diabetes-project

November 11, 2023

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Loading the Dataset

[4]:
```
data = pd.read_csv('diabetes.csv')
data
```

[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	
..	…	…	…	…	…	…	
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	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
..	…	…	…
763	0.171	63	0
764	0.340	27	0
765	0.245	30	0

```
766                    0.349   47        1
767                    0.315   23        0

[768 rows x 9 columns]
```

Data Exploration

[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.describe()
```

[6]:

	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

[7]:
```
data.head()
```

[7]:
```
   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
```

[10]:
```
data.tail()
```

[10]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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  Outcome
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

[12]:
```
data.shape
```

[12]:
```
(768, 9)
```

[13]:
```
data.columns
```

[13]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[14]:
```
data.dtypes
```

[14]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
```

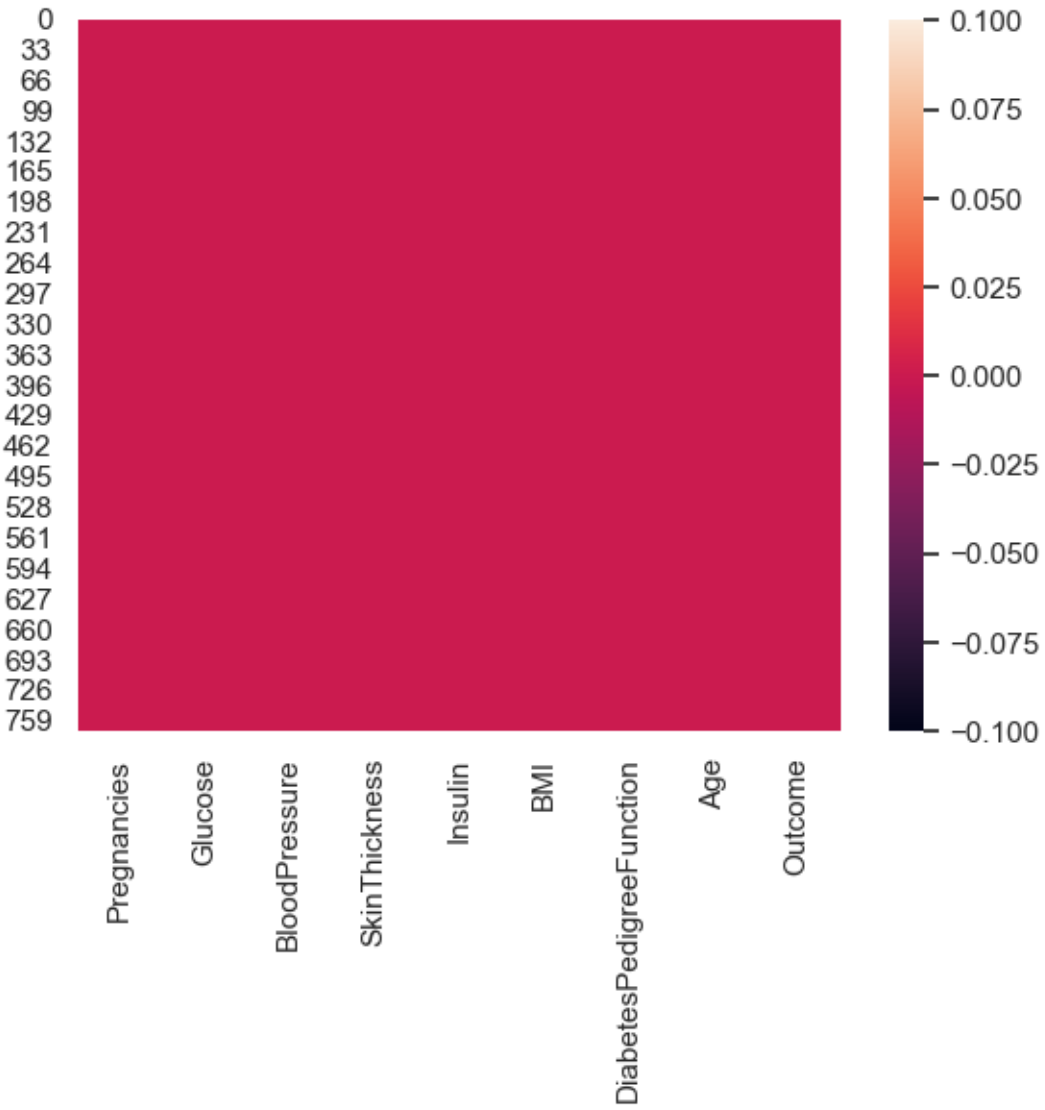
```
SkinThickness                int64
Insulin                      int64
BMI                        float64
DiabetesPedigreeFunction   float64
Age                          int64
Outcome                      int64
dtype: object
```

[15]:
```
data.isnull().sum()
```

[15]:
```
Pregnancies                0
Glucose                    0
BloodPressure              0
SkinThickness              0
Insulin                    0
BMI                        0
DiabetesPedigreeFunction   0
Age                        0
Outcome                    0
dtype: int64
```

[16]:
```
sns.heatmap(data.isnull())
```

[16]:
```
<Axes: >
```



[17]:
```
data.drop_duplicates()
```

[17]:

	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  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
..                        …   …       …
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[18]:
```
data.dropna()
```

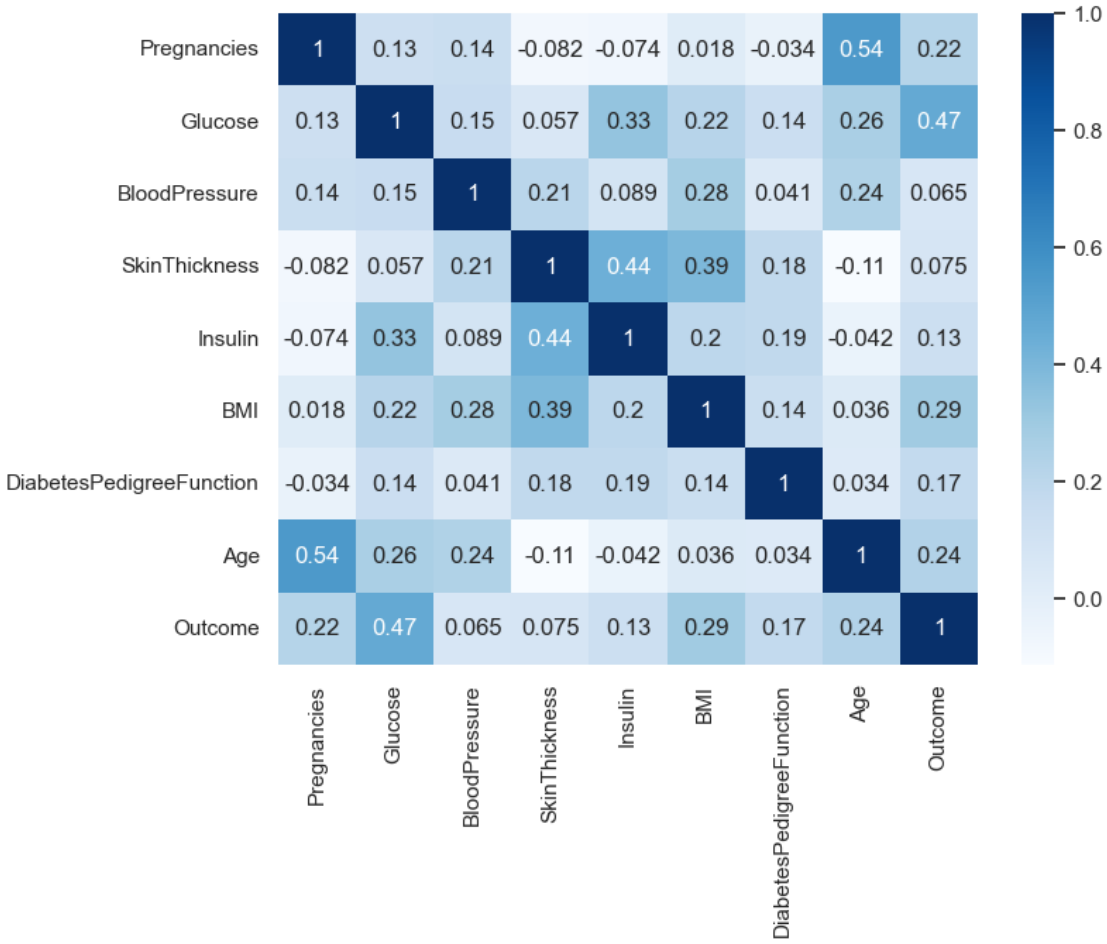
[18]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              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  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
..                        …   …       …
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

[22]:
```
correlation = data.corr(method='pearson')
```

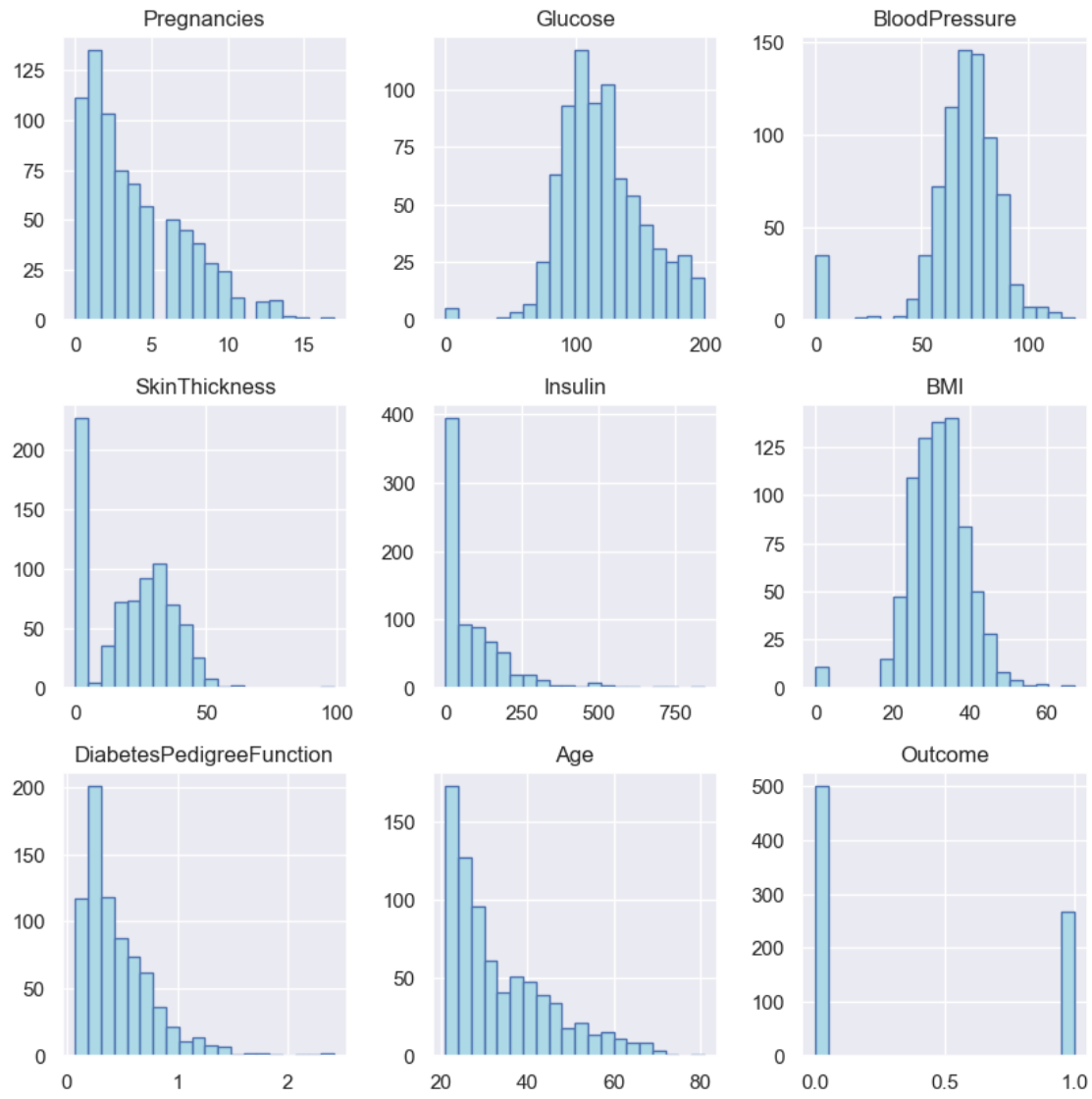
[26]:
```
plt.figure(figsize=(8,6))
sns.heatmap(correlation, annot=True, cmap='Blues')
plt.show()
```



Correlation observations 1. Pregnancies: Moderate correlation with Age (0.54) and weakly correlation with BMI (0.018) 2. Glucosa: Moderate correlation with Outcome (0.47) 3. BloodPressure: In general, a weak correlation with the other variables 4. SkinThickness: A moderate correlation with Insulin (0.44) and BMI (0.39)

Data Visualization

[35]:
```
data.hist(bins=20, figsize=(10,10), color='lightblue', edgecolor='b', alpha=1,␣
↪lw=1)
plt.show()
```



[37]:
```
skewness = data.skew()
print(skewness)
```

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```

Training the model

[38]:
```
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)
```

[39]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
c:\Users\Karen\AppData\Local\Programs\Python\Python312\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(
```

[39]:
```
LogisticRegression()
```

[40]:
```
prediction = model.predict(x_test)
```

[41]:
```
print(prediction)
```

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

Accuracy

[42]:
```
accuracy = accuracy_score(prediction, y_test)
```

[43]:
```
print(accuracy)
```

```
0.7597402597402597
```

Regression Metrics

[44]:
```
from sklearn import metrics
from sklearn.metrics import mean_squared_error
```

[46]:
```
print('MAE', metrics.mean_absolute_error(y_test, prediction))
print('MSE', mean_squared_error(y_test, prediction))
print('RMSE', np.sqrt(mean_squared_error(y_test, prediction)))
print('R squared', metrics.r2_score(y_test, prediction))
```

```
MAE 0.24025974025974026
MSE 0.24025974025974026
RMSE 0.4901629731627434
R squared -0.07428355957767718
```

Confusion Matrix

[52]:
```
from sklearn.metrics import confusion_matrix
```

[53]:
```
print(confusion_matrix(y_test, prediction))
```

```
[[88 14]
 [23 29]]
```

Classification Report

[54]:
```
from sklearn.metrics import classification_report
```

[55]:
```
c_report = classification_report(y_test, prediction)
print(c_report)
```

```
              precision    recall  f1-score   support

           0       0.79      0.86      0.83       102
           1       0.67      0.56      0.61        52

    accuracy                           0.76       154
   macro avg       0.73      0.71      0.72       154
weighted avg       0.75      0.76      0.75       154
```

[ ]:
```

```