Importing the required libraries

In [1]:

```
import  numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Reading the dataset

In [3]:

```
data=pd.read_csv('diabetes_data_upload.csv')
data.head()
```

Out[3]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching
0	40	Male	No	Yes	No	Yes	No	No	No	Yes
1	58	Male	No	No	No	Yes	No	No	Yes	No
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes

Defining rows and columns

In [4]:

```
data
```

Out[4]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itchi
0	40	Male	No	Yes	No	Yes	No	No	No	
1	58	Male	No	No	No	Yes	No	No	Yes	
2	41	Male	Yes	No	No	Yes	Yes	No	No	
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	
...	...	...	...	...	...	...	...	...	...	
515	39	Female	Yes	Yes	Yes	No	Yes	No	No	
516	48	Female	Yes	Yes	Yes	Yes	Yes	No	No	
517	58	Female	Yes	Yes	Yes	Yes	Yes	No	Yes	
518	32	Female	No	No	No	Yes	No	No	Yes	
519	42	Male	No	No	No	No	No	No	No	

520 rows × 17 columns

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 520 non-null    int64 
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

Defining null values

In [6]:

```
print("Number of null values :" , data.isnull().sum().sum())
data.describe(include='all')
```

```
Number of null values : 0
```

Out[6]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	vi blur
count	520.000000	520	520	520	520	520	520	520	
unique	NaN	2	2	2	2	2	2	2	
top	NaN	Male	No	No	No	Yes	No	No	
freq	NaN	328	262	287	303	305	283	404	
mean	48.028846	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
std	12.151466	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
min	16.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
25%	39.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
50%	47.500000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
75%	57.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
max	90.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

Defining columns

In [7]:

```
{ column: len(data[column].unique())for column in data.columns}
```

Out[7]:

```
{'Age': 51,
 'Gender': 2,
 'Polyuria': 2,
 'Polydipsia': 2,
 'sudden weight loss': 2,
 'weakness': 2,
 'Polyphagia': 2,
 'Genital thrush': 2,
 'visual blurring': 2,
 'Itching': 2,
 'Irritability': 2,
 'delayed healing': 2,
 'partial paresis': 2,
 'muscle stiffness': 2,
 'Alopecia': 2,
 'Obesity': 2,
 'class': 2}
```

Data preprocessing

In [29]:

```
def preprocessing(df):
    df= df.copy()

    # Gender column Binary Encoding
    df['Gender'] = df ['Gender'].replace({'Female':0,'Male':1 })

    #Symptom Column Binary Encoding
    for column in df.columns.drop(['Age','Gender','class']):
     df[column]= df[column].replace({'No':0 , 'Yes': 1})

    #train
    y=df["class"]
    X=df.drop("class", axis=1)
    X_train, X_test,y_train,y_test = train_test_split(X,y
                        ,train_size=0.7,shuffle=True,random_state=1)
    scaler=StandardScaler()
    scaler.fit(X_train)
    X_train=pd.DataFrame(scaler.transform(X_train)
                        ,index=X_train.index , columns=X_train.columns)
    X_test=pd.DataFrame(scaler.transform(X_test)
                        ,index=X_test.index, columns=X_test.columns)

    return X_train,X_test,y_train,y_test
```

In [17]:

```
X_train,X_test,y_train,y_test= preprocessing(data)
```

In [18]:

```
X_train
```

Out[18]:

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	
122	-0.658902	0.740902	-0.994521	1.129159	-0.846747	0.841974	1.104315	-0.560428	-
168	-0.913060	0.740902	-0.994521	-0.885615	-0.846747	0.841974	-0.905539	-0.560428	-
23	0.018852	0.740902	-0.994521	1.129159	1.180990	0.841974	-0.905539	-0.560428	
13	1.120204	0.740902	1.005510	1.129159	1.180990	0.841974	1.104315	1.784351	
61	-1.082499	-1.349706	1.005510	1.129159	1.180990	0.841974	1.104315	-0.560428	
...	...	...	...	...	...	...	...	...	
129	0.018852	0.740902	1.005510	1.129159	1.180990	0.841974	-0.905539	-0.560428	-
144	1.713239	0.740902	1.005510	1.129159	-0.846747	-1.187685	1.104315	-0.560428	
72	1.459081	-1.349706	-0.994521	-0.885615	-0.846747	-1.187685	-0.905539	1.784351	-
235	-1.844973	0.740902	-0.994521	-0.885615	-0.846747	-1.187685	-0.905539	-0.560428	-
37	1.289643	0.740902	1.005510	1.129159	1.180990	0.841974	1.104315	-0.560428	

364 rows × 16 columns

In [19]:

```
y_test
```

Out[19]:

```
273    Negative
272    Negative
329    Negative
480    Negative
173    Positive
         ...   
335    Negative
407    Negative
330    Negative
257    Positive
95     Positive
Name: class, Length: 156, dtype: object
```

Decision tree algorithm

In [21]:

```
models= {

    '      DecisionTree ': DecisionTreeClassifier()



}
for name,model in models.items():
    model.fit(X_train,y_train)
    print(name+ ': trained')
```

```
      DecisionTree : trained
```

In [22]:

```
for name,model in models.items():
    print(name+ ": {:.2f}%".format(model.score(X_test,y_test) * 100))
```

```
      DecisionTree : 96.15%
```

Random forest algortihm

In [23]:

```
models= {
    '    Randomforest   ': RandomForestClassifier()


}
for name,model in models.items():
    model.fit(X_train,y_train)
    print(name+ ': trained')
```

```
    Randomforest   : trained
```

In [24]:

```
for name,model in models.items():
    print(name+ ": {:.2f}%".format(model.score(X_test,y_test) * 100))
```

```
    Randomforest   : 98.08%
```