```
# Load the diabetes dataset

from sklearn import tree
from pandas import read_csv
import os
import numpy as np
df = read_csv("diabetes.csv")
```

```
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	7	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

```
# Separate the target variable (Outcome) from the features
x = np.array(df.drop(["Outcome"], 1))
y = np.array(df["Outcome"])
```

```
<ipython-input-13-d8842ffd1975>:2: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument
  x = np.array(df.drop(["Outcome"], 1))
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 100)
```

```
# Standardize the features (mean=0, std=1)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
x_train
```

```
array([[ 1.88463592,  1.27687979,  0.77783004, ..., -0.51456163,
        -0.84288402,  1.79638519],
       [ 0.07136908,  0.7783476 ,  0.82920565, ..., -0.3654626 ,
        -0.82219993, -0.54025457],
       [-0.5330532 , -0.34334984,  0.26407397, ...,  0.06940956,
         0.68182914, -0.54025457],
       ...,
       [ 0.07136908, -0.90419856,  0.57232761, ...,  1.28705162,
        -0.68036615, -0.36717014],
       [-0.5330532 , -0.28103332, -0.14693089, ..., -0.85003444,
        -0.47352521, -0.79988121],
       [-0.5330532 , -1.65199685,  0.05857154, ..., -0.85003444,
        -0.82810967, -0.71333899]])
```

```
# Create a Logistic Regression model
from sklearn.linear_model import LogisticRegression
logistic_regression = LogisticRegression()
```

```
# Fit the model to the training data
logistic_regression.fit(x_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Make predictions on the test data
y_pred = logistic_regression.predict(x_test)
```

```
# Evaluate the model
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
accuracy = accuracy_score(y_test, y_pred)
print(accuracy*100,"%")
```

```
73.37662337662337 %
```