# Untitled109

July 2, 2023

## 1 Module 8: Random Forest

Assignment Contact us: support@intellipaat.com / © Copyright Intellipaat / All rights reserved Intel iPaat Python for Data Science Certification Course Problem Statement: You work in XYZ Company as a Python Data Scientist. The company officials have collected some data on diabetes based on years of experience and wish for you to create a model from it. Dataset: diabetes.csv Tasks To Be Performed: 1. Load the dataset using pandas 2. Extract data from outcome column is a variable named Y 3. Extract data from every column except outcome column in a variable named X 4. Divide the dataset into two parts for training and testing in 70% and 30% proportion 5. Create and train Random Forest Model on training set 6. Make predictions based on the testing set using the trained model 7. Check the performance by calculating the confusion matrix and accuracy score of the model

[1]:
```
## import the required library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
##Load the dataset using pandas
data=pd.read_csv(r'C:/Users/Vikas/Desktop/diabetes-1.csv')
```

[6]:
```
data.head(5)
```

[6]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | \ |
|---|---|---|---|---|---|---|---|
| 0 | 6 | 148 | 72 | 35 | 0 | 33.6 | |
| 1 | 1 | 85 | 66 | 29 | 0 | 26.6 | |
| 2 | 8 | 183 | 64 | 0 | 0 | 23.3 | |
| 3 | 1 | 89 | 66 | 23 | 94 | 28.1 | |
| 4 | 0 | 137 | 40 | 35 | 168 | 43.1 | |

| | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|
| 0 | 0.627 | 50 | 1 |
| 1 | 0.351 | 31 | 0 |
| 2 | 0.672 | 32 | 1 |
| 3 | 0.167 | 21 | 0 |
| 4 | 2.288 | 33 | 1 |

[7]:
```
data.shape
```

[7]:
```
(768, 9)
```

[10]:
```
data.columns
```

[10]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[11]:
```
data.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
```

[13]:
```
data.describe().T
```

[13]:

| | count | mean | std | min | 25% | \ |
|---|---|---|---|---|---|---|
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | |
| Glucose | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | |
| BloodPressure | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | |
| SkinThickness | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | |
| Insulin | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | |
| BMI | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | |

| | 50% | 75% | max |
|---|---|---|---|
| Pregnancies | 3.0000 | 6.00000 | 17.00 |
| Glucose | 117.0000 | 140.25000 | 199.00 |
| BloodPressure | 72.0000 | 80.00000 | 122.00 |
| SkinThickness | 23.0000 | 32.00000 | 99.00 |
| Insulin | 30.5000 | 127.25000 | 846.00 |

```
BMI                         32.0000   36.60000   67.10
DiabetesPedigreeFunction     0.3725    0.62625    2.42
Age                         29.0000   41.00000   81.00
Outcome                      0.0000    1.00000    1.00
```

[14]:
```
data.isnull().sum()
```

[14]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[15]:
```
##Extract data from outcome column is a variable named Y
y=pd.DataFrame(data.iloc[:,-1])
```

[17]:
```
y
```

[17]:
```
     Outcome
0          1
1          0
2          1
3          0
4          1
..       …
763        0
764        0
765        0
766        1
767        0

[768 rows x 1 columns]
```

[16]:
```
##Extract data from every column except outcome column in a variable named X
x=pd.DataFrame(data.iloc[:,:-1])
```

[18]:
```
x
```

[18]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
```

```
4              0      137             40             35      168  43.1
..           …      …               …              …       …    …
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        …  …
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

[19]:
```
##Divide the dataset into two parts for training and testing in 70% and 30%␣
 ↪proportion
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(
x, y, test_size=0.30, random_state=0)
```

[23]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf
```

[23]:
```
RandomForestClassifier()
```

[26]:
```
## best fit the modul
rf.fit(x_train,y_train)
```

```
C:\Users\Vikas\AppData\Local\Temp\ipykernel_18324\2915513246.py:2:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  rf.fit(x_train,y_train)
```

[26]:
```
RandomForestClassifier()
```

[27]:
```
##Make predictions based on the testing set using the trained model
y_pred=rf.predict(x_test)
```

```
y_pred
```

[27]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1], dtype=int64)
```

[28]:
```
## confusion matrixc
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print('confusionmatrix:\n',cm)
```

```
confusionmatrix:
 [[137  20]
 [ 32  42]]
```

[29]:
```
##accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:",accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.7748917748917749
```

[ ]:
```

```