Daibetes_analysis_ML(SVM)

August 17, 2025

[ ]:
```
import numpy as np
```

[ ]:
```
import pandas as pd
```

[ ]:
```
import seaborn as sns
```

[ ]:
```
import matplotlib.pyplot as plt
```

[ ]:
```
from sklearn.preprocessing import  StandardScaler
```

[ ]:
```
from sklearn.model_selection import train_test_split
```

[ ]:
```
from sklearn.metrics import accuracy_score, precision_score,␣
↪recall_score,confusion_matrix
```

[ ]:
```
from sklearn.impute import SimpleImputer
```

[ ]:
```
from sklearn import svm    # model that we are using
```

[ ]:
```
# importing Diabetes the data from source
import os
```

[ ]:
```
import os
```

[ ]:
```
os.getcwd()
```

[ ]:
```
'c:\\Users\\HP\\Downloads\\datascience.py\\data_files'
```

[ ]:
```
os.listdir(os.getcwd())
```

[ ]:
```
['diabetes.csv', 'IRIS.csv', 'irisdata.csv', 'Ng_MLY01_13.pdf']
```

[ ]:
```
os.listdir(os.chdir('c:\\Users\\HP\\Downloads\\datascience.py\\data_files'))
```

[ ]:
```
['diabetes.csv', 'IRIS.csv', 'irisdata.csv', 'Ng_MLY01_13.pdf']
```

[ ]:
```
diabetes_link = 'c:\\Users\\HP\\Downloads\\datascience.py\\data_files\\diabetes.
↪csv'
```

[ ]:
```
diabetes_data =  pd.read_csv(diabetes_link)
```

[ ]:
```
diabetes_data.head(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
5            5      116             74              0        0  25.6

   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                     0.201   30        0
```

[ ]:
```
# Data exploration
```

[ ]:
```
# colum, dim, structure and summary of data
```

[ ]:
```
diabetes_data.size    # get the size of the data
```

[ ]:
```
6912
```

[ ]:
```
diabetes_data.columns   # column names in data
```

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

[ ]:
```
diabetes_data.info()     # get the structure of the data
```

```
<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
```

[ ]:
```
diabetes_data.describe()    # describes the data
```

[ ]:
```
       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
```

[ ]:
```
#data Preparation
```

[ ]:
```
# rename the columns
```

[ ]:
```
data=diabetes_data.rename(columns={'Pregnancies':'n_pregnancies',
                                   'Glucose':'glucose','BloodPressure':'bp',
                                   'SkinThickness':'skinthic','Insulin':'insulin',
                                   'BMI':'bmi', 'DiabetesPedigreeFunction': 'dpf',
                                   'Age':'age', 'Outcome':'diab_present'})
```

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

[ ]:
```
   n_pregnancies  glucose  bp  skinthic  insulin   bmi    dpf  age  \
0              6      148  72        35        0  33.6  0.627   50
1              1       85  66        29        0  26.6  0.351   31
2              8      183  64         0        0  23.3  0.672   32
3              1       89  66        23       94  28.1  0.167   21
4              0      137  40        35      168  43.1  2.288   33

   diab_present
0             1
```

```
1              0
2              1
3              0
4              1
```

[ ]:
```
#duplicate
```

[ ]:
```
data.duplicated().sum()    #or
```

[ ]:
```
np.int64(0)
```

[ ]:
```
data.duplicated().value_counts()
```

[ ]:
```
False    768
Name: count, dtype: int64
```

[ ]:
```
data.isna().value_counts()
```

[ ]:
```
n_pregnancies  glucose  bp     skinthic  insulin  bmi    dpf    age
diab_present
False          False    False  False     False    False  False  False  False
768
Name: count, dtype: int64
```

[ ]:
```
insulin_column = data.loc[:,"insulin"]
```

[ ]:
```
insulin_column.describe()
```

[ ]:
```
count    768.000000
mean      79.799479
std      115.244002
min        0.000000
25%        0.000000
50%       30.500000
75%      127.250000
max      846.000000
Name: insulin, dtype: float64
```

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

[ ]:
```
   n_pregnancies  glucose  bp  skinthic  insulin   bmi    dpf  age  \
0              6      148  72        35        0  33.6  0.627   50
1              1       85  66        29        0  26.6  0.351   31
2              8      183  64         0        0  23.3  0.672   32
3              1       89  66        23       94  28.1  0.167   21
4              0      137  40        35      168  43.1  2.288   33

   diab_present
```

```
0              1
1              0
2              1
3              0
4              1
```

[ ]:
```
#check if there are missing values...in
```

[ ]:
```
#this case 0 are in each column
```

[ ]:
```
#'Glucose','BloodPressure','SkinThickness','Insulin','BMI'
```

[ ]:
```
# cant have a 0 values this signifies invalid info,
```

[ ]:
```
# we need to replace this values with
```

[ ]:
```
# check each value with 0's
```

[ ]:
```
data[data['glucose'] == 0] #to identify the rows
```

[ ]:
```
     n_pregnancies  glucose  bp  skinthic  insulin   bmi    dpf  age  \
75               1        0  48        20        0  24.7  0.140   22
182              1        0  74        20       23  27.7  0.299   21
342              1        0  68        35        0  32.0  0.389   22
349              5        0  80        32        0  41.0  0.346   37
502              6        0  68        41        0  39.0  0.727   41

     diab_present
75              0
182             0
342             0
349             1
502             1
```

[ ]:
```
(data['glucose'] == 0).value_counts() #to get the number of zeros
```

[ ]:
```
glucose
False    763
True       5
Name: count, dtype: int64
```

[ ]:
```
data[data['bp'] == 0] #to identify the rows
```

[ ]:
```
     n_pregnancies  glucose  bp  skinthic  insulin   bmi    dpf  age  \
7               10      115   0         0        0  35.3  0.134   29
15               7      100   0         0        0  30.0  0.484   32
49               7      105   0         0        0   0.0  0.305   24
60               2       84   0         0        0   0.0  0.304   21
```

78	0	131	0	0	0	43.2	0.270	26
81	2	74	0	0	0	0.0	0.102	22
172	2	87	0	23	0	28.9	0.773	25
193	11	135	0	0	0	52.3	0.578	40
222	7	119	0	0	0	25.2	0.209	37
261	3	141	0	0	0	30.0	0.761	27
266	0	138	0	0	0	36.3	0.933	25
269	2	146	0	0	0	27.5	0.240	28
300	0	167	0	0	0	32.3	0.839	30
332	1	180	0	0	0	43.3	0.282	41
336	0	117	0	0	0	33.8	0.932	44
347	3	116	0	0	0	23.5	0.187	23
357	13	129	0	30	0	39.9	0.569	44
426	0	94	0	0	0	0.0	0.256	25
430	2	99	0	0	0	22.2	0.108	23
435	0	141	0	0	0	42.4	0.205	29
453	2	119	0	0	0	19.6	0.832	72
468	8	120	0	0	0	30.0	0.183	38
484	0	145	0	0	0	44.2	0.630	31
494	3	80	0	0	0	0.0	0.174	22
522	6	114	0	0	0	0.0	0.189	26
533	6	91	0	0	0	29.8	0.501	31
535	4	132	0	0	0	32.9	0.302	23
589	0	73	0	0	0	21.1	0.342	25
601	6	96	0	0	0	23.7	0.190	28
604	4	183	0	0	0	28.4	0.212	36
619	0	119	0	0	0	32.4	0.141	24
643	4	90	0	0	0	28.0	0.610	31
697	0	99	0	0	0	25.0	0.253	22
703	2	129	0	0	0	38.5	0.304	41
706	10	115	0	0	0	0.0	0.261	30

	diab_present
7	0
15	1
49	0
60	0
78	1
81	0
172	0
193	1
222	0
261	1
266	1
269	1
300	1
332	1

```
336             0
347             0
357             1
426             0
430             0
435             1
453             0
468             1
484             1
494             0
522             0
533             0
535             1
589             0
601             0
604             1
619             1
643             0
697             0
703             0
706             1
```

[ ]:
```
(data['bp'] == 0).value_counts()#to get the number of zeros
```

[ ]:
```
bp
False    733
True      35
Name: count, dtype: int64
```

[ ]:
```
data[data['skinthic'] == 0] #to identify the rows
```

[ ]:
```
     n_pregnancies  glucose  bp  skinthic  insulin   bmi    dpf  age  \
2                8      183  64         0        0  23.3  0.672   32
5                5      116  74         0        0  25.6  0.201   30
7               10      115   0         0        0  35.3  0.134   29
9                8      125  96         0        0   0.0  0.232   54
10               4      110  92         0        0  37.6  0.191   30
..             ...      ...  ..       ...      ...   ...    ...  ...
757              0      123  72         0        0  36.3  0.258   52
758              1      106  76         0        0  37.5  0.197   26
759              6      190  92         0        0  35.5  0.278   66
762              9       89  62         0        0  22.5  0.142   33
766              1      126  60         0        0  30.1  0.349   47

     diab_present
2               1
5               0
```

```
7               0
9               1
10              0
..            …
757             1
758             0
759             1
762             0
766             1

[227 rows x 9 columns]
```

[ ]:
```
(data['skinthic'] == 0).value_counts()#to get the number of zeros
```

[ ]:
```
skinthic
False    541
True     227
Name: count, dtype: int64
```

[ ]:
```
data[data['bmi'] == 0] #to identify the rows
```

[ ]:
```
     n_pregnancies  glucose  bp  skinthic  insulin  bmi    dpf  age  \
9                8      125  96         0        0  0.0  0.232   54
49               7      105   0         0        0  0.0  0.305   24
60               2       84   0         0        0  0.0  0.304   21
81               2       74   0         0        0  0.0  0.102   22
145              0      102  75        23        0  0.0  0.572   21
371              0      118  64        23       89  0.0  1.731   21
426              0       94   0         0        0  0.0  0.256   25
494              3       80   0         0        0  0.0  0.174   22
522              6      114   0         0        0  0.0  0.189   26
684              5      136  82         0        0  0.0  0.640   69
706             10      115   0         0        0  0.0  0.261   30

     diab_present
9               1
49              0
60              0
81              0
145             0
371             0
426             0
494             0
522             0
684             0
706             1
```

```
[ ]: (data['bmi'] == 0).value_counts()#to get the number of zeros
```

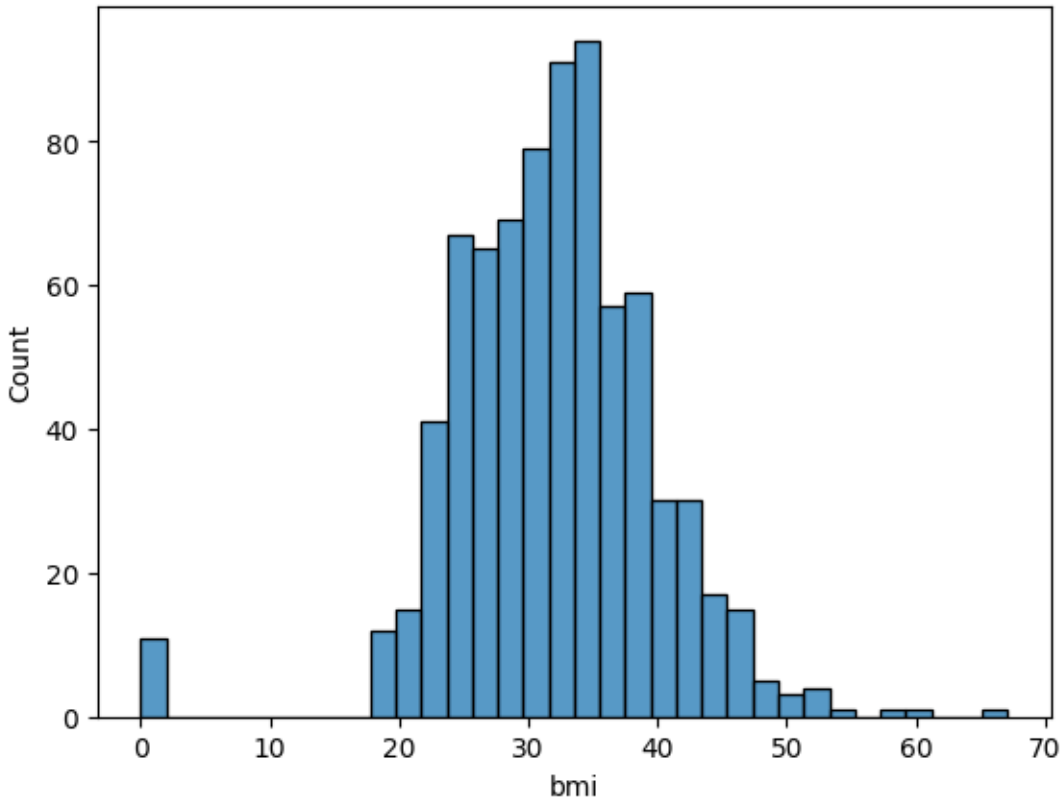
```
[ ]: bmi
     False    757
     True      11
     Name: count, dtype: int64
```

```
[ ]: data.skew()
```

```
[ ]: n_pregnancies    0.901674
     glucose          0.173754
     bp              -1.843608
     skinthic         0.109372
     insulin          2.272251
     bmi             -0.428982
     dpf              1.919911
     age              1.129597
     diab_present     0.635017
     dtype: float64
```

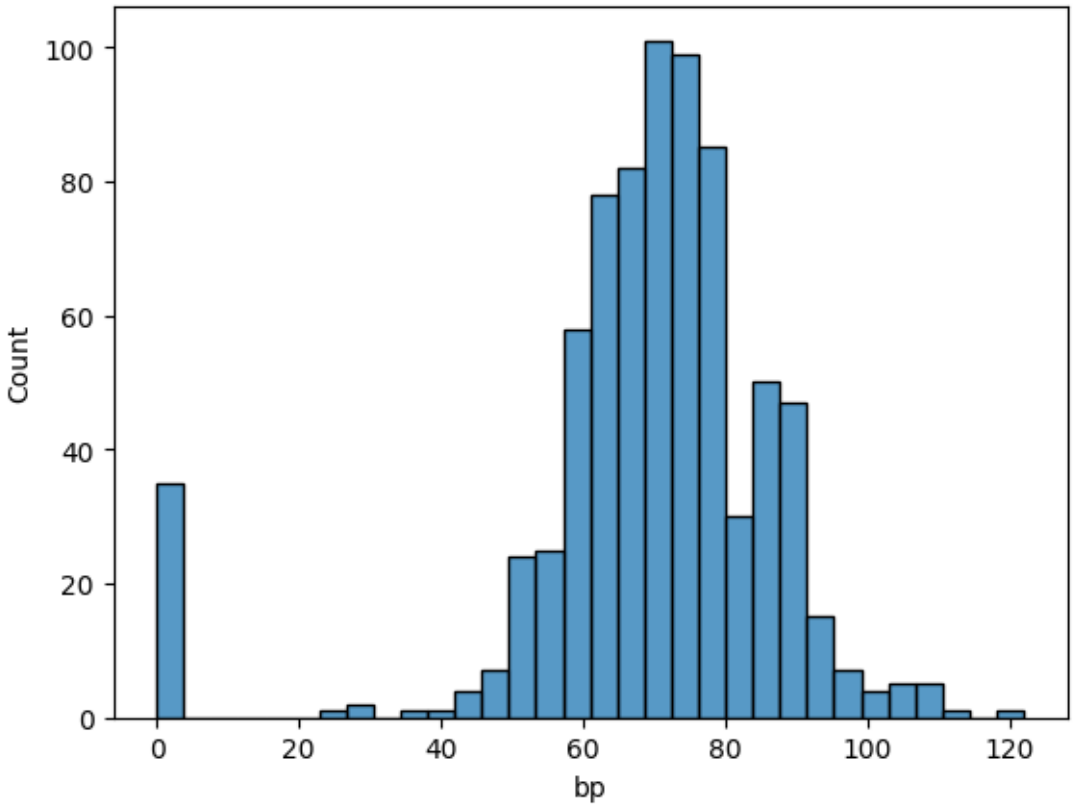
```
[ ]: sns.histplot(data['bmi'])
```

```
[ ]: <Axes: xlabel='bmi', ylabel='Count'>
```



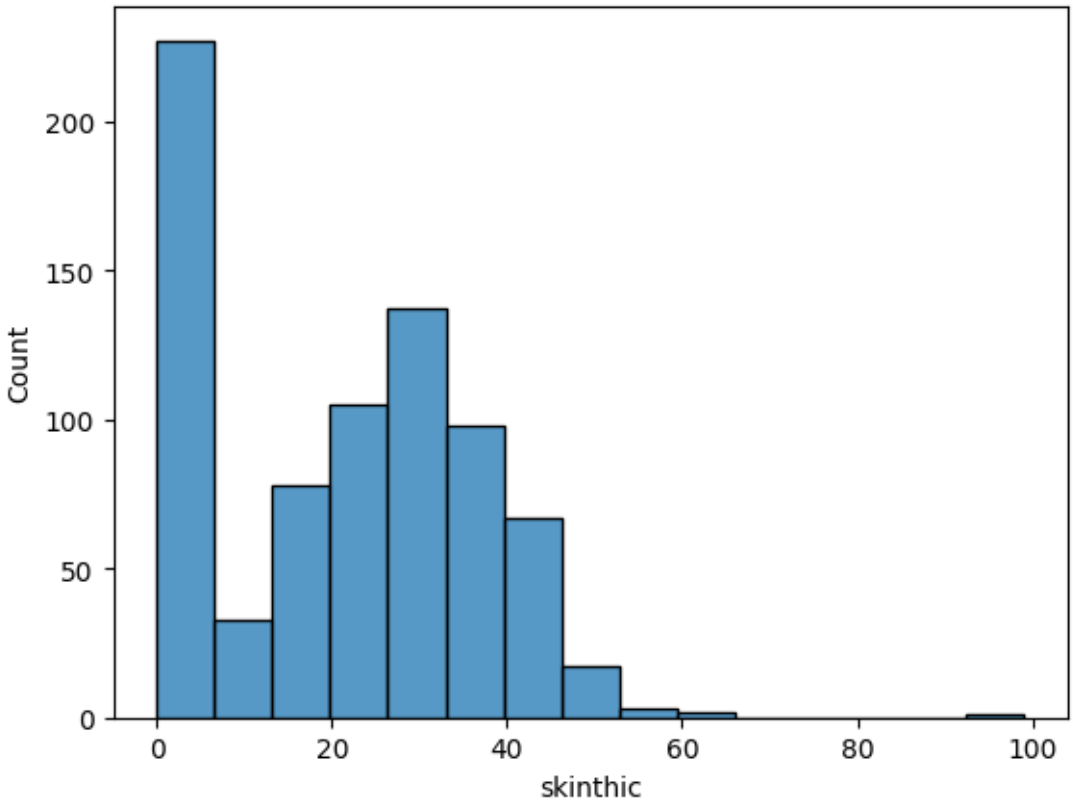
[ ]:
```
sns.histplot(data["bp"])
```

[ ]:
```
<Axes: xlabel='bp', ylabel='Count'>
```



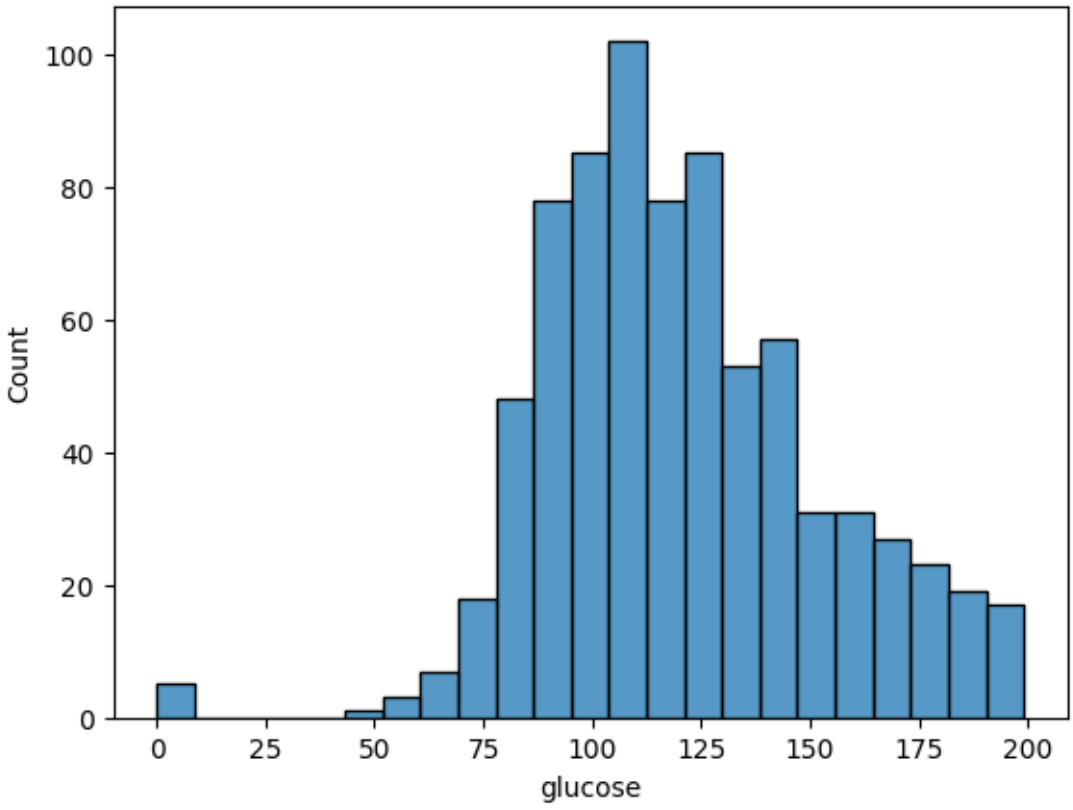
[ ]:
```
sns.histplot(data["skinthic"])
```

[ ]:
```
<Axes: xlabel='skinthic', ylabel='Count'>
```



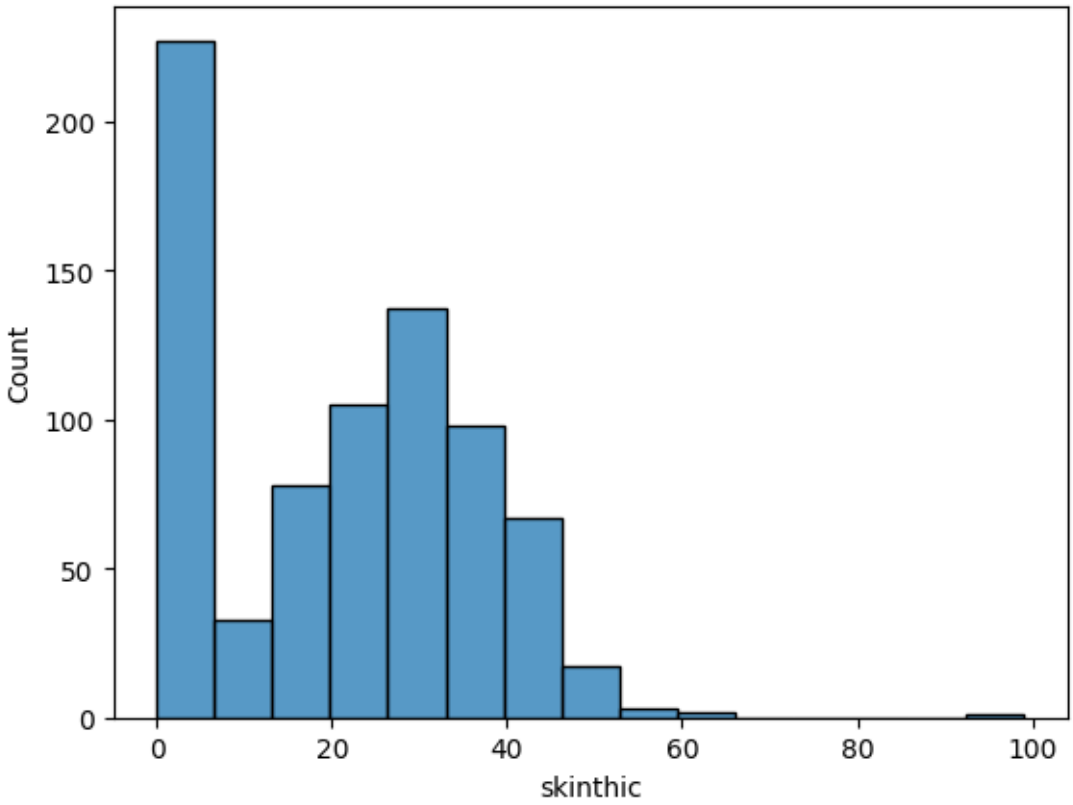
[ ]:
```
sns.histplot(data["glucose"])
```

[ ]:
```
<Axes: xlabel='glucose', ylabel='Count'>
```



[ ]:
```
sns.histplot(data['skinthic'])
```

[ ]:
```
<Axes: xlabel='skinthic', ylabel='Count'>
```



[ ]:
```
zero_values =['glucose','bp','skinthic','bmi']
```

[ ]:
```
imputer = SimpleImputer(missing_values=0,strategy="median")
```

[ ]:
```
data[zero_values] = imputer.fit_transform(data[zero_values])
```

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

[ ]:

	n_pregnancies	glucose	bp	skinthic	insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.656250	72.386719	29.108073	79.799479	
std	3.369578	30.438286	12.096642	8.791221	115.244002	
min	0.000000	44.000000	24.000000	7.000000	0.000000	
25%	1.000000	99.750000	64.000000	25.000000	0.000000	
50%	3.000000	117.000000	72.000000	29.000000	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	dpf	age	diab_present
count	768.000000	768.000000	768.000000	768.000000
mean	32.455208	0.471876	33.240885	0.348958

```
std      6.875177    0.331329   11.760232      0.476951
min     18.200000    0.078000   21.000000      0.000000
25%     27.500000    0.243750   24.000000      0.000000
50%     32.300000    0.372500   29.000000      0.000000
75%     36.600000    0.626250   41.000000      1.000000
max     67.100000    2.420000   81.000000      1.000000
```

[ ]:
```
(data[zero_values] == 0).sum()
```

[ ]:
```
glucose     0
bp          0
skinthic    0
bmi         0
dtype: int64
```

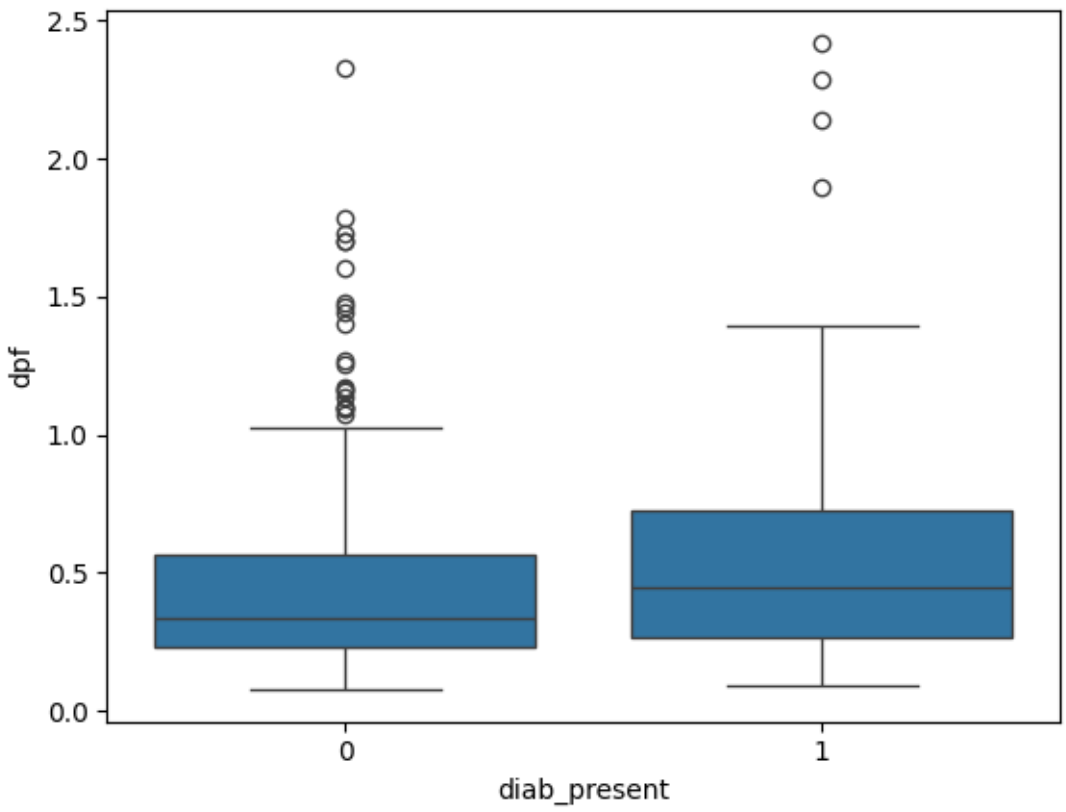
[ ]:
```
data.head()
```

[ ]:
```
   n_pregnancies  glucose    bp  skinthic  insulin   bmi    dpf  age  \
0              6    148.0  72.0      35.0        0  33.6  0.627   50
1              1     85.0  66.0      29.0        0  26.6  0.351   31
2              8    183.0  64.0      29.0        0  23.3  0.672   32
3              1     89.0  66.0      23.0       94  28.1  0.167   21
4              0    137.0  40.0      35.0      168  43.1  2.288   33

   diab_present
0             1
1             0
2             1
3             0
4             1
```

[ ]:
```
sns.boxplot(x= 'diab_present', y= "dpf",data=data)
```

[ ]:
```
<Axes: xlabel='diab_present', ylabel='dpf'>
```



[ ]:
```
scale_data = ['n_pregnancies', 'glucose', 'bp', 'skinthic', 'bmi', 'dpf',␣
↪'age','insulin']
```

[ ]:
```
scaler = StandardScaler()
```

[ ]:
```
data[scale_data] =scaler.fit_transform(data[scale_data])
```

[ ]:
```
X = data.drop("diab_present",axis = 1)
```

[ ]:
```
Y = data['diab_present']
```

[ ]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2,
                                                 random_state = 2)
```

[ ]:
```
print(X.shape,X_test.shape,X_train.shape)
```

```
(768, 8) (154, 8) (614, 8)
```

[ ]:
```
print(Y.shape,Y_test.shape,Y_train.shape)
```

```
(768,) (154,) (614,)
```

[ ]:
```
classifier = svm.SVC(kernel='linear')
```

[ ]:
```
classifier.fit(X_train,Y_train)
```

[ ]:
```
SVC(kernel='linear')
```

[ ]:
```
X_train_prediction = classifier.predict(X_train)
```

[ ]:
```
X_train_prediction = classifier.predict(X_train)
```

[ ]:
```
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

[ ]:
```
print('Accuracy :', training_data_accuracy)
```

```
Accuracy : 0.7703583061889251
```

[ ]:
```
print('Accuracy :', training_data_accuracy)
```

```
Accuracy : 0.7703583061889251
```

[ ]:
```
X_test_prediction = classifier.predict(X_test)
```

[ ]:
```
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

[ ]:
```
print('Accuracy :', test_data_accuracy,'the model has not overtrain hence there␣
↪is no overfitting')
```

```
Accuracy : 0.7662337662337663 the model has not overtrain hence there is no
overfitting
```

[ ]:
```
input_data = (0,137, 40,35, 168,43,2,33)
```

[ ]:
```
input_data_as_numpy_array = np.array(input_data)
```

[ ]:
```
reshape_input_data_as_numpy_array = input_data_as_numpy_array.reshape(1,-1)
```

[ ]:
```
std_data1 = scaler.transform(reshape_input_data_as_numpy_array )
```

```
c:\Users\HP\AppData\Local\Programs\Python\Python313\Lib\site-
packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid
feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

[ ]:
```
prediction = classifier.predict(std_data1)
```

```
c:\Users\HP\AppData\Local\Programs\Python\Python313\Lib\site-
packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid
feature names, but SVC was fitted with feature names
  warnings.warn(
```

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

```
[1]
```

[ ]:
```
diabetes_data.iloc[4,:]
```

[ ]:
```
Pregnancies                 0.000
Glucose                   137.000
BloodPressure              40.000
SkinThickness              35.000
Insulin                   168.000
BMI                        43.100
DiabetesPedigreeFunction    2.288
Age                        33.000
Outcome                     1.000
Name: 4, dtype: float64
```