1.Load the data set

Here, I used 'Pandas' library to load the CSV file into the DataFrame.. I mounted my google drive to read the file for the same

```
import pandas as pd
df=pd.read_csv('/content/drive/MyDrive/Data Set-Luminar/diabetes.csv')
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	D
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

Features and class labels are provided within the data set, so I can use the supervised machine learning process here.

2.In the below step, I checked for the data types of each column and I noticed that there is no 'object' data type. If there is any data type as 'object', we need to convert it to machine readable format

```
df.dtypes
```

```
    Pregnancies                   int64
    Glucose                       int64
    BloodPressure                 int64
    SkinThickness                 int64
    Insulin                       int64
    BMI                         float64
    DiabetesPedigreeFunction    float64
    Age                           int64
    Outcome                       int64
    dtype: object
```

3.Here, I checked for the sum of null values from each column and I donot see any null values. If there is any null value reported, we should fill it with mean/mode values of that particular column or have to check and remove the row if there is no dependancy for the same in the data set.

```
df.isna().sum()
```

```
    Pregnancies                 0
    Glucose                     0
    BloodPressure               0
    SkinThickness               0
    Insulin                     0
```

```
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

4.Afer the check of null values, I divided the data set into 2- features and class labels and converted both into arrays using the function 'values'. Here X holds the features and y holds the class labels

```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

5.I used train_test_split function from library Scikit Learn to split the feature column data set into train data and test data. The main intention to do this step is to train a particular percentage of data to perform some analysis. Here the percentage of data taken is 0.3 and we have given random state function to keep the data always from state 1, so that the analysis value will be same on every executions.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.3,random_state=1)
```

6.Since the data set values are huge, it will be difficult to perform analysis and hence we transform the X values into simple values using scalers. MinMaxScaler is one of the function used for scaling. The equation of MinMaxScaling is Xnew = (Xi-Xmin)/(Xmax-Xmin).

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
```

```
X_train_new=ms.fit_transform(X_train)
X_test_new=ms.fit_transform(X_test)
```

7.Here, we need to predict the values only on a categorical condition and hence we used a classifier and KNeighborsClassifier is one among the classifiers which provide good accuracy and result.

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_new,y_train)
y_pred=knn.predict(X_test_new)
y_pred
```

```
    array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
           1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
           0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
           1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
           1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
           0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1,
           0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
           1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
           0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

8.To find the accuracy of the model created, we need to use the function 'accuracy score' and we should comparre the test class labels with the predicted values of y.

```
  from sklearn.metrics import accuracy_score
  print(accuracy_score(y_test,y_pred)*100)
```

```
74.45887445887446
```

For prediction, I used respective values for each columns Pregnancies: 2 Glucose:123 BloodPressure:73 SkinThickness:37 Insulin:100 BMI:27.3 DiabetesPedigreeFunction:0.351 Age:42

Note: I have used transform in the below step as I already transformed my previous X values into simple values. So the same step is repeated for given values as well.

```
knn.predict(ms.transform([[2,123,73,37,100,27.3,0.351,42]]))
```

```
array([0])
```

Conclusion: The diabetic patients outcome value in the raw file is 1 and no diabetes is 0.Here the result is array[0] and hence it indicates no diabetes. As per the above result, there is no chance for a patient to be diabetic with the above given values.