```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
import os
```

```
[ ] from google.colab import drive
    drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[ ] path = "/content/drive/MyDrive/Dataset/diabetes.csv"
    df = pd.read_csv(path)
    df.head(10)
```

[ ]

	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
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

```
[ ] #values of columns like 'Glucose','BloodPressure' cannot be accepted as zeros because it will affect the outcome.
    #We can replace such values with the mean of the respective column.
    zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
    for column in zero_not_accepted:
      df[column] = df[column].replace(0,np.NaN)
      mean = int(df[column].mean(skipna=True))
      df[column] = df[column].replace(np.NaN, mean)
      df.head(5)
```

```
[ ] df.head(5)
```

[ ]

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

```
[ ] # split dataset
    X = df.iloc[:, 0:8]
    Y = df.iloc[:, 8]
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=0, test_size=0.2)
```

```
[ ] # Feature Scaling
    sc_X = StandardScaler()
    X_train = sc_X.fit_transform(X_train)
    X_test = sc_X.transform(X_test)
```

```
import math
math.sqrt(len(Y_train))
math.sqrt(len(Y_test))
```

```
12.409673645990857
```

```
[ ] # Define the model: Init KNN
    classifier = KNeighborsClassifier(n_neighbors=11, p=2, metric='euclidean')
```

```
[ ] #Fit Model
    classifier.fit(X_train, Y_train)
    KNeighborsClassifier(algorithm= 'auto', leaf_size=30, metric='euclidean',metric_params=None, n_jobs=1, n_neighbors=11, p=2, weights='uniform')
```

KNeighborsClassifier

KNeighborsClassifier(metric='euclidean', n_jobs=1, n_neighbors=11)

```
[ ] #predict the test and set results
    y_pred = classifier.predict(X_test)
    y_pred
```

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

```
[ ] #Evaluate Model
    cm = confusion_matrix(Y_test, y_pred)
    print (cm)
    print(f1_score(Y_test, y_pred))
```

```
[[94 13]
 [15 32]]
0.6956521739130436
```

```
[ ] print(f1_score(Y_test,y_pred))
```

```
0.6956521739130436
```

```
[ ] print(accuracy_score(Y_test,y_pred))
```

```
0.8181818181818182
```