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

In [1]:
```
import pandas as pd

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
```

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

Logistic regression is a supervised classification algorithm. Although it can be extended to classify among more than 2 classes, the algorithm in its nature can only predict if a data point belongs to a class or not. Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature. Dichotomous means there are only two possible classes. For example, it can be used for diabetes detection problems. It computes the probability of an event occurrence.

Advantages and Disadvantges of the algorithm

Advantages:
* Because of its efficient and straightforward nature, doesn't require high computation power, easy to implement, easily interpretable, used widely by data analyst and scientist.
* Also, it doesn't require scaling of features. Logistic regression provides a probability score for observations.

Disadvantages:
* Logistic regression is not able to handle a large number of categorical features/variables.

```
* It is vulnerable to overfitting.
* It can't solve the non-linear problem with the logistic regression that is why it requires a transformation of non-linear features.
* Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.
```

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]:
```
classifier = LogisticRegression()
classifier.fit(X_train_std, y_train)
```

Out[8]:
```
LogisticRegression()
```

In [9]:
```
classifier.score(X_test_std, y_test) ## Accuracy score on test dataset
```

Out[9]:
```
0.7359307359307359
```

In [10]:
```
X_full = ss.transform(X)
predictions = classifier.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	1

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

```
The accuracy score of KNN on the dataset is: 0.7682291666666666
```

Summary

- The score on train and test datasets are similar
- The model is not suffering for either of underfitting or overfitting
- The data doesn't seem to be linearly separable
- The performance of the model can be improved by feature engineering

In [ ]:
```

```