DiabetesPrediction

February 25, 2024

1 DIABETES PATIENTS

We are using the dataset from National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset.

Importing Required Libraries

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

Importing Important Libraries For Prediction

Train Test Split: Technique for splitting data into training and testing sets to assess model performance.

Logistic Regression: Method for predicting the probability of a binary outcome using the logistic function.

Accuracy: Metric measuring the proportion of correctly classified instances in a classification model.

Sklearn: Python's Scikit-learn, a powerful machine learning library providing tools for data analysis and model building.

[2]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[3]:
```
#Loading the dataset
ds = pd.read_csv(r'C:\Users\Bikash␣
 ↪shah\Downloads\MeriSkill\Project_Second\diabetes.csv')
```

[4]:
```
ds.head()
```

[4]:

	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
```

[5]:
```
ds.shape
```

[5]:
```
(768, 9)
```

[6]:
```
#looking if data type is correct
ds.dtypes
```

[6]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

[7]:
```
ds.describe()
```

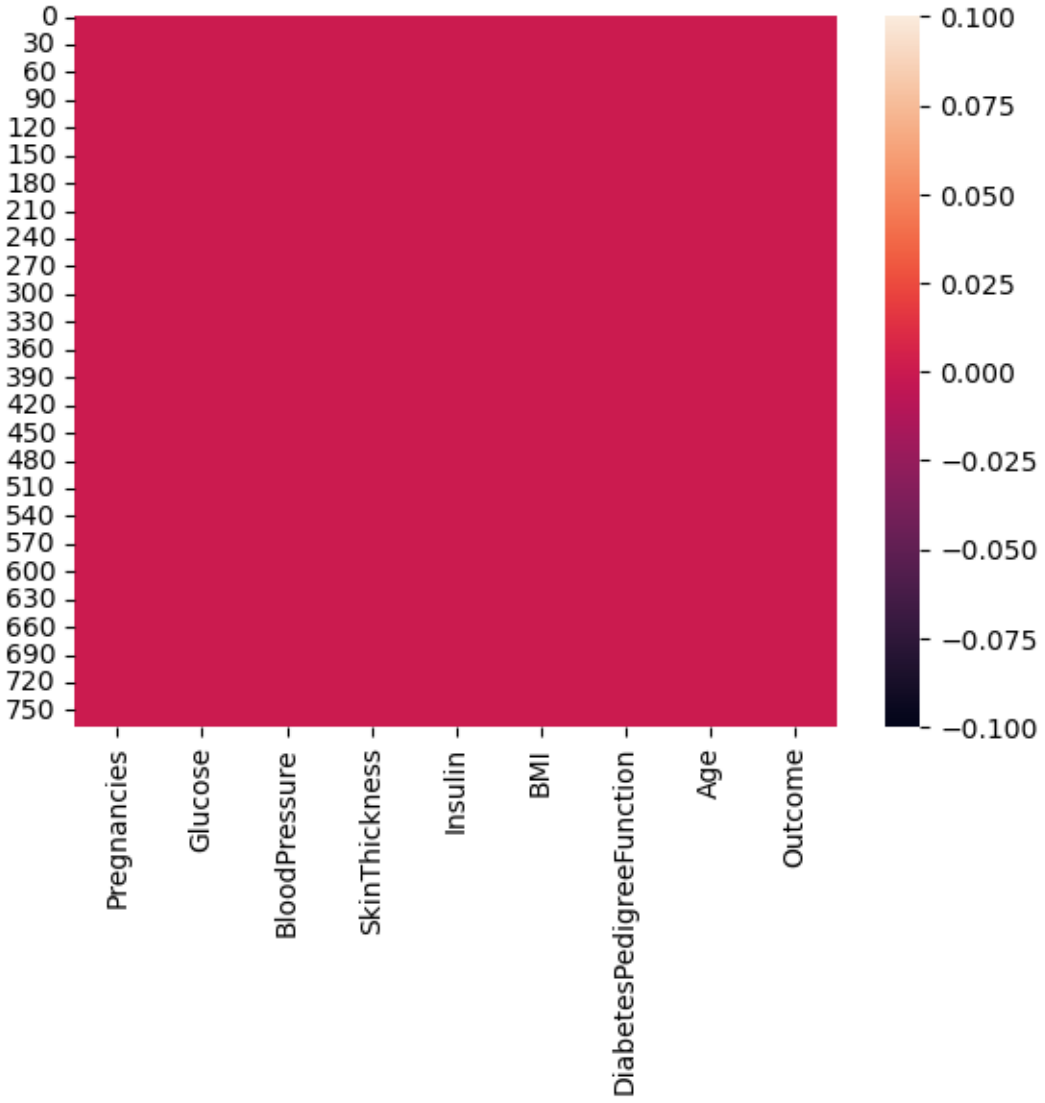
[7]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
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
```

[8]:
```
#looking if there is any null/missing value
sns.heatmap(ds.isnull())
```

[8]:
```
<Axes: >
```



[9]:
```
#Calculating correlation between each and every data (Correlation Matrix)
correlation = ds.corr()
print(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
```

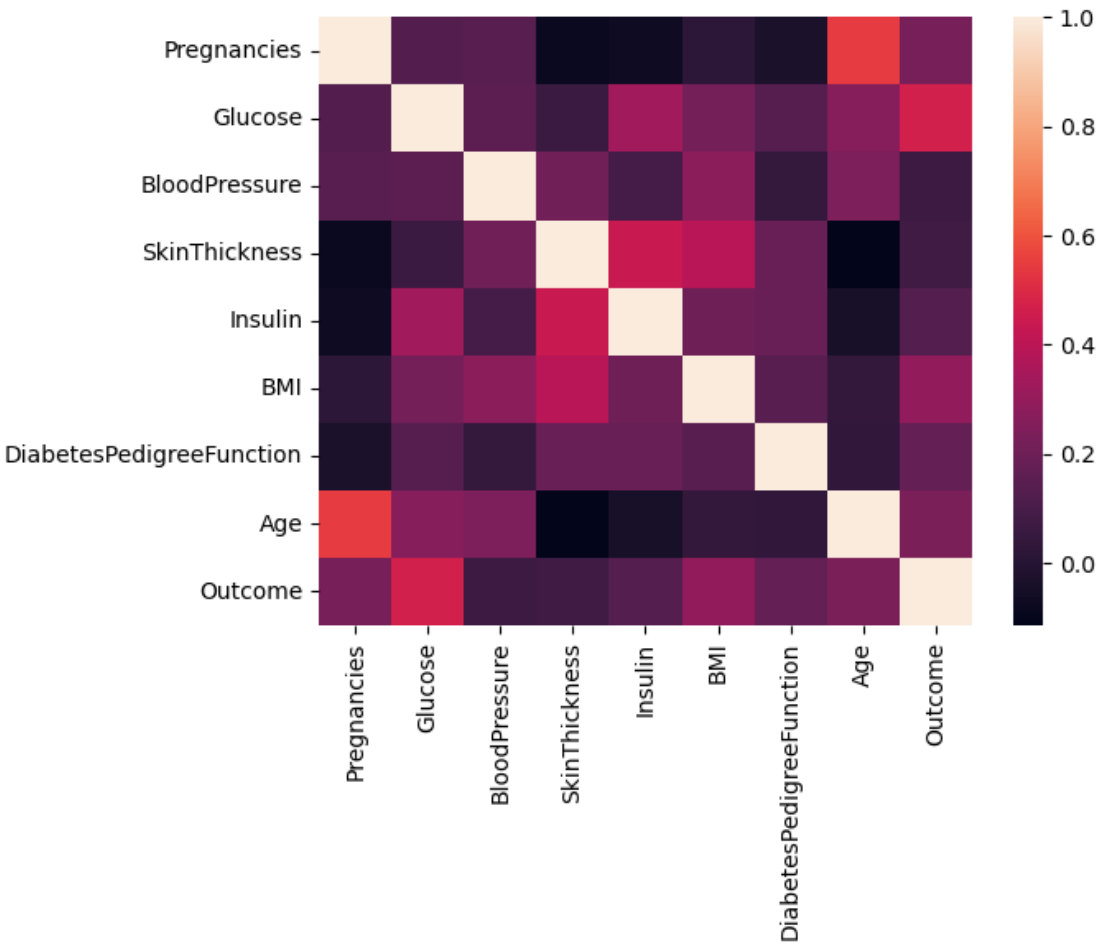
```
Glucose                       0.129459  1.000000       0.152590       0.057328
BloodPressure                 0.141282  0.152590       1.000000       0.207371
SkinThickness                -0.081672  0.057328       0.207371       1.000000
Insulin                      -0.073535  0.331357       0.088933       0.436783
BMI                           0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction     -0.033523  0.137337       0.041265       0.183928
Age                           0.544341  0.263514       0.239528      -0.113970
Outcome                       0.221898  0.466581       0.065068       0.074752

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.073535  0.017683                 -0.033523
Glucose                   0.331357  0.221071                  0.137337
BloodPressure             0.088933  0.281805                  0.041265
SkinThickness             0.436783  0.392573                  0.183928
Insulin                   1.000000  0.197859                  0.185071
BMI                       0.197859  1.000000                  0.140647
DiabetesPedigreeFunction  0.185071  0.140647                  1.000000
Age                      -0.042163  0.036242                  0.033561
Outcome                   0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

[10]:
```
#Visualizing the correlation
sns.heatmap(correlation)
```

[10]: <Axes: >



Training the model with Logistic Regression

[11]:
```
X = ds.drop("Outcome",axis=1) #independent variable
Y = ds["Outcome"] #dependent variable
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2) #train-test␣
↪split with 0.2 test size
```

[12]:
```
model = LogisticRegression()
```

[13]:
```
model.fit(X_train,Y_train) #fitting train data of X with Y
```

```
C:\Users\Bikash shah\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
```

```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[13]:
```
LogisticRegression()
```

Making Prediction

[14]:
```
prediction = model.predict(X_test)
```

[15]:
```
print(prediction)
```

```
[0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 1 0
 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1
 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0
 1 0 0 1 0 0]
```

[16]:
```
accuracy = accuracy_score(prediction,Y_test)
print(accuracy)
```

```
0.7402597402597403
```

Hence, the accuracy of this model is 0.74.

[ ]:
```

```