AI LABS

Diabetes Prediction model using Decision Tree

Table of Contents

Overview

This code is designed to train a decision tree model to predict whether a patient has diabetes or not, based on certain factors. The code takes in a dataset called "diabetes.csv", which contains information about a number of patients, and uses it to train the decision tree model. Once the model is trained, the code allows the user to input data for a new patient and receive a prediction on whether or not that patient is likely to have diabetes.

Before you start (Pre-requisites)

Have a Gmail account to use google Colab (easy to use, with all the necessary python libraries pre-installed), alternatively you can use different IDE's (e.g. Visual studio code, jupyter notebook, pycharm, etc). But make sure you have Python installed on your computer and all the required library.

Data

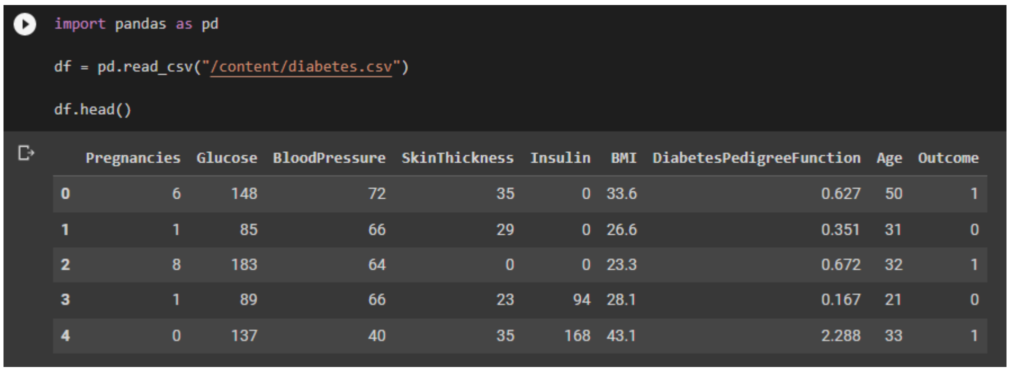
You can download the dataset from here:

Make sure you give the correct path for the dataset in the code.

Tasks

1. Importing libraries and Reading in the dataset:

The first line of code imports the "pandas" library, which is used for data analysis and manipulation in Python. It is commonly used to read and write data in various file formats, such as CSV, Excel, and SQL databases.
The second line of code reads in a CSV file called "diabetes.csv" and creates a Pandas dataframe called "df" to hold the data. This file contains information about a number of patients, including their age, BMI, blood pressure, and other health-related factors.

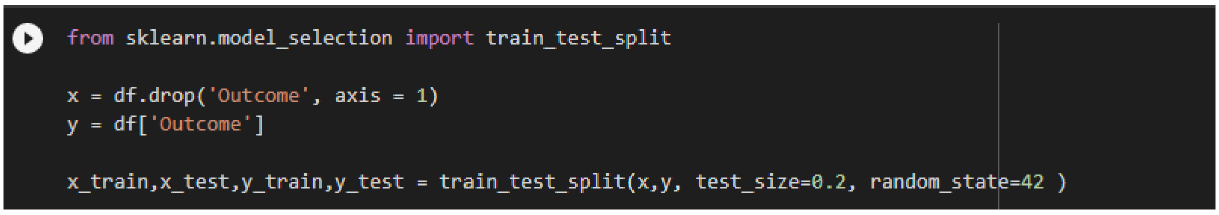


```python
import pandas as pd

df = pd.read_csv("/content/diabetes.csv")

df.head()
```

	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

2. Splitting the data into training and testing sets:

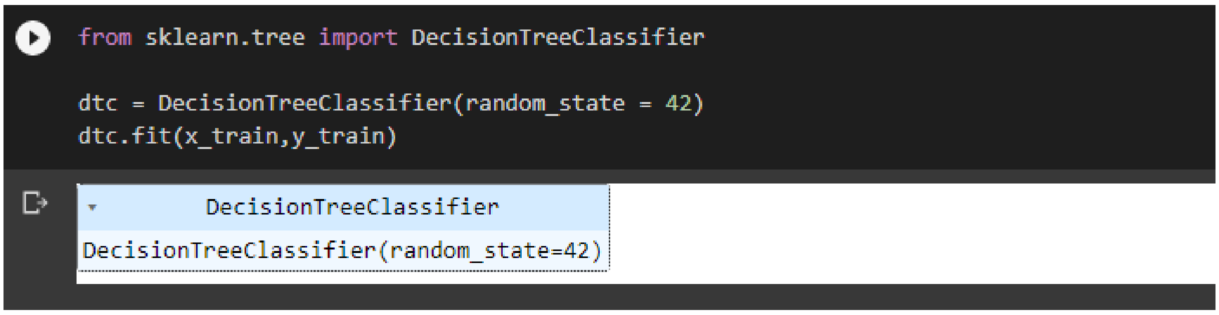
The third line of code imports a function called "train_test_split" from the "sklearn.model_selection" library, which is used to split the data into training and testing sets. The "x" variable contains all of the features (i.e. the factors that the model will use to make predictions), while the "y" variable contains the target variable (i.e. the label that the model is trying to predict, which in this case is whether or not a patient has diabetes). The data is split into 80% for training and 20% for testing, and a random seed (i.e. "random_state") is set to ensure that the same split is used each time the code is run.



```python
from sklearn.model_selection import train_test_split

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

x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=42 )
```

3. Training the model:

The fourth line of code imports a decision tree classifier from the "sklearn.tree" library, which is a type of machine learning model that makes decisions based on a set of rules. The "random_state" parameter is set to ensure that the same results are obtained each time the code is run. The model is trained on the training data using the "fit" method.

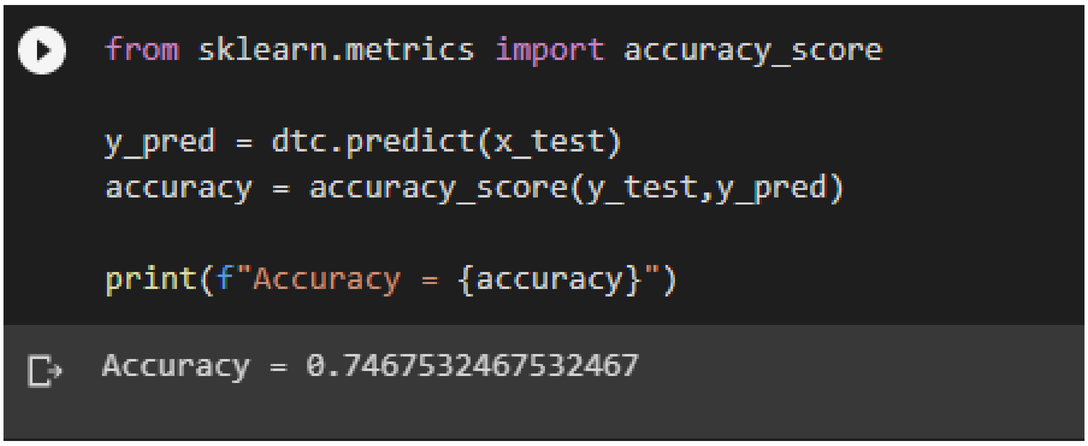


```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(random_state = 42)
dtc.fit(x_train,y_train)
```

```
          DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

4. Evaluating the model:

   The fifth line of code imports a function called "accuracy_score" from the "sklearn.metrics" library, which is used to evaluate the performance of the model on the test data. The "y_pred" variable contains the predictions made by the model on the test data, while the "y_test" variable contains the actual labels for the test data. The accuracy of the model is calculated by comparing the predicted labels to the actual labels using the "accuracy_score" function, and the result is printed to the console.

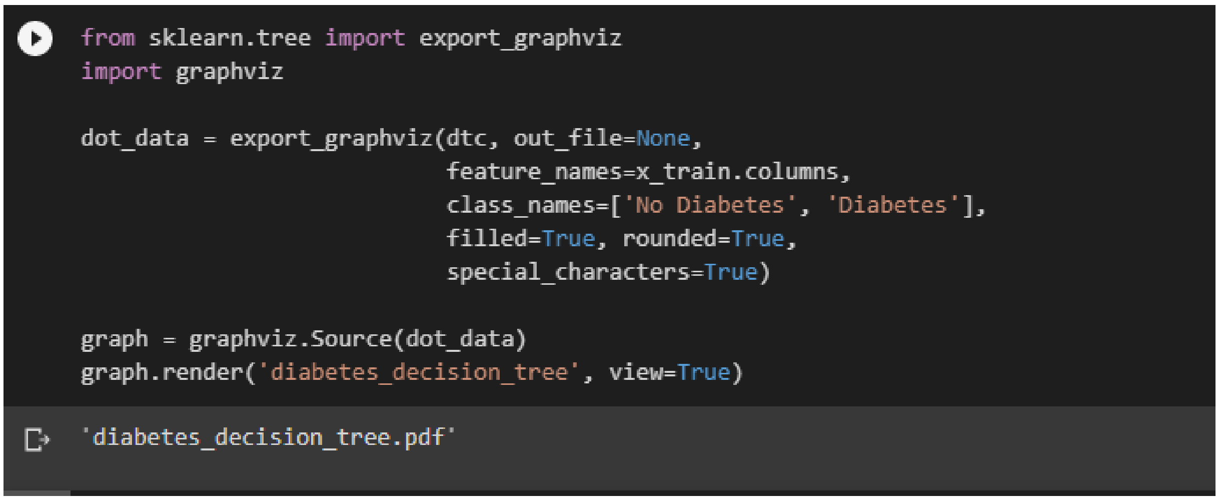


```
from sklearn.metrics import accuracy_score

y_pred = dtc.predict(x_test)
accuracy = accuracy_score(y_test,y_pred)

print(f"Accuracy = {accuracy}")
```

```
Accuracy = 0.7467532467532467
```

5. Visualizing the decision tree:

The sixth line of code imports two functions from the "sklearn.tree" and "graphviz" libraries, respectively, which are used to visualize the decision tree model that was trained in step 4. The "dot_data" variable contains a string that represents the decision tree in a Graphviz-compatible format. This string is used to create a "Source" object, which is then rendered as a PDF file called "diabetes_decision_tree.pdf" that can be viewed in a PDF viewer.

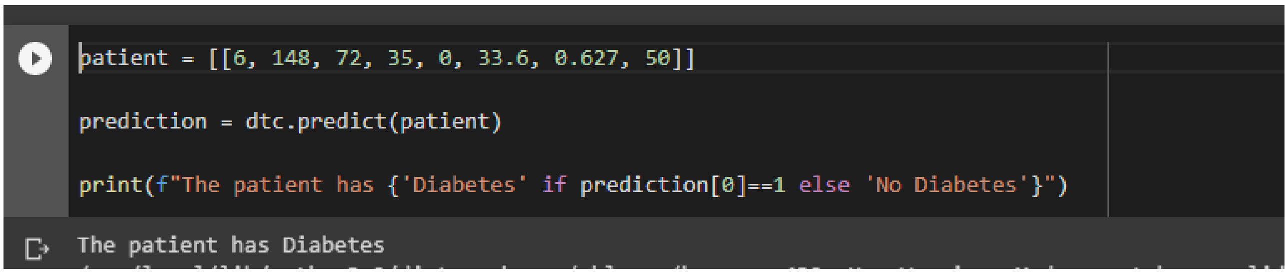


```
from sklearn.tree import export_graphviz
import graphviz

dot_data = export_graphviz(dtc, out_file=None,
                           feature_names=x_train.columns,
                           class_names=['No Diabetes', 'Diabetes'],
                           filled=True, rounded=True,
                           special_characters=True)

graph = graphviz.Source(dot_data)
graph.render('diabetes_decision_tree', view=True)
```

```
'diabetes_decision_tree.pdf'
```

6. Making predictions:

The seventh line of code creates a list called "patient" that contains the values of the different features for a new patient. These values are used to make a prediction on whether or not the patient has diabetes using the decision tree model that was trained in step 4. The prediction is printed to the console.



```
patient = [[6, 148, 72, 35, 0, 33.6, 0.627, 50]]

prediction = dtc.predict(patient)

print(f"The patient has {'Diabetes' if prediction[0]==1 else 'No Diabetes'}")
```

```
The patient has Diabetes
[illegible]
```

Conclusion

The code tasks above demonstrates how to train a decision tree classifier to predict whether or not a patient has diabetes based on their age, BMI, blood pressure, and other health-related factors. The model is trained on a dataset of patients with known outcomes, and then evaluated on a separate set of test data to measure its accuracy. The decision tree is visualised using Graphviz and the model is used to make prediction on a new data. This example lab highlights the power of machine Learning in the field of healthcare and shows how it can be used to improve patient outcomes by identifying potential health issues early on.