Kingdom of Saudi Arabia
King Saud University
College of Computer & Information Sciences
Department of Computer Science

CSC361

Assignment #3

Machine Learning

Predicting Diabetes using Decision Tree Classifier

Students name

Amal Abdulrahman Almuarik

ID

442200454

Section Number

44095

Course Name

Artificial Intelligence

Instructor

Dr.Nouf Alshunaifi.

Monday, 6 February 2023

Contents

Introduction

1.1 Problem Statement

Diabetes is one of the deadliest diseases in the world. It is not only a disease but also creator of different kinds of diseases like heart attack, blindness etc. The normal identifying process is that patients need to visit a diagnostic center, consult their doctor, and sit tight for a day or more to get their reports.
So, the objective is to predict whether a patient has diabetes or not. based on certain diagnostic measurements included in the dataset provided in Kaggle.

1.2 Methodology

In machine Learning there are various classifiers can be used to predict diabetes of patients with different accuracy rates. However , as we have to classify the data into patient having diabetes or not the classifier used in here is the Decision Tree classifier. Along with the confusion matrix to calculate the accuracy of this model.

1.3 The DateSet

The diabetes data set was originated from https://www.kaggle.com/johndasilva/diabetes. Diabetes dataset containing 2000 cases. The main objective is to predict based on the measures to whether the patient is diabetic or not.

	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

```
<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
```

- The diabetes data set consists of 2000 data points, with 9 features each.
- “Outcome” is the feature we are going to predict, 0 means No diabetes, 1 means diabetes.
- There is no null values in the data set.

The Tools And Libraries

Programming language used: python.

The Packages used : Python-Scikit Learn

Libraries & modules used :

1- **pandas** : an open-source data analysis and data manipulation library. It is used to load and manipulate data.

2- **sklearn** : it contains a lot of efficient tools for machine learning and statistical modeling including classification , regression ,…,etc

3- **pydotplus** : It is a python module that provides an interface to Graphviz's Dot language. It is used to visualize Decision Trees.

4- **metrics:** It is a module from sklearn that provides various performance metrics like accuracy, recall, precision, etc.

3- **externals.six** : a module from sklearn that is used to visualize decision trees

Functions :

1- **train_test_split**: It is a function from sklearn.model_selection that is used to split the dataset into training and testing datasets.

2- **confusion_matrix:** It is a function from sklearn.metrics that is used to calculate the confusion matrix of the model's performance.

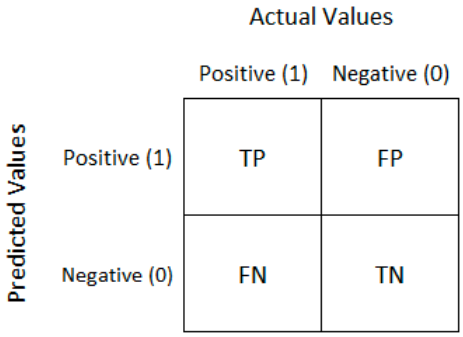
All of These libraries , modules , function work together to build a Decision Tree Classifier model and evaluate its performance.

Experimental result

The Evaluation is the final step of prediction model. Here, the evaluation of the prediction results using evaluated using metrics like confusion matrix and f1-score.

Confusion Matrix- It gives us gives us a matrix as output and describes the complete performance of the model. Where,

TP: True Positive **FP:** False Positive
FN: False Negative **TN:** True Negative



Accuracy for the matrix can also be calculated by taking average of the values lying across the main diagonal. It is given as-

$$Accuracy = \frac{TP + FT}{N}$$

Where, **N :** Total number of samples

Precision: It is the number of correct positive results divided by the number of positive results predicted by the classifier. It is expressed as-

$$Precision = \frac{TP}{(TP + FP)}$$

Recall: It is the number of correct positive results divided by the number of all relevant samples. In mathematical form it is given as-

$$Recall = \frac{TP}{(TP + FN)}$$

F1 score - It is used to measure a test's accuracy. F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells how precise your classifier is as well as how robust it is. Mathematically, it is given as-

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

F1 Score tries to find the balance between precision and recall.

The Confusion Matrix for the decision Tree given below :

	Diabetic	Non-Diabetic
Diabetic	124	22
Non-Diabetic	31	54

THE RESULTS :

Classification report before applying the entropy filter :

- Previous model Accuracy score : 70.99567

```
Evaluation using Classification report:
              precision    recall  f1-score   support

           0       0.75      0.80      0.78       146
           1       0.62      0.55      0.58        85

    accuracy                           0.71       231
   macro avg       0.69      0.68      0.68       231
weighted avg       0.70      0.71      0.71       231
```

Classification report After Applying the entropy filter:

- model Accuracy after applying entropy filter : 77.0562

```
Evaluation using Classification report:
              precision    recall  f1-score   support

           0       0.80      0.85      0.82       146
           1       0.71      0.64      0.67        85

    accuracy                           0.77       231
   macro avg       0.76      0.74      0.75       231
weighted avg       0.77      0.77      0.77       231
```

After applying Machine Learning Algorithm on the dataset . The accuracy raised to 77.05 % which is better than the previous model that have accuracy rate 69.26%

Implementation Detail

Predecting Diabetes Using Decision Trees

In [58]:
```
#import pandas library
import pandas as pd

# Import Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier


# Import train_test_split function
from sklearn.model_selection import train_test_split

# import confusion matrix
from sklearn.metrics import confusion_matrix

#Import scikit-learn metrics module for accuracy calculation , recall , precision
from sklearn import metrics

#import modules for sklearn.external
import six
import sys
sys.modules['sklearn.externals.six'] = six

#Import modules for Visualizing Decision trees
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

Load the data

In [59…
```
data_frame = pd.read_csv("diabetes.csv")
```

In [60…
```
data_frame.head()
```

Out[60]:

	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 [61…
```
data_frame.tail()
```

Out[61]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

In [62…
```
#feature variables
feature_variable = data_frame.drop(['Outcome'], axis=1)
feature_variable
```

Out[62]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 8 columns

```
#target variable
target_variable
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

```
data_frame.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
```

Training and evaluating on the training dataset

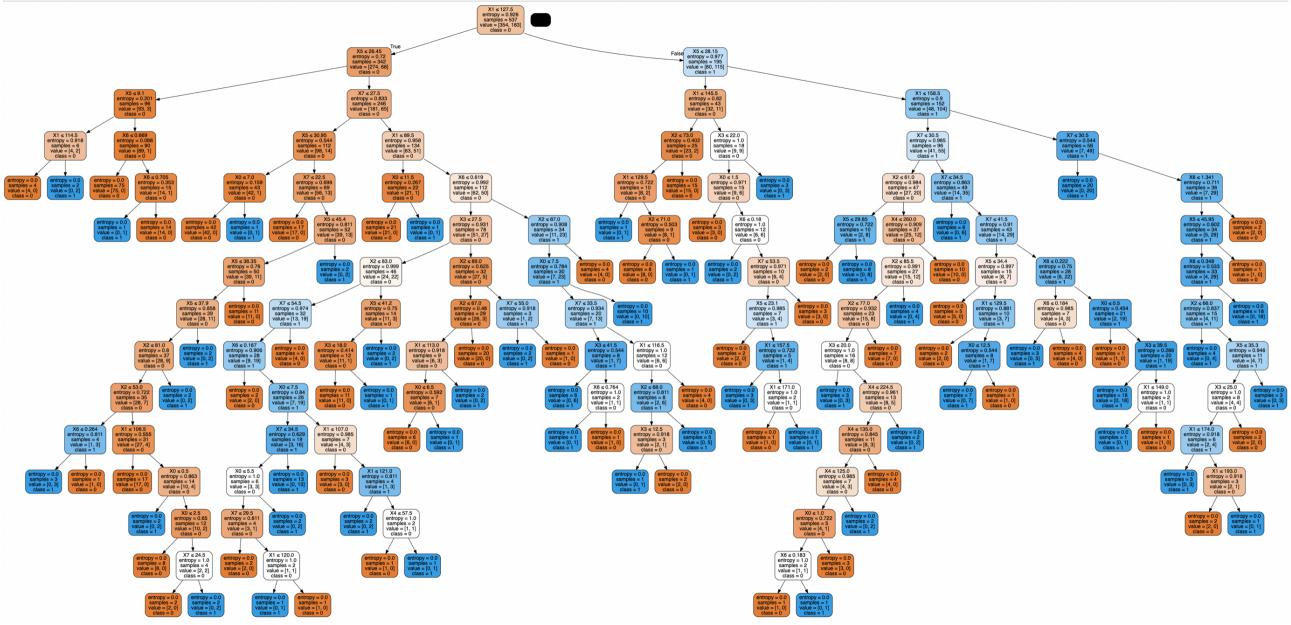
```
x_train, x_test, y_train, y_test = train_test_split(feature_variable,
                                                    target_variable,
                                                    test_size=0.3,
                                                    random_state=1)
```

```
# Create Decision Tree classifer object
model = DecisionTreeClassifier()

# Train Decision Tree Classifer
model = model.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)
```

```
#Evaluation using Accuracy score
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 70.995670995671
```

```
dot_data = StringIO()
export_graphviz(model,
                out_file = dot_data,
                filled=True, rounded=True,
                special_characters=True,
                feature_names = None,
                class_names=['0','1']
                )
graph = pydotplus.graph_from_dot_data(
        dot_data.getvalue()
)
graph.write_png('diabetes.png')
display(Image(graph.create_png()))
```



```
# Create Decision Tree classifer object
model = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
model = model.fit(x_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(x_test)

# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Accuracy: 77.05627705627705
```

Evaluation metrics

Confusion matrix

```
confusion_matrix = metrics.confusion_matrix(
    y_test,
    y_pred
)
confusion_matrix
```

```
0]: array([[124,  22],
           [ 31,  54]])
```

Precision

```
precision = metrics.precision_score(
    y_test,
    y_pred,
    average="macro"
)
print("Evaluation using Precision:",precision)
```

```
Evaluation using Precision: 0.7552631578947369
```

Recall

```
recall = metrics.recall_score(y_test,
                              y_pred,
                              average="macro")
print("Evaluation using Recall:",recall)
```

```
Evaluation using Recall: 0.7423045930701047
```

```
F1 = metrics.f1_score(y_test,
                      y_pred,
                      average="macro")
print("Evaluation using F1 Score :",F1)
```

```
Evaluation using F1 Score : 0.74736385959844
```

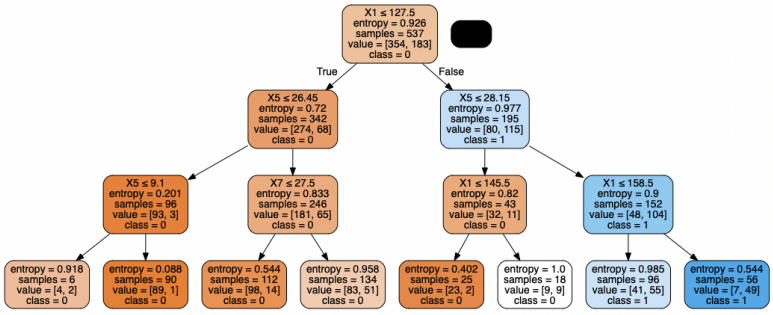
```
#Evaluation using Classification report
report = metrics.classification_report(
       y_test,
       y_pred
)
print("Evaluation using Classification report:\n",report)
```

```
Evaluation using Classification report:
              precision    recall  f1-score   support

           0       0.80      0.85      0.82       146
           1       0.71      0.64      0.67        85

    accuracy                           0.77       231
   macro avg       0.76      0.74      0.75       231
weighted avg       0.77      0.77      0.77       231
```

```
#checking prediction value
model.predict([[6,148,72,35,0,33.6,0.627,50]])
```

```
/Users/amalalmuarik/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but I
  warnings.warn(
5]: array([1])
```

```
dot_data = StringIO()
export_graphviz(model,
                out_file = dot_data,
                filled=True, rounded=True,
                special_characters=True,
                feature_names = None,
                class_names=['0','1']
                )
graph = pydotplus.graph_from_dot_data(
        dot_data.getvalue()
)
graph.write_png('diabetes.png')
display(Image(graph.create_png()))
```




```
feature = feature_variable.columns
feature
```

```
5]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
           'BMI', 'DiabetesPedigreeFunction', 'Age'],
          dtype='object')
```

The classification rate increased to 77.05%, which is better accuracy than the previous model.