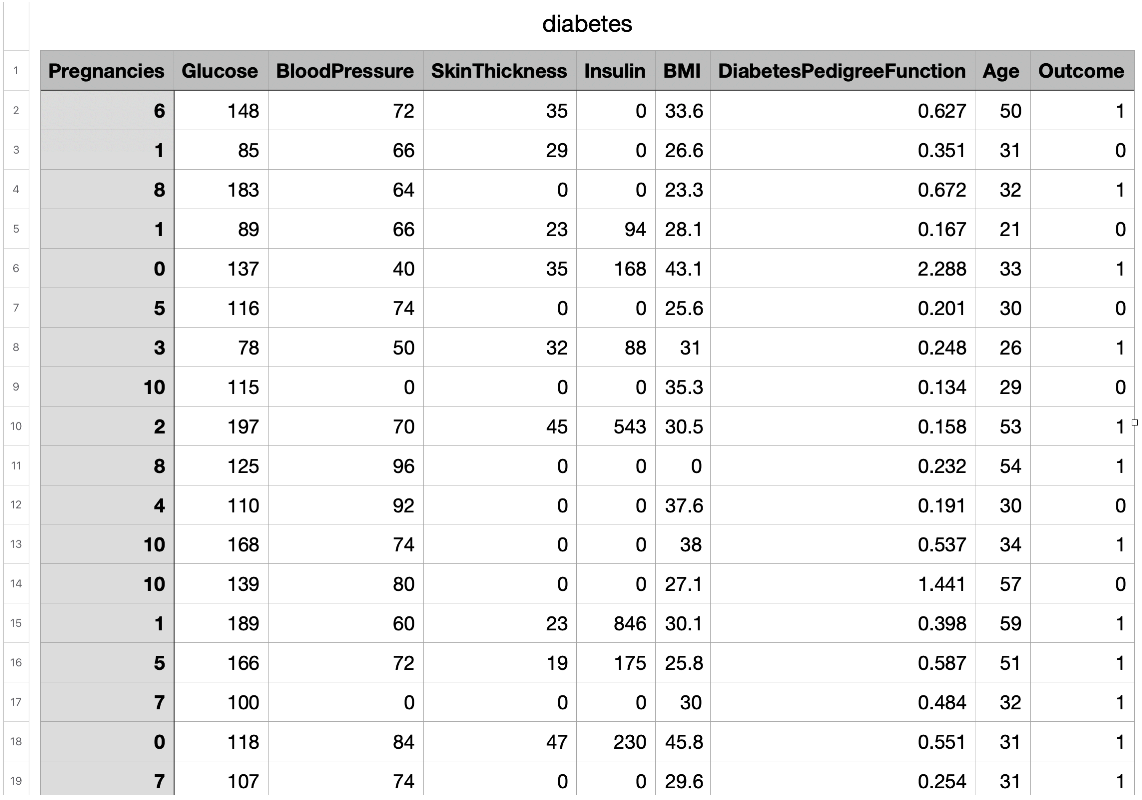
ABOUT DATASET:-



diabetes

1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1
16	5	166	72	19	175	25.8	0.587	51	1
17	7	100	0	0	0	30	0.484	32	1
18	0	118	84	47	230	45.8	0.551	31	1
19	7	107	74	0	0	29.6	0.254	31	1

Dataset contains columns

790 people medical data which is

Input medical data

pregnancies , blood pressure , glucose , insulin , BMI, Diabetes pedigree function , age

Output medical data

Outcome

- **1 - patient have diabetes**
- **0 - patient doesn’t have diabetes**

Decision Tree Classifier in sci-kit learn for class assigment week 10

Load required libraries

In [52]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
import joblib
```

load the dataset (diabeties.csv)

```
having colums
[Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome]
```

In [53]:

```
# Load the dataset
df = pd.read_csv('diabetes.csv')
```

Replace zero values in certain columns with their median

In [54]:

```
columns_to_replace_zero = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']
for column in columns_to_replace_zero:
    df[column] = df[column].replace(0, df[column].median())
```

Split the data into features (X) and target (y)

In [55]:

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

Split the data into training (60%) and testing (40%) sets

```
In [56]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)
```

Standardize the data

```
In [57]: scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(X_train)
         X_test_scaled = scaler.transform(X_test)
```

Train the Decision Tree Classifier

```
In [58]: decision_tree_model = DecisionTreeClassifier(random_state=42)
         decision_tree_model.fit(X_train_scaled, y_train)
```

```
Out[58]: ▾ DecisionTreeClassifier
         DecisionTreeClassifier(random_state=42)
```

Predict on the training set

```
In [59]: y_train_pred_tree = decision_tree_model.predict(X_train_scaled)
```

Predict on the test set

```
In [60]: y_test_pred_tree = decision_tree_model.predict(X_test_scaled)
```

Calculate training accuracy

```
In [61]: training_accuracy_tree = accuracy_score(y_train, y_train_pred_tree)
         print(f"Decision Tree Training Accuracy: {training_accuracy_tree * 100:.2f}%")
```

```
Decision Tree Training Accuracy: 100.00%
```

Calculate testing accuracy

In [62]:

```
testing_accuracy_tree = accuracy_score(y_test, y_test_pred_tree)
print(f"Decision Tree Testing Accuracy: {testing_accuracy_tree * 100:.2f}%")
```

```
Decision Tree Testing Accuracy: 70.45%
```

Print classification report for detailed performance

In [63]:

```
print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_test_pred_tree))
```

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.81      0.73      0.77       206
           1       0.55      0.65      0.59       102

    accuracy                           0.70       308
   macro avg       0.68      0.69      0.68       308
weighted avg       0.72      0.70      0.71       308
```

Save the trained model and scaler

In [64]:

```
joblib.dump(decision_tree_model, 'decision_tree_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

Out[64]:

```
['scaler.pkl']
```

Load the trained model and scaler

In [65]:

```
import pandas as pd
import joblib

loaded_model = joblib.load('decision_tree_model.pkl')
loaded_scaler = joblib.load('scaler.pkl')
```

Define the column names based on the training data Example input data for prediction

```
'Pregnancies': [2],
'Glucose': [130],
'BloodPressure': [80],
'SkinThickness': [30],
'Insulin': [100],
'BMI': [32],
'DiabetesPedigreeFunction': [0.5],
'Age': [45]
```

In [66]:

```
training_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', '

new_data = {
    'Pregnancies': [2], 'Glucose': [130], 'BloodPressure': [80], 'SkinThickness': [30], 'Insulin': [100],
    'BMI': [32], 'DiabetesPedigreeFunction': [0.5], 'Age': [45]
}
new_data_df = pd.DataFrame(new_data, columns=training_columns)
```

Standardize the new data using the loaded scaler

In [67]:

```
new_data_scaled = loaded_scaler.transform(new_data_df)
```

Make predictions

In [68]:

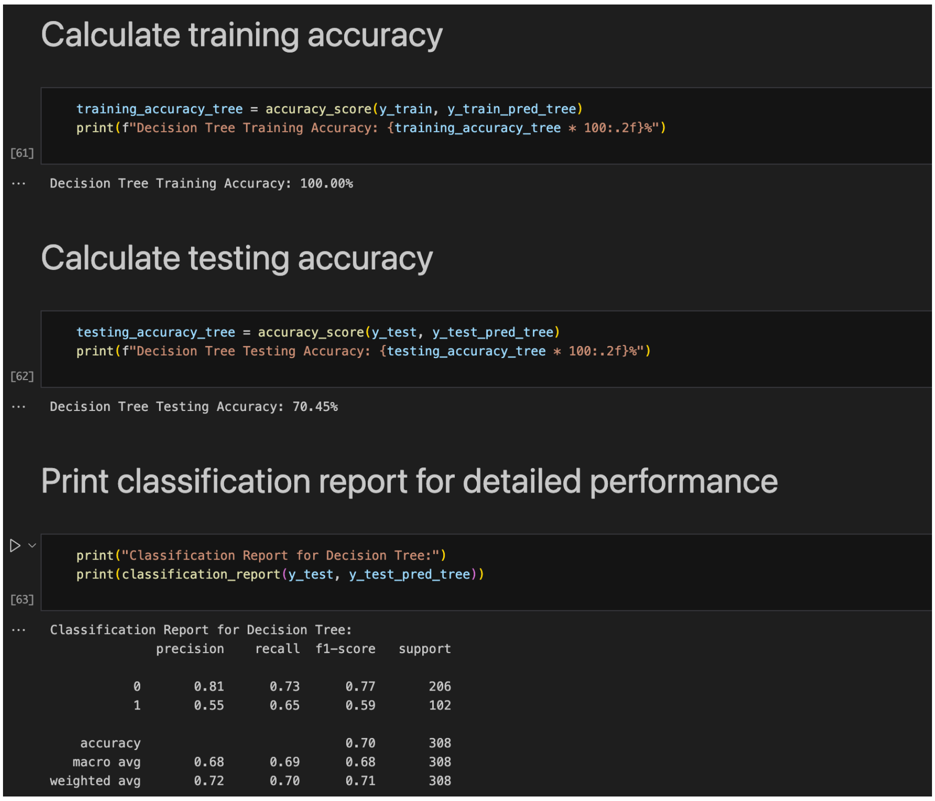
```
predicted_class_encoded = loaded_model.predict(new_data_scaled)

if predicted_class_encoded[0] == 1:
    print("The model predicts that the patient has diabetes.")
else:
    print("The model predicts that the patient does not have diabetes.")
```

```
The model predicts that the patient does not have diabetes.
```

Outcome at 60 / 40 split of dataset

60% - training data set
40% - testing dataset

Calculating Accuracy for training and testing



Calculate training accuracy

```
training_accuracy_tree = accuracy_score(y_train, y_train_pred_tree)
print(f"Decision Tree Training Accuracy: {training_accuracy_tree * 100:.2f}%")
```

[61]

```
Decision Tree Training Accuracy: 100.00%
```

Calculate testing accuracy

```
testing_accuracy_tree = accuracy_score(y_test, y_test_pred_tree)
print(f"Decision Tree Testing Accuracy: {testing_accuracy_tree * 100:.2f}%")
```

[62]

```
Decision Tree Testing Accuracy: 70.45%
```

Print classification report for detailed performance

```
print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_test_pred_tree))
```

[63]

```
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.81      0.73      0.77       206
           1       0.55      0.65      0.59       102

    accuracy                           0.70       308
   macro avg       0.68      0.69      0.68       308
weighted avg       0.72      0.70      0.71       308
```

Decision Tree Training Accuracy: 100.00%
Decision Tree Testing Accuracy: 70.45%

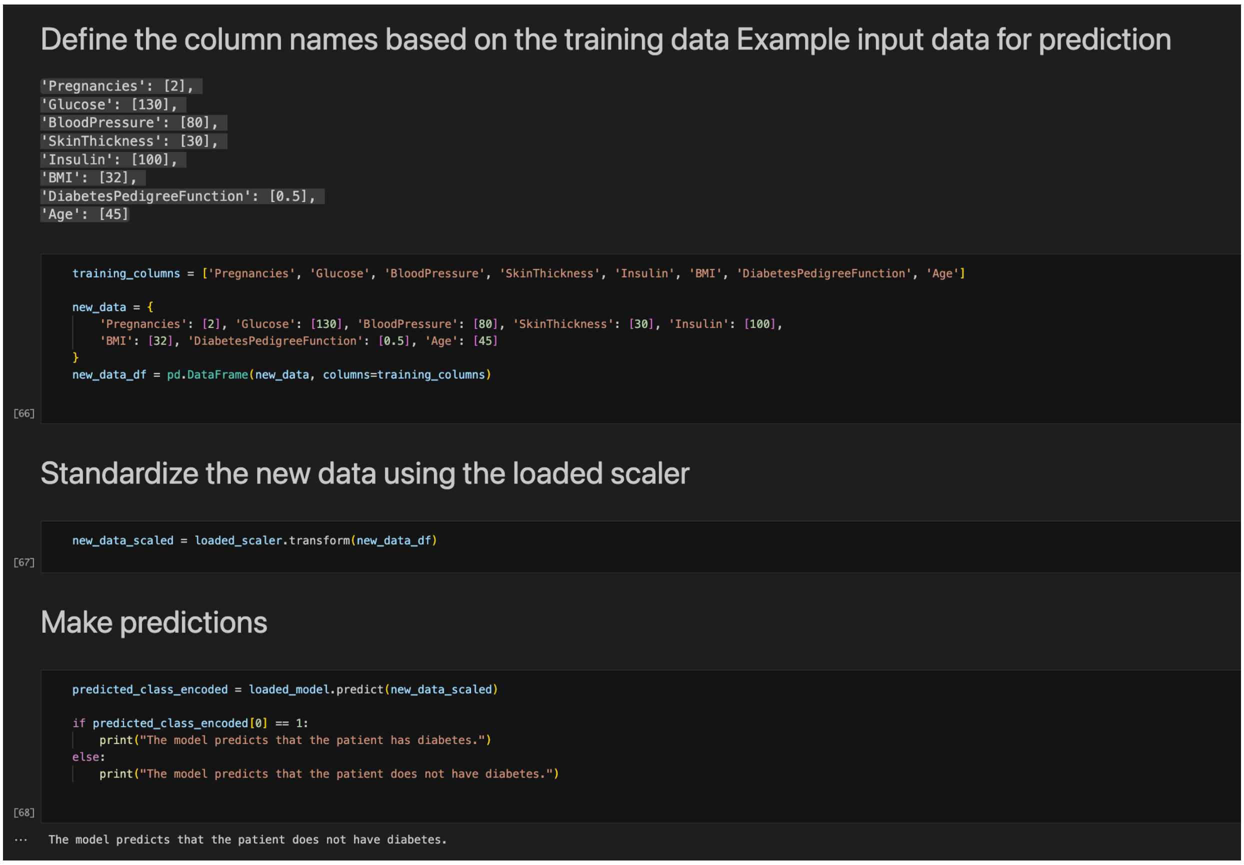
Classification Report for Decision Tree:

```
              precision    recall  f1-score   support

           0       0.81      0.73      0.77       206
           1       0.55      0.65      0.59       102

    accuracy                           0.70       308
   macro avg       0.68      0.69      0.68       308
weighted avg       0.72      0.70      0.71       308
```

TESTING THE MODEL

USING NEW PATIENT DATA TO PREDICT THAT THE PATIENT HAVE DIABETES OR NOT

NEW PATIENT DATA = {‘Pregnancies': [2], 'Glucose': [130], 'BloodPressure': [80],
'SkinThickness': [30], 'Insulin': [100], 'BMI': [32],
'DiabetesPedigreeFunction': [0.5], 'Age': [45] ‘}



Define the column names based on the training data Example input data for prediction

```
'Pregnancies': [2],
'Glucose': [130],
'BloodPressure': [80],
'SkinThickness': [30],
'Insulin': [100],
'BMI': [32],
'DiabetesPedigreeFunction': [0.5],
'Age': [45]
```

```
training_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

new_data = {
    'Pregnancies': [2], 'Glucose': [130], 'BloodPressure': [80], 'SkinThickness': [30], 'Insulin': [100],
    'BMI': [32], 'DiabetesPedigreeFunction': [0.5], 'Age': [45]
}
new_data_df = pd.DataFrame(new_data, columns=training_columns)
```

[66]

Standardize the new data using the loaded scaler

```
new_data_scaled = loaded_scaler.transform(new_data_df)
```

[67]

Make predictions

```
predicted_class_encoded = loaded_model.predict(new_data_scaled)

if predicted_class_encoded[0] == 1:
    print("The model predicts that the patient has diabetes.")
else:
    print("The model predicts that the patient does not have diabetes.")
```

[68]

```
... The model predicts that the patient does not have diabetes.
```

OUTPUT

MODEL PREDICTS THAT THE PATIENT DOES NOT HAVE DIABETES

INFERENCE

- **Precision:** The proportion of positive identifications that were actually correct. For instance, a precision of 0.76 for class 0 means that 76% of the instances predicted as class 0 are actually class 0.
- **Recall:** The proportion of actual positives that were correctly identified. For instance, a recall of 0.87 for class 0 means that 87% of the actual class 0 instances were correctly identified.
- **F1-Score:** The harmonic mean of precision and recall. It balances precision and recall, providing a single metric that combines both.
- **Support:** The number of true instances for each class in the dataset.

Report:-

High Training Accuracy with Lower Testing Accuracy