Diabetic prediction system

January 15, 2024

[1]:
```
#DIABETIC PREDICTION SYSTEM
```

[2]:
```
#importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
data=pd.read_csv("diabetes.csv")
data.head()
```

[3]:

	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

[4]:
```
# information about the dataset
data.shape
```

[4]:
```
(768, 9)
```

[5]:
```
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
```

[6]:
```
data.dtypes
```

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

[7]:
```
data.describe().T
```

[7]:

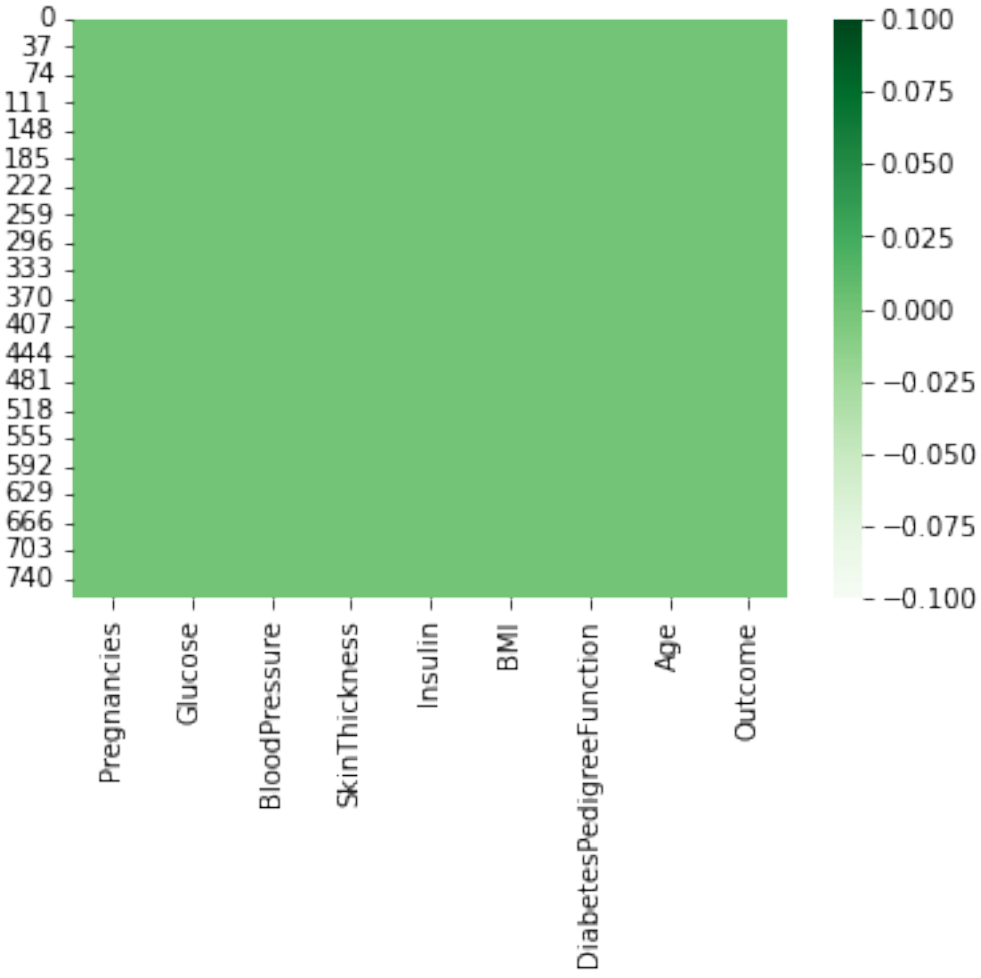
	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

[8]:
```
# Finding null values
data.isnull().sum()
```

[8]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[9]:
```
# visualization of the null values using heat map
sns.heatmap(data.isnull(), cmap="Greens")
# no null values present in the dataset
```

[9]:
```
<AxesSubplot: >
```

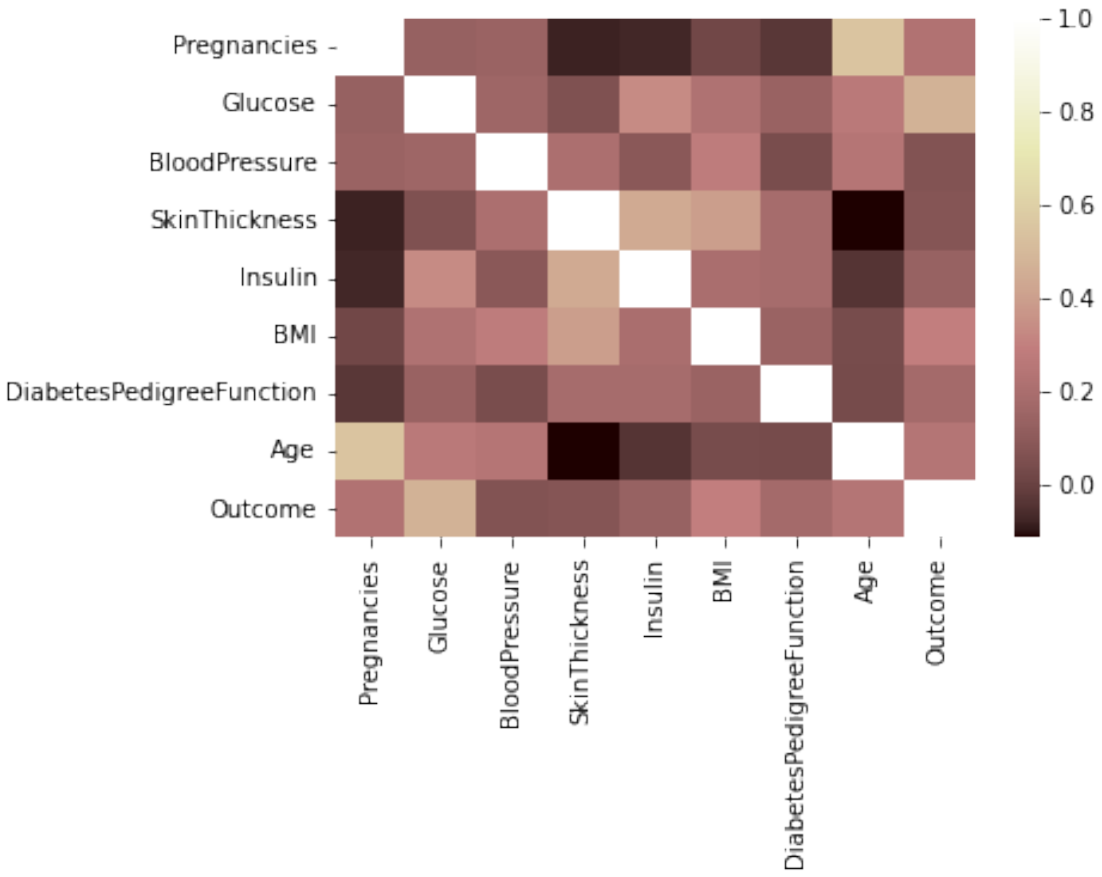


[10]:
```
# Correlation matrix
correlation=data.corr()
print(correlation)
```

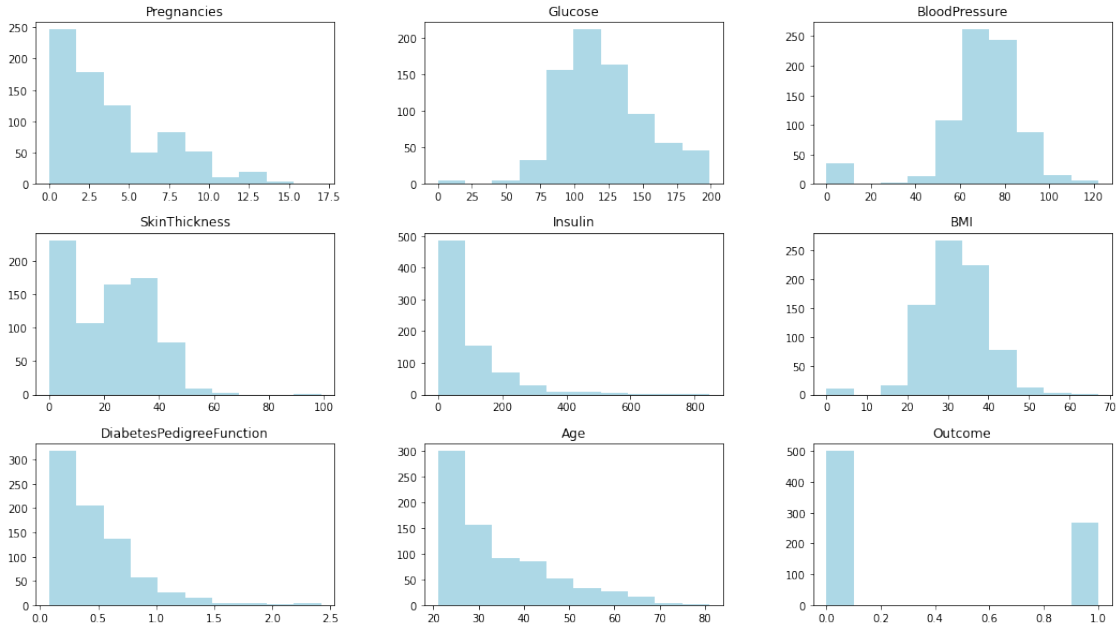
```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       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
```

[11]:
```
# visualizing the correlation
sns.heatmap(data.corr(), cmap="pink")
```

[11]: <AxesSubplot: >



[12]:
```
# creating histogram distribution in all levels
data.hist(figsize=(18, 10), grid=False, color='#ADD8E6')
plt.suptitle("Histogram Distribution levels", size=30)
plt.show()
```



[13]:
```
# Importing the libraries for prediction
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings("ignore")
```

[14]:
```
x=data.drop("Outcome", axis=1)
y=data["Outcome"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

[15]:
```
# In X all the independent variables are stored
# In Y the predictor variable("OUTCOME") is stored.
```

[ ]:
```

```

[16]:
```
# Training the model
training_model=LogisticRegression()
training_model.fit(x_train,y_train)
```

[16]:
```
LogisticRegression()
```

[17]:
```
# Fitting the X train and y train data into the variable called model
```

[18]:
```
# prediction making
prediction=training_model.predict(x_test)
print(prediction)
```

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

[19]:
```
accuracy=accuracy_score(prediction,y_test)
print(accuracy)
```

```
0.7857142857142857
```

[ ]:
```
# The accuracy of the model is then calculated and determined
```

[ ]:
```

```

[ ]:
```

```