CURNEU MEDTECH INNOVATIONS PRIVATE LIMITED

Nikinprasad V R

1832038

Problem Statement:

Diabetes is considered as one of the deadliest and chronic diseases which causes an increase in blood sugar. Many complications occur if diabetes remains untreated and unidentified. But the rise in machine learning approaches solves this critical problem. Here we have a problem to classify the patients who have diabetes with the given glucose and blood pressure values. So, we implement the K Nearest Neighbours classifier to give us the classified output with its accuracy. KNN algorithm is one of the simplest classification algorithms and it is one of the most used learning algorithms. The main aim is to analyse and predict whether the respective patient has diabetes or not based on several factors.

About the Dataset:

The dataset contains the real time data of the diabetic patients with the corresponding factors that indicates that the patient has diabetes. This data set contains 9 columns "Pregnancies", "Glucose", "Blood Pressure", "Skin Thickness", "Insulin", "BMI", "Diabetes Pedigree Function", "Age" and "Outcome". Containing these features, the "Outcome" feature is the target variable which tells us whether the patient has diabetes or not. "0" corresponds to the decision that the patient does not have diabetes and "1" represents that the patient has diabetes.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1

Diabetes Database

Exploratory Data Analysis:

In order to find whether the person is affected with diabetes, we are using the diabetes dataset. We use different types of graphs and finally come into a conclusion.

In this problem we explore our data to analyse the patterns among the features so as to get the full knowledge about the data.

According to the dataset given for the problem we do exploratory data analysis on only one feature "Outcome" since the main feature is the target variable that we focus on.

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import
confusion_matrix,accuracy_score,precision_score,classification_repor
t
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

The data is imported and is ready to use.

Code:

```
data=pd.read_csv("D:\#PROJECTS\Curneu\SD03Q016\Diabetes
Database.csv")
print('\n')
print("Description of the dataset")
print(data.describe())
print(data.info())
```

```
print(data.isnull().sum())

print('\n')

print("Finding the Square Root of number of obervations in the
dataset") #Thumb rule

print(data.shape)

print("sqrt", np.sqrt(769))
```

Output:

```
Description of the dataset
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
<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)
Finding the Square Root of number of obervations in the dataset
(768, 9)
sqrt 27.730849247724095
```

Diabetes positive Outcome Frequency Distribution:

Code:

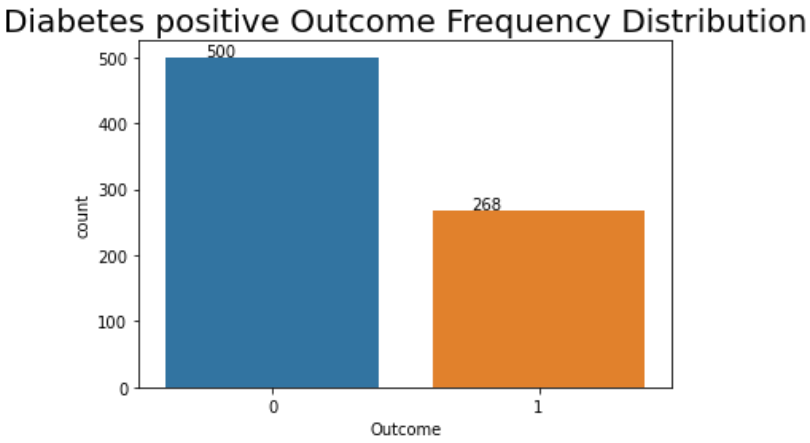
```
ax = sns.countplot(data=data, x ='Outcome')

ax.set_title('Diabetes positive Outcome Frequency Distribution',
fontsize=20)

for p in ax.patches:

        ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.15,
p.get_height()+1))
```

Output:



Diabetes positive vs Pregnancy Frequency Distribution:

Code:

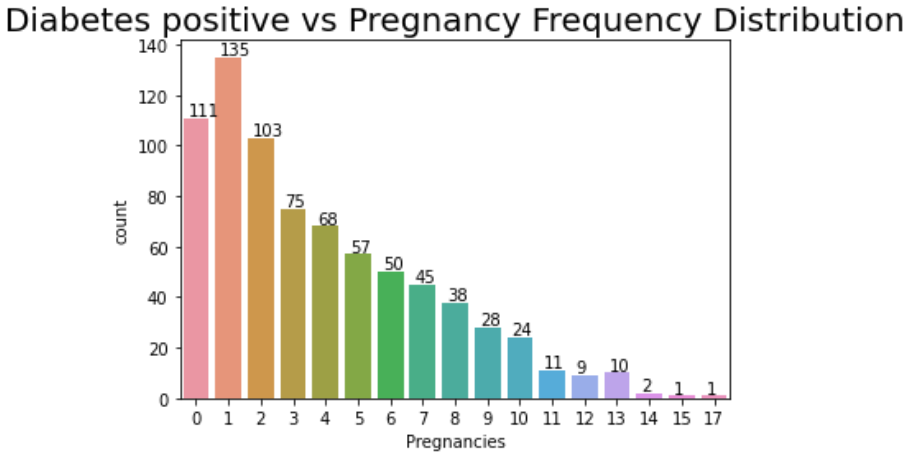
```
ax = sns.countplot(data=data, x ='Pregnancies')

ax.set_title('Diabetes positive vs Pregnancy Frequency Distribution',
fontsize=20)

for p in ax.patches:

        ax.annotate('{:}'.format(p.get_height()),    (p.get_x()+0.15,
p.get_height()+1))
```

Output:



Diabetes positive vs Pregnancy Frequency Distribution:

Code:

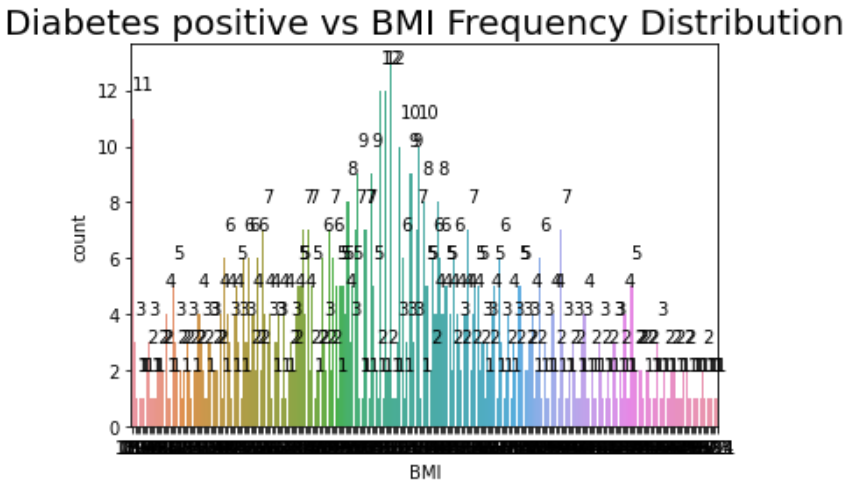
```
ax = sns.countplot(data=data, x ='BMI')

ax.set_title('Diabetes positive vs BMI Frequency Distribution', fontsize=20)

for p in ax.patches:

        ax.annotate('{:}'.format(p.get_height()),    (p.get_x()+0.15, p.get_height()+1))
```

Output:



Splitting, Scaling and Model fitting:

Code:

```
#--SPLITTING THE DATASET--#
x=data.iloc[:,:-1].values
y=data.iloc[:,-1].values
```

```
x_train,x_test,y_train,y_test= train_test_split(x,y, test_size=0.1,
random_state=0)

#--SCALLING--#
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.fit_transform(x_test)
X_train = pd.DataFrame(x_train)

#--FITTING THE MODEL--#
knn= KNeighborsClassifier(n_neighbors=27)
knn.fit(x_train,y_train)
y_test_pred=knn.predict(x_test)
y_test_values=y_test
print("Accuracy of the model is :", knn.score(x_test,y_test)*100)
y_pred = knn.predict(x_test)
print(confusion_matrix(y_test,y_pred))
classification = classification_report(y_test,y_pred)
print("CALSSIFICATION REPORT OF KNN \n",classification)
print('\n')
```

Output:

```
Accuracy of the model is : 84.4155844155844
[[47  4]
 [ 8 18]]
CALSSIFICATION REPORT OF KNN
              precision    recall  f1-score   support

           0       0.85      0.92      0.89        51
           1       0.82      0.69      0.75        26

    accuracy                           0.84        77
   macro avg       0.84      0.81      0.82        77
weighted avg       0.84      0.84      0.84        77
```

Test Case:
Code:

```
#--PREDICTING THE VALUES--#
print("PREDICTING VALUES")
y_predict=pd.DataFrame(data=[y_test_pred,y_test_values])
print(knn.predict([[6,148,72,35,0,33,1,50]]))
```

Output:

```
PREDICTING VALUES
[1]
```

Conclusion:

From the model and the plots above, we find that maximum accuracy is obtained. Here we have an accuracy of 84.41% which proves that the model has worked efficiently. By giving input variables, we found that the patient is diabetic.