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

In [1]:
```
import math
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn import metrics
```

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

```
* A perceptron is a neural network unit (an artificial neuron) that does certain computation. During the
learning phase, the network learns by adjusting these weights in order to be able to predict the correct class
for input data.
* It is supervised linear classifier.
```

How does it work

```
* Take inputs
* Add bias (if required)
* Assign random weights to input features
* Run the code for training.
* Find the error in prediction.
* Update the weight by the error.
* Repeat the training phase with updated weights.
* Make predictions.
```

Advantages and Disadvantges of the algorithm

```
Advantages:
* Perceptron can be used only for simple problems.
* It's computation time is very fast.

Disadvantages:
* The output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer
function
* Perceptrons can only classify linearly separable sets of vectors.
```

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]:
```
df['Outcome'].replace({0: -1}, inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

	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	-1
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	-1
4	0	137	40	35	168	43.1	2.288	33	1

In [7]:
```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)
```

In [9]:
```
ss = StandardScaler()

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

In [10]:
```
X_train.shape
```

Out[10]:
```
(537, 8)
```

In [11]:
```
classifier = Perceptron()
classifier.fit(X_train_std, y_train)
```

Out[11]:
```
Perceptron()
```

In [12]:
```
classifier.score(X_test, y_test)
```

Out[12]:
```
0.3463203463203463
```

In [13]:
```
classifier.score(X_all, y)
```

Out[13]:
```
0.734375
```

Summary

- Clearly perceptron is not able to classify the dataset
- This is due to the limitation of the perceptron as it is works only on linearly separable data
- Feature engineering could help in improving the model

In [ ]: