- Dataset can be found at [Pima Indians Diabetes Database (https://www.kaggle.com/uciml/pima-indians-diabetes-database)](https://www.kaggle.com/uciml/pima-indians-diabetes-database)
- More about K-Means clustering at [K-Nearest Neighbors (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)

In [1]:

```
import pandas as pd

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df = pd.read_csv("diabetes.csv")
df.head()
```

Out[2]:

	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

What does the dataset contain ?

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

What is the algorithm

```
K-Nearest Neighbours algorithms is a supervised algorithm used to classify a point based on its neighbours.
```

How does it work

```
In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor.
Suppose P1 is the point, for which label needs to predict. First, we find the K closest points to P1 and then
the label of the major number of these closest points is assigned to P1.
```

Advantages and Disadvantges of the algorithm

```
Advantages:
* It is extremely easy to implement
* Requires no training prior to making real time predictions. This makes the KNN algorithm much faster than
other algorithms that require training
```

```
* There are only two parameters required to implement KNN i.e. the value of K and the distance function (e.g. Euclidean or Manhattan etc.)

Disadvantages:
* The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension
* KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.
* In large datasets the cost of calculating distance between new point and each existing point becomes higher.
```

How is it performed on the dataset

In [4]:
```
df.head()
```

Out[4]:

	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

In [5]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)
```

In [7]:
```
ss = StandardScaler()

X_train_std = ss.fit_transform(X_train)
X_test_std = ss.transform(X_test)
```

In [8]:
```python
knn = KNeighborsClassifier(n_neighbors=15)
knn.fit(X_train_std, y_train)
```
Out[8]:
```
KNeighborsClassifier(n_neighbors=15)
```
In [9]:
```python
knn.score(X_test_std, y_test) ## Accuracy score on test dataset
```
Out[9]:
```
0.7402597402597403
```
In [10]:
```python
X_full = ss.transform(X)
predictions = knn.predict(X_full)
df['Prediction'] = predictions
df.head()
```
Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	Prediction
0	6	148	72	35	0	33.6	0.627	50	1	1
1	1	85	66	29	0	26.6	0.351	31	0	0
2	8	183	64	0	0	23.3	0.672	32	1	1
3	1	89	66	23	94	28.1	0.167	21	0	0
4	0	137	40	35	168	43.1	2.288	33	1	0

In [11]:
```python
print("The accuracy score of KNN on the dataset is: {}".format(knn.score(X_full, y)))
```
```
The accuracy score of KNN on the dataset is: 0.77734375
```

Summary

- The accuracy of training and test sets are similar
- The model is not suffering from either of overfitting and underfitting
- Better accuracy could be achieved with fine hyperparameter tuning
- The model might also perform better by feature engineering

In [ ]: