

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

```
In [2]: from sklearn.linear_model import LogisticRegression
```

```
In [3]: df=pd.read_csv(r"E:\154\C5_health care diabetes - C5_health care diabetes.csv")
df
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.17
764	2	122	70	27	0	36.8	0.34
765	5	121	72	23	112	26.2	0.24
766	1	126	60	0	0	30.1	0.34
767	1	93	70	31	0	30.4	0.31

768 rows × 9 columns



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

Out[4]:

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



In [5]: 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
```

In [6]: df.describe()

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabetes
<b>count</b>	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
<b>mean</b>	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
<b>std</b>	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
<b>min</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
<b>25%</b>	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
<b>50%</b>	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
<b>75%</b>	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
<b>max</b>	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [7]: df.columns

Out[7]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'], dtype='object')

In [8]: feature\_matrix = df.iloc[:,0:8]  
target\_vector = df.iloc[:,-1]

In [9]: fs=StandardScaler().fit\_transform(feature\_matrix)  
logr=LogisticRegression()  
logr.fit(fs,target\_vector)

Out[9]: LogisticRegression()

```
In [12]: observation=[[1,2,3,4,5,6,7,8]]
```

```
In [13]: prediction=logr.predict(observation)
print(prediction)

[1]
```

```
In [14]: logr.classes_
```

```
Out[14]: array([0, 1], dtype=int64)
```

```
In [15]: logr.predict_proba(observation)[0][0]
```

```
Out[15]: 0.00029236948687560993
```

```
In [16]: logr.predict_proba(observation)[0][1]
```

```
Out[16]: 0.9997076305131244
```