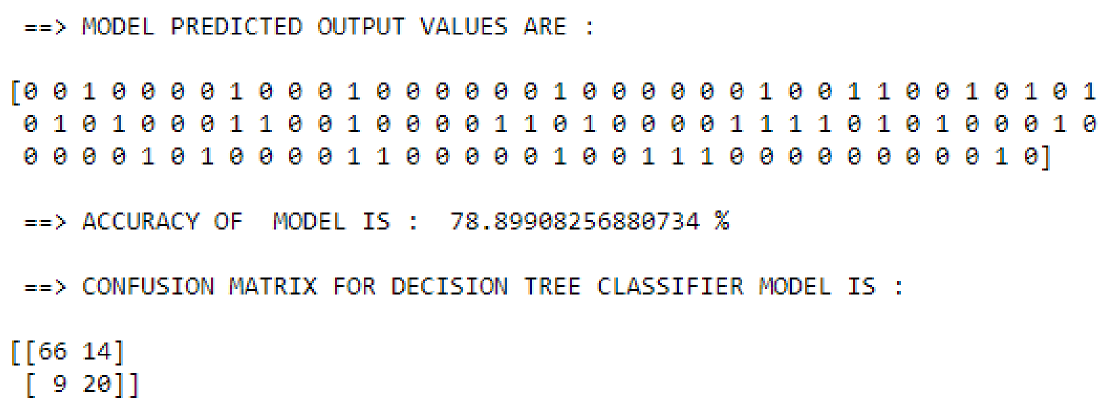


```
    print(" ==> MODEL PREDICTED OUTPUT VALUES ARE : \n")
    print(Y_pred)
    print("    ")
    print(" ==> ACCURACY OF  MODEL IS : ",b*100,"%")
    print("    ")
    print(" ==> CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL IS : \n")
    cm = confusion_matrix(Y_test,Y_pred)
    print(cm)
    print("    ")
    print(" ==> CLASSIFICATION REPORT OF DECISION TREE CLASSIFIER MODEL IS : \n")
    print(classification_report(Y_test,Y_pred))
    print("    ")

    i=i+1
    if(b>a):
        a=b
    print("==================================================================================================================\n")
print("THE HIGHEST ACCURACY OF A DECISION TREE CLASSIFIER MODEL USING ",i," FOLD CROSS VALIDATION IS : ",a*100,"%\n")
print("   ")
```

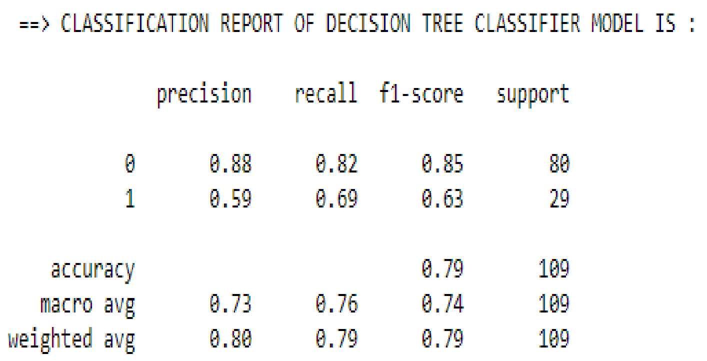
```
 DECISION TREE CLASSIFIER FOR  6 st FOLD IS :

 ==> MODEL INPUT TRAINING VALUES ARE :

     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
0              6    148.0           72.0      35.000000   79.799479  33.6
1              1     85.0           66.0      29.000000   79.799479  26.6
2              8    183.0           64.0      20.536458   79.799479  23.3
3              1     89.0           66.0      23.000000   94.000000  28.1
4              0    137.0           40.0      35.000000  168.000000  43.1
..           ...      ...            ...            ...         ...   ...
763           10    101.0           76.0      48.000000  180.000000  32.9
764            2    122.0           70.0      27.000000   79.799479  36.8
765            5    121.0           72.0      23.000000  112.000000  26.2
766            1    126.0           60.0      20.536458   79.799479  30.1
767            1     93.0           70.0      31.000000   79.799479  30.4
```

```
     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[659 rows x 8 columns]


 ==> MODEL OUTPUT TRAINING VALUES ARE :

0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 659, dtype: int64
```

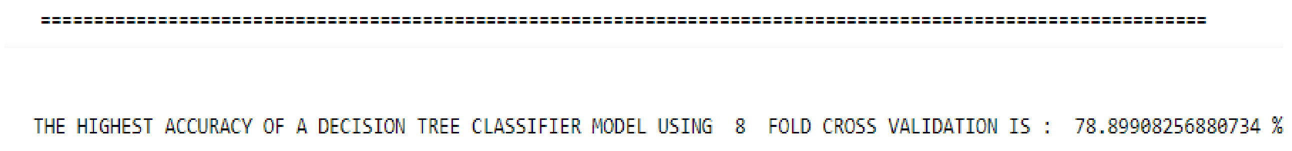
```
 ==> MODEL INPUT TESTING VALUES ARE :

     Pregnancies  Glucose  BloodPressure  SkinThickness     Insulin   BMI  \
550            1    116.0           70.0      28.000000   79.799479  27.4
551            3     84.0           68.0      30.000000  106.000000  31.9
552            6    114.0           88.0      20.536458   79.799479  27.8
553            1     88.0           62.0      24.000000   44.000000  29.9
554            1     84.0           64.0      23.000000  115.000000  36.9
..           ...      ...            ...            ...         ...   ...
654            1    106.0           70.0      28.000000  135.000000  34.2
655            2    155.0           52.0      27.000000  540.000000  38.7
656            2    101.0           58.0      35.000000   90.000000  21.8
657            1    120.0           80.0      48.000000  200.000000  38.9
658           11    127.0          106.0      20.536458   79.799479  39.0

      DiabetesPedigreeFunction  Age
 550                     0.204   21
 551                     0.591   25
 552                     0.247   66
 553                     0.422   23
 554                     0.471   28
 ..                        ...  ...
 654                     0.142   22
 655                     0.240   25
 656                     0.155   22
 657                     1.162   41
 658                     0.190   51

 [109 rows x 8 columns]


 ==> MODEL OUTPUT TESTING VALUES ARE :

550    0
551    0
552    0
553    0
554    0
      ..
654    0
655    1
656    0
657    0
658    0
Name: Outcome, Length: 109, dtype: int64
```



```
 ==> MODEL PREDICTED OUTPUT VALUES ARE :

[0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 0 1
 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 0
 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0]

 ==> ACCURACY OF  MODEL IS :  78.89908256880734 %

 ==> CONFUSION MATRIX FOR DECISION TREE CLASSIFIER MODEL IS :

[[66 14]
 [ 9 20]]
```



```
 ==> CLASSIFICATION REPORT OF DECISION TREE CLASSIFIER MODEL IS :

              precision    recall  f1-score   support

           0       0.88      0.82      0.85        80
           1       0.59      0.69      0.63        29

    accuracy                           0.79       109
   macro avg       0.73      0.76      0.74       109
weighted avg       0.80      0.79      0.79       109
```



```
========================================================================================================

 THE HIGHEST ACCURACY OF A DECISION TREE CLASSIFIER MODEL USING  8  FOLD CROSS VALIDATION IS :  78.89908256880734 %
```

PERFORMANCE COMPARISION BETWEEN MODELS :-

In training of model the diabetes data is divided using cross validation method of Kfold function. Here, I have used the 8 fold cross validation.

So, that accuracy of KNN algorithm is 83 percent and the accuracy of Decision Tree algorithm is 79 percent.

Hence we can conclude that for given Diabetes dataset KNN model perform better than the Decision Tree model for 8 KFold cross validation of training and test data.