**Problem Statement :**

This dataset is originally 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.

**Explaination:**

The following features have been provided to help us predict whether a person is diabetic or not:

* **Pregnancies:**Number of times pregnant
* **Glucose:** Plasma glucose concentration over 2 hours in an oral glucose tolerance test
* **BloodPressure:**Diastolic blood pressure (mm Hg)
* **SkinThickness:** Triceps skin fold 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 (a function which scores likelihood of diabetes based on family history)
* **Age:** Age (years)
* **Outcome:**  0 if non-diabetic, 1 if diabetic

**IDE:** Google Colab

**Packages used:**

1. Pandas
2. Numpy
3. Scikit-learn

**Libraries used:**

1. From sklearn.model\_selection: train\_test\_split
2. From sklearn.preprocessing: StandartScaler
3. From sklearn.neighbors: KNeighborsClassifier
4. From sklearn.metrics: confusion\_matrix
5. From sklearn.metrics: accuracy\_score
6. math

**Solution:**

1. We imported basic libraries: Pandas and Numpy
2. Using pandas we opened our dataset and initialised to variable df.
3. While exploring data we found out that many columns contained zero entries such as glucose, insulin, bmi, bloodpressure, skin thickness etc. Zero values of these features is not possible hence we had to first change these zero entries to null values(np.NaN) and then these null values to mean.
4. We then divided our data in training data and testing data using train\_test\_split library. Test split = 0.2 which means we have split 20% of our data for testing the model and the rest 80% to train our model.
5. Next we scaled the input using StandardScaler library because we need to fit in the data in standard scale(between -1 and 1). Only the input has to be transformed(test\_x and train\_x).
6. Before starting knn classification we need to find out n\_neighbors(k) value. We found that using math.sqrt() as 12.409, however, n\_neighbors must contain odd value hence we can say k=12.409 $≈$11
7. We then applied KNeighborsClassifier library with n\_neighbors = 11, p=2(since outcome is 1-diabetic or 0-nondiabetic), metric = Euclidean.
8. We then predicted and printed our predicted data.
9. To evaluate the efficiency of our model we used confusion matrix library and accuracy score library.
10. Confusion matrix value = $\begin{matrix}95&12\\18&29\end{matrix}$ and accuracy score = 0.805. Hence accuracy = 80.5%

**Code(Screenshots):**











**Conclusion:**

Using correlation method we can see that glucose levels, age, BMI and number of pregnancies all have significant correlation with the outcome variable.

Blood pressure has a negative influence on the prediction, i.e. higher blood pressure is correlated with a person not being diabetic.



Looking at the confusion matrix we can say that:

* True positive = 95 diabetic patients were correctly classified by the model.
* True Negative = 29 non diabetic patients were correctly classified by the model.
* False Positive = 12 non-diabetic patients were incorrectly classified as diabetic patients by the model.
* False Negative = 18 diabetic patients were incorrectly classified non-diabetic patients by the model.

The accuracy score of the model is 0.805 hence the accuracy of the model is **80.5%** which is pretty decent.