At St. Xavier's College, a Faculty has the following data in My SQL in database named as Class having

table student related to Semester Examination

Enrollment No. Student Name Section Subject Id Marks

1 Tim A 1 70

2 Jim A 2 75

3 Kim B 3 65

4 Tom B 4 77

5 John C 5 60

6 Joe C 1 82

7 James B 2 76

8 Henry C 5 68

9 Matt B 3 71

10 Paul A 4 79

The Faculty needs a section-wise Number of candidates who have secured more than or equal to 75

marks in the Semester Exam.

Note: Enrollment No. is declared as Primary Key

We need to create a table like this on our database so create a database(my db is task1 given)

```
No default schema selected; type \use <schema> to set one.
MySQL  localhost:33060+ ssl  SQL > use task1;
Default schema set to `task1`.
Fetching global names, object names from `task1` for auto-completion... Press ^C to stop.
```

then create a table and insert all values (i created te table name college)

```
MySQL  localhost:33060+ ssl  task1  SQL > select*from college;
+--------------+--------------+---------+--------+-------+
| enrollment_no | student_name | section | sub_id | Marks |
+--------------+--------------+---------+--------+-------+
|            1 | Tim          | A       |      1 |    70 |
|            2 | jim          | A       |      2 |    75 |
|            3 | kim          | B       |      3 |    65 |
|            4 | tom          | B       |      4 |    77 |
|            5 | john         | C       |      5 |    60 |
|            6 | joe          | C       |      1 |    82 |
|            7 | james        | B       |      2 |    76 |
|            8 | Heney        | C       |      5 |    68 |
|            9 | matt         | B       |      3 |    71 |
|           10 | paul         | A       |      4 |    79 |
+--------------+--------------+---------+--------+-------+
10 rows in set (0.0020 sec)
```

after view the table

execute the query

select section,count(section) as 'No of candidates reater ten or equal to 75 marks' from college group by section;

```
MySQL  localhost:33060+ ssl  task1  SQL > select section,count(section) as 'No.of Candidates greater than or
+---------+-----------------------------------------------+
| section | No.of Candidates greater than or equal to 75 marks |
+---------+-----------------------------------------------+
| A       |                                             3 |
| B       |                                             4 |
| C       |                                             3 |
+---------+-----------------------------------------------+
3 rows in set (0.0013 sec)
MySQL  localhost:33060+ ssl  task1  SQL >
```

Tableau

#Save the data in a csv file format

#Import the data to the working sheet

#Go to sheet 1

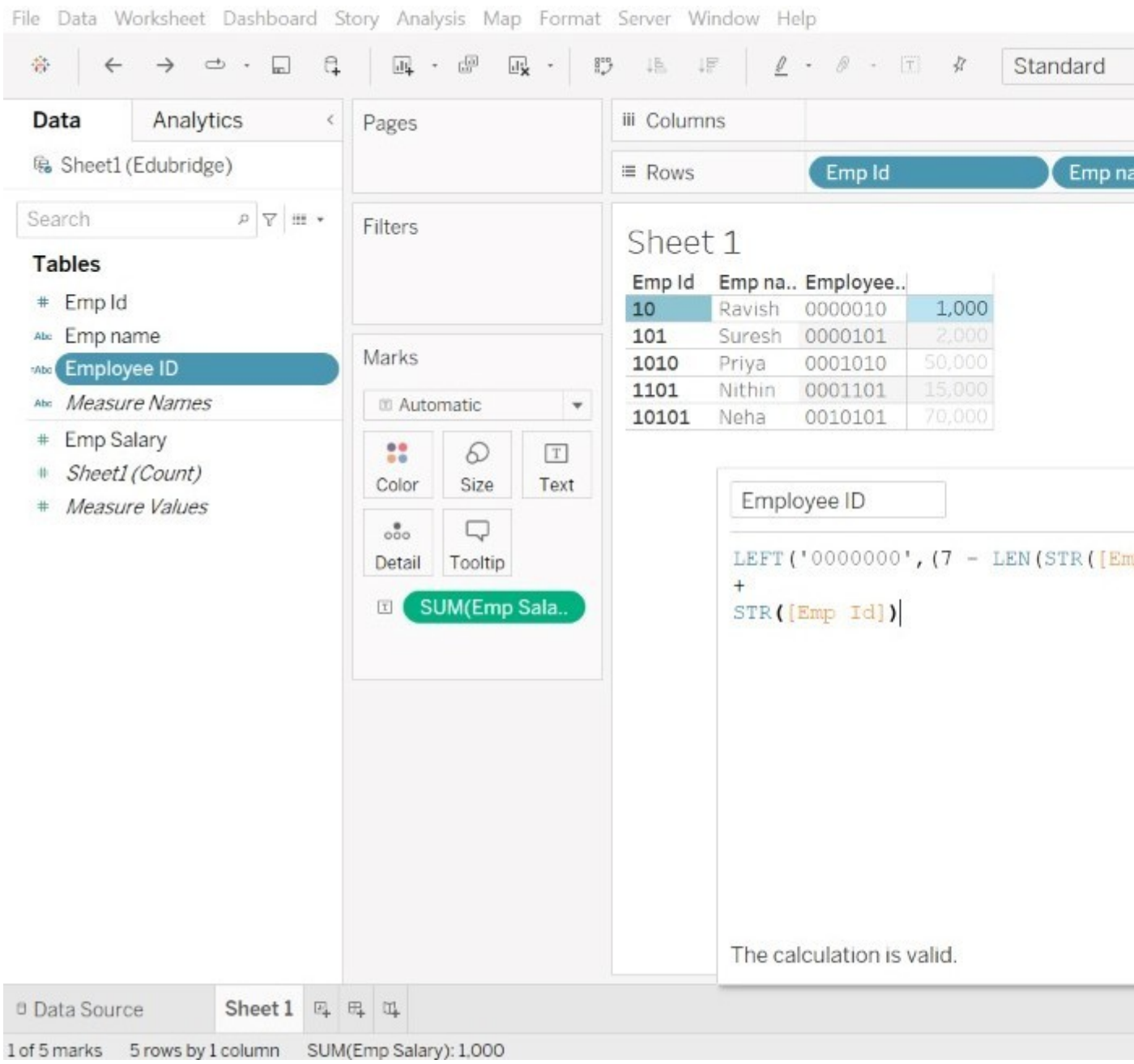
#Drag emp name and id

#Right click on the id on the left side

#a drop down box is appear

#Rename that dialogue box type

LEFT(“0000000”,(7-LEN(STR[ID])+STR([ID])



#A new column empid is created under tables ,drag and drop it into the rows

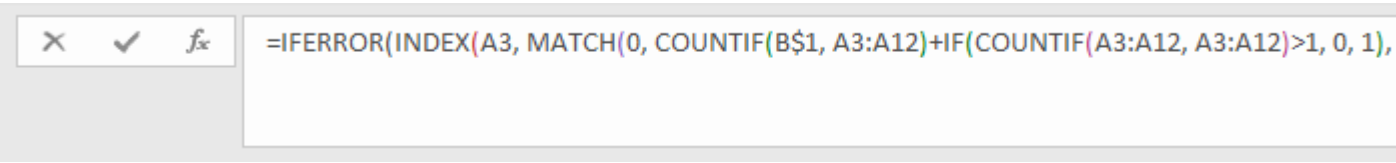
Task2

#Need to include these data in excel

#then select the column and o to home go to filter and then go to advanced select duplicates then highlight the duplicates

#then execute the query:

=IFERROR(INDEX(A3, MATCH(0, COUNTIF(B$1, A3:A12)+IF(COUNTIF(A3:A12, A3:A12)>1, 0, 1), 0)), "")



	A	B
	Customer name	Duplicate
	Akshay Rathod	
	Amit kumar	Amit kumar
	Amit kumar	
	Animesh verma	
	Arti Ahuja	Arti Ahuja
	Arti Ahuja	
	Ashutosh Mahajan	
	Eshank sharma	Eshank sharma
	Eshank sharma	
	Harmeet kaur	
	Kapil khatri	Kapil khatri
	Kapil khatri	
	Mohit Jain	
	Raj Sharma	
	Sunil Yadav	
	Swati Singh	

Fatal prediction using KNN Classifier

Importing required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Read the csv file

```
df=pd.read_csv("train.csv")
df.head()
```

```
   baseline value  accelerations  fetal_movement  uterine_contractions
\
0           142.0          0.000           0.000                 0.007

1           122.0          0.000           0.000                 0.006

2           129.0          0.005           0.003                 0.001

3           136.0          0.006           0.000                 0.008

4           144.0          0.000           0.000                 0.006


   light_decelerations  severe_decelerations  prolongued_decelerations
\
0                0.000                   0.0                       0.0

1                0.002                   0.0                       0.0

2                0.000                   0.0                       0.0

3                0.000                   0.0                       0.0

4                0.000                   0.0                       0.0


   abnormal_short_term_variability
mean_value_of_short_term_variability  \
0                             58.0
0.4
1                             27.0
```

```
1.4
2                              34.0
1.7
3                              45.0
0.8
4                              32.0
1.0

   percentage_of_time_with_abnormal_long_term_variability  ...
histogram_min  \
0                                                  9.0       ...
136.0
1                                                  4.0       ...
91.0
2                                                  0.0       ...
78.0
3                                                  2.0       ...
129.0
4                                                  0.0       ...
122.0

   histogram_max  histogram_number_of_peaks
histogram_number_of_zeroes  \
0          156.0                        0.0
0.0
1          144.0                        4.0
0.0
2          196.0                       10.0
0.0
3          158.0                        2.0
0.0
4          160.0                        1.0
0.0

   histogram_mode  histogram_mean  histogram_median
histogram_variance  \
0           148.0           147.0             149.0
1.0
1           126.0           120.0             122.0
6.0
2           137.0           136.0             137.0
6.0
3           144.0           143.0             145.0
1.0
4           150.0           147.0             149.0
2.0

   histogram_tendency  fetal_health
0                 0.0           1.0
1                 0.0           1.0
```

```
2                 0.0           1.0
3                 0.0           1.0
4                 1.0           1.0

[5 rows x 22 columns]
```

Finding the info

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1700 entries, 0 to 1699
Data columns (total 22 columns):
 #   Column                                                  Non-Null
Count  Dtype
---  ------
--------------  -----
 0   baseline value                                          1700 non-
null   float64
 1   accelerations                                           1700 non-
null   float64
 2   fetal_movement                                          1700 non-
null   float64
 3   uterine_contractions                                    1700 non-
null   float64
 4   light_decelerations                                     1700 non-
null   float64
 5   severe_decelerations                                    1700 non-
null   float64
 6   prolongued_decelerations                                1700 non-
null   float64
 7   abnormal_short_term_variability                         1700 non-
null   float64
 8   mean_value_of_short_term_variability                    1700 non-
null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  1700 non-
null   float64
 10  mean_value_of_long_term_variability                     1700 non-
null   float64
 11  histogram_width                                         1700 non-
null   float64
 12  histogram_min                                           1700 non-
null   float64
 13  histogram_max                                           1700 non-
null   float64
 14  histogram_number_of_peaks                               1700 non-
null   float64
 15  histogram_number_of_zeroes                              1700 non-
null   float64
 16  histogram_mode                                          1700 non-
null   float64
```

```
 17  histogram_mean                                          1700 non-
null   float64
 18  histogram_median                                        1700 non-
null   float64
 19  histogram_variance                                      1700 non-
null   float64
 20  histogram_tendency                                      1700 non-
null   float64
 21  fetal_health                                            1700 non-
null   float64
dtypes: float64(22)
memory usage: 292.3 KB
```

```
df.shape
```

```
(1700, 22)
```

```
# check missing value of the data
df.isna().sum()
```

```
baseline value                                            0
accelerations                                             0
fetal_movement                                            0
uterine_contractions                                      0
light_decelerations                                       0
severe_decelerations                                      0
prolongued_decelerations                                  0
abnormal_short_term_variability                           0
mean_value_of_short_term_variability                      0
percentage_of_time_with_abnormal_long_term_variability    0
mean_value_of_long_term_variability                       0
histogram_width                                           0
histogram_min                                             0
histogram_max                                             0
histogram_number_of_peaks                                 0
histogram_number_of_zeroes                                0
histogram_mode                                            0
histogram_mean                                            0
histogram_median                                          0
histogram_variance                                        0
histogram_tendency                                        0
fetal_health                                              0
dtype: int64
```

```
# describe numeric column
df.describe()
```

```
       baseline value  accelerations  fetal_movement
uterine_contractions  \
count     1700.000000    1700.000000     1700.000000
1700.000000
```

```
mean       133.213529       0.003212        0.010211
0.004356
std          9.873344       0.003888        0.050124
0.002943
min        106.000000       0.000000        0.000000
0.000000
25%        126.000000       0.000000        0.000000
0.002000
50%        133.000000       0.002000        0.000000
0.004000
75%        140.000000       0.006000        0.003000
0.006000
max        159.000000       0.019000        0.481000
0.015000

       light_decelerations  severe_decelerations
prolongued_decelerations  \
count          1700.000000           1700.000000
1700.000000
mean              0.001899              0.000004
0.000158
std               0.002976              0.000059
0.000587
min               0.000000              0.000000
0.000000
25%               0.000000              0.000000
0.000000
50%               0.000000              0.000000
0.000000
75%               0.003000              0.000000
0.000000
max               0.015000              0.001000
0.005000

       abnormal_short_term_variability
mean_value_of_short_term_variability  \
count                      1700.000000
1700.000000
mean                         46.508824
1.345353
std                          17.276801
0.898037
min                          12.000000
0.200000
25%                          32.000000
0.700000
50%                          48.000000
1.200000
75%                          61.000000
1.700000
```

```
max                          87.000000
7.000000

       percentage_of_time_with_abnormal_long_term_variability  ...  \
count                                        1700.000000       ...
mean                                            9.738235       ...
std                                            18.227303       ...
min                                             0.000000       ...
25%                                             0.000000       ...
50%                                             0.000000       ...
75%                                            11.000000       ...
max                                            91.000000       ...

       histogram_min  histogram_max  histogram_number_of_peaks  \
count    1700.000000    1700.000000                1700.000000
mean       93.121176     163.842353                   4.088824
std        29.520766      17.651851                   2.927774
min        50.000000     122.000000                   0.000000
25%        67.000000     152.000000                   2.000000
50%        93.000000     162.000000                   4.000000
75%       120.000000     174.000000                   6.000000
max       159.000000     238.000000                  18.000000

       histogram_number_of_zeroes  histogram_mode  histogram_mean  \
count                 1700.000000     1700.000000     1700.000000
mean                     0.324118      137.128235      134.461176
std                      0.683795       16.608926       15.616890
min                      0.000000       60.000000       75.000000
25%                      0.000000      129.000000      125.000000
50%                      0.000000      139.000000      135.500000
75%                      0.000000      148.000000      145.000000
max                     10.000000      187.000000      182.000000

       histogram_median  histogram_variance  histogram_tendency
fetal_health
count       1700.000000         1700.000000         1700.000000
1700.000000
mean         137.862941           19.046471            0.324706
1.304706
std           14.552801           29.575447            0.608340
0.614788
min           77.000000            0.000000           -1.000000
1.000000
25%          128.000000            2.000000            0.000000
1.000000
50%          139.000000            7.000000            0.000000
1.000000
75%          148.000000           24.000000            1.000000
1.000000
max          186.000000          269.000000            1.000000
```

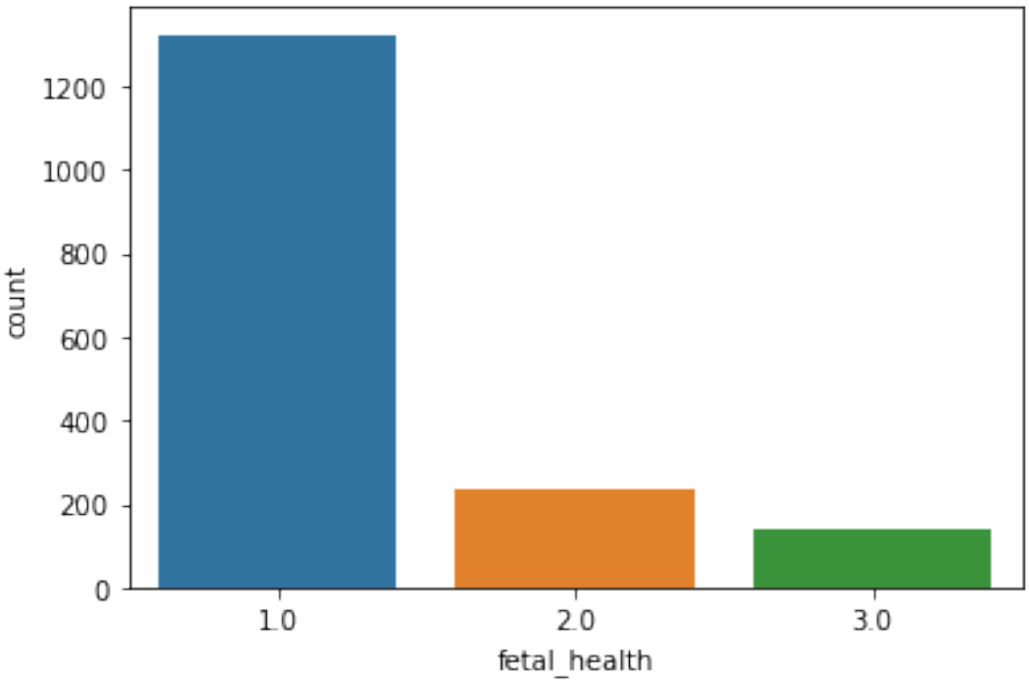
```
3.000000

[8 rows x 22 columns]
```

To understand the different types in our target variable (fatal_health)

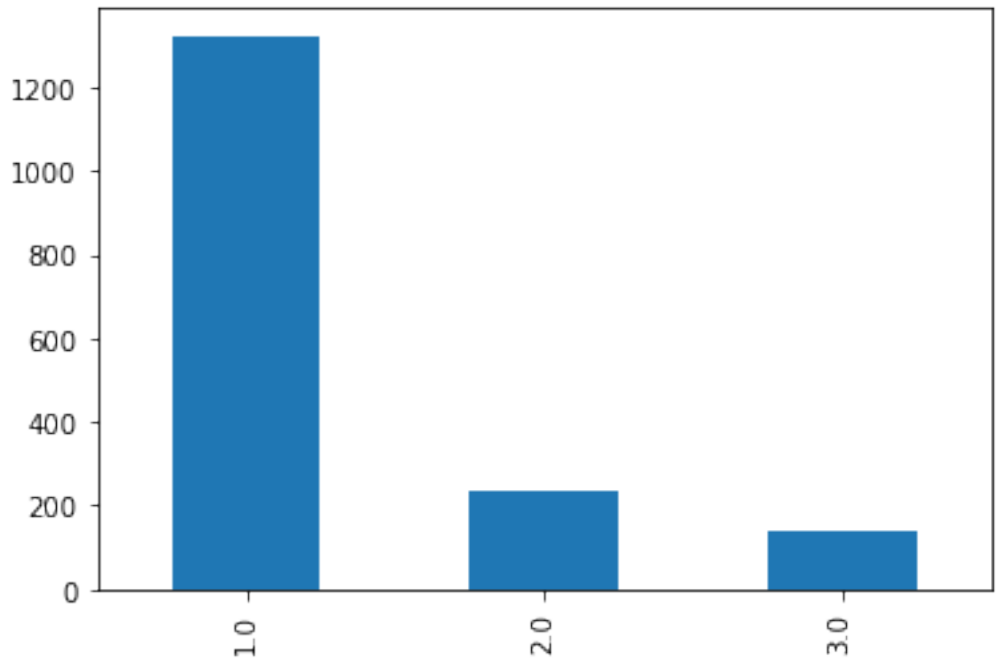
```
sns.countplot(df.fetal_health)
```

```
<AxesSubplot:xlabel='fetal_health', ylabel='count'>
```




```
df['fetal_health'].value_counts().plot(kind='bar')
```

```
<AxesSubplot:>
```



to find all datatypes in our file

```
cats = list(df.select_dtypes(include=['object','bool']) )
nums = list(df.select_dtypes(include=['int64','float64']))
print(cats)
print(nums)
```

```
[]
['baseline value', 'accelerations', 'fetal_movement',
'uterine_contractions', 'light_decelerations', 'severe_decelerations',
'prolongued_decelerations', 'abnormal_short_term_variability',
'mean_value_of_short_term_variability',
'percentage_of_time_with_abnormal_long_term_variability',
'mean_value_of_long_term_variability', 'histogram_width',
'histogram_min', 'histogram_max', 'histogram_number_of_peaks',
'histogram_number_of_zeroes', 'histogram_mode', 'histogram_mean',
'histogram_median', 'histogram_variance', 'histogram_tendency',
'fetal_health']
```

splitting the dataset X,y

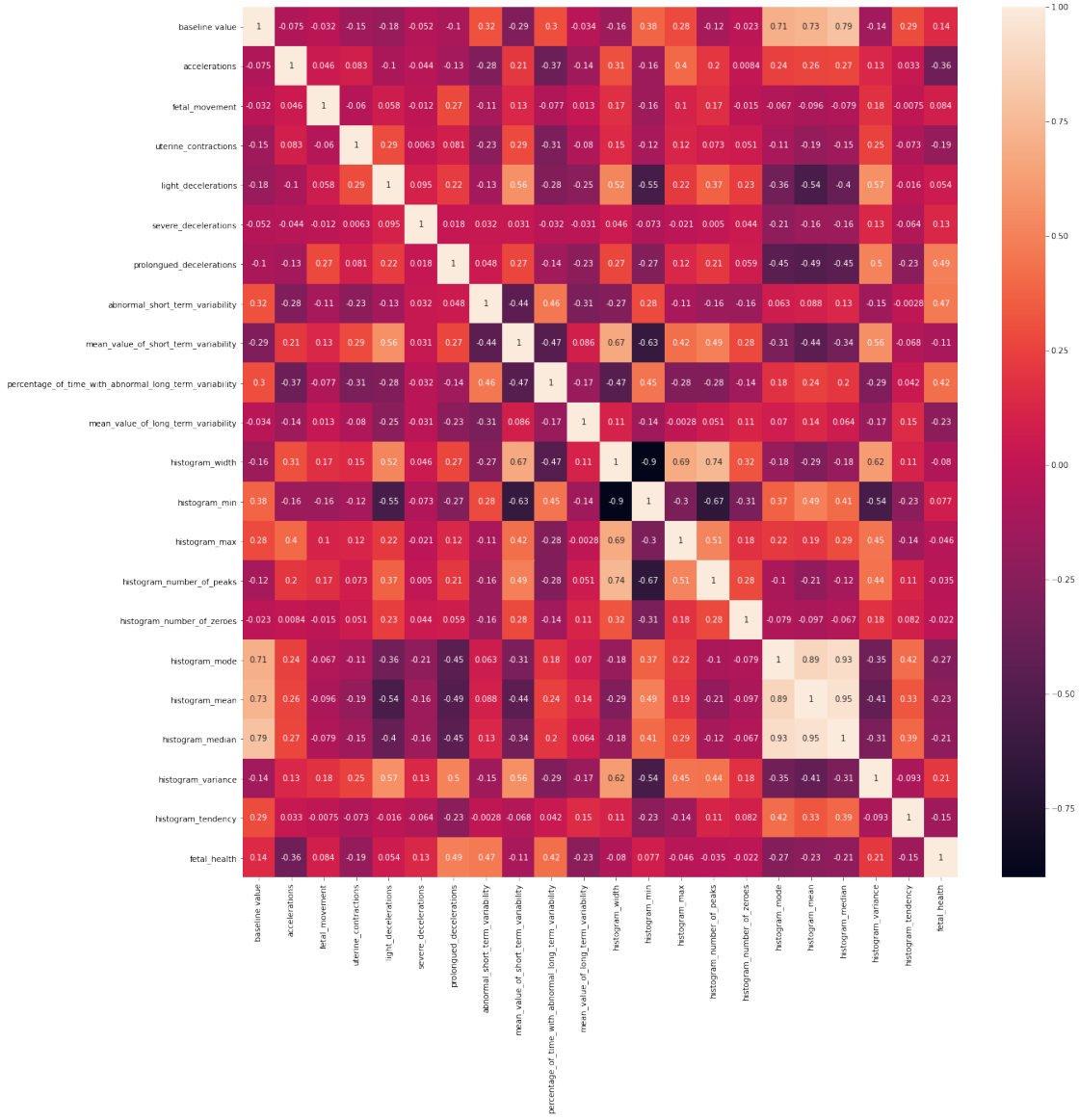
```
X=df.iloc[:,:-1].values
y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,rando
m_state=21)
```

Finding the correlation of df

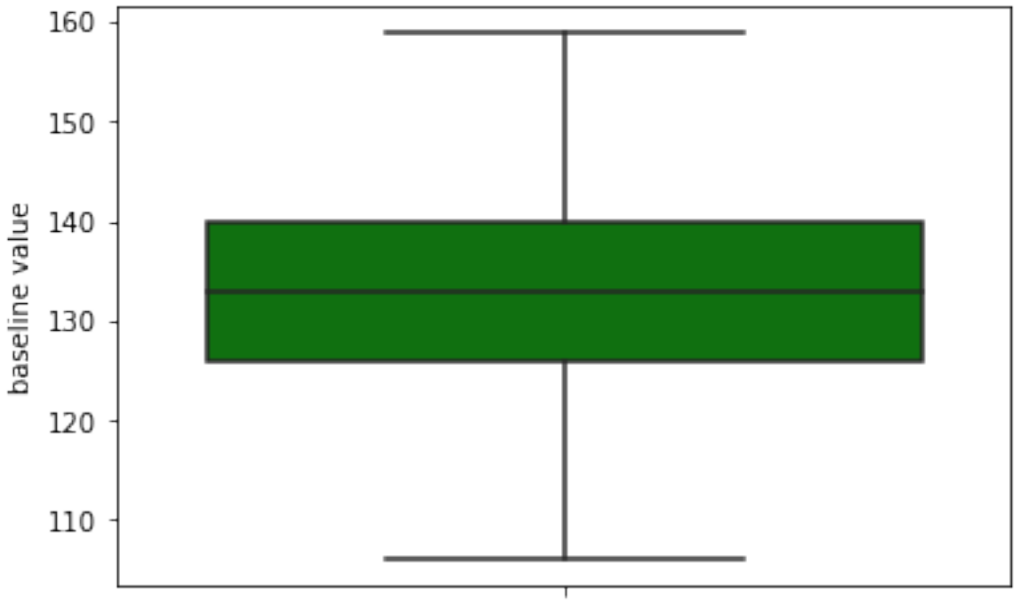
```
corr=df.corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr,annot=True)
```

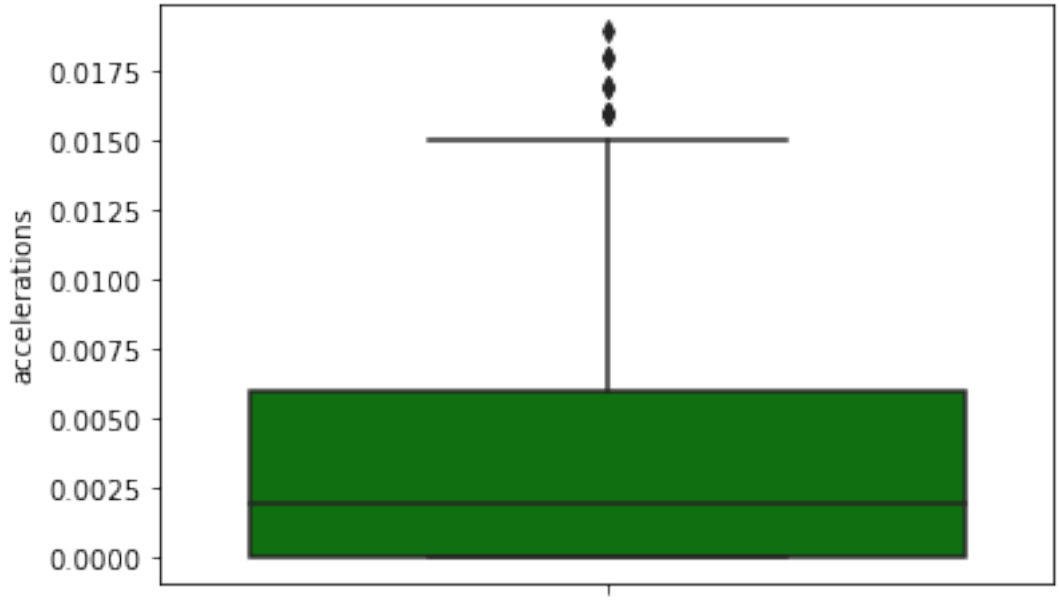
```
<AxesSubplot:>
```

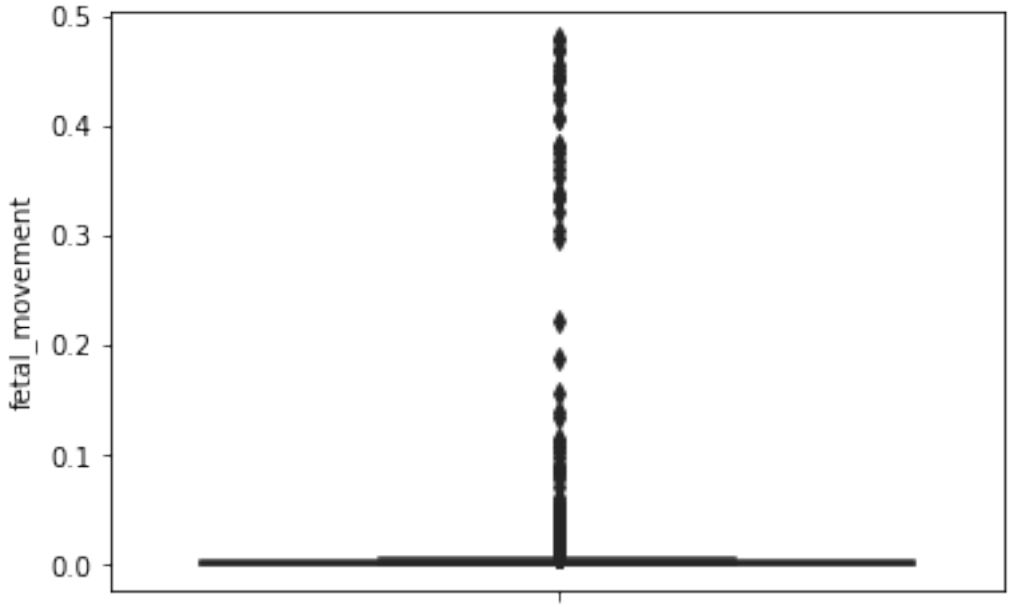


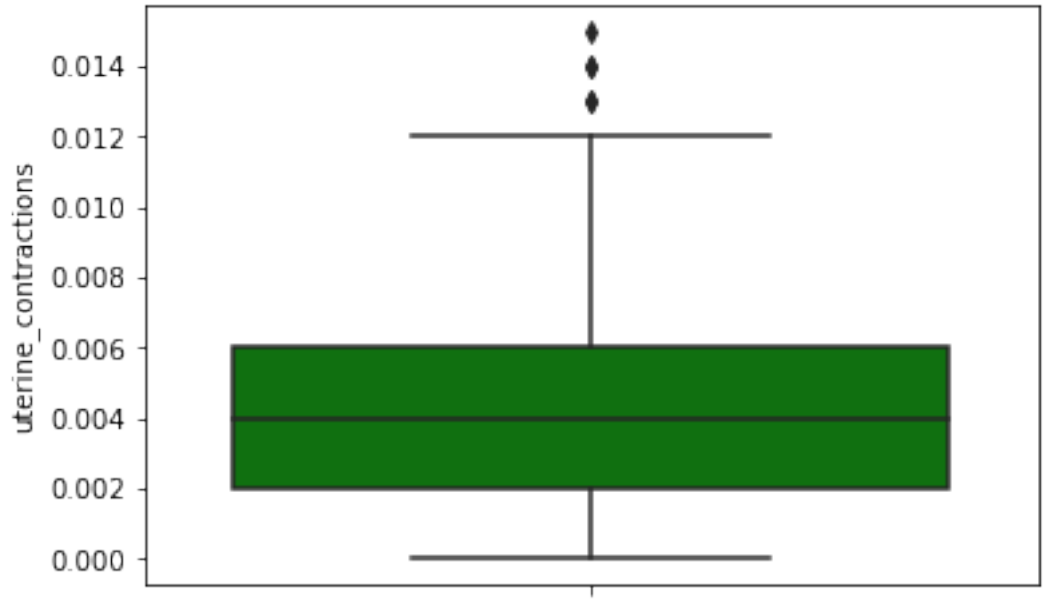
Finding the outlier using boxplot

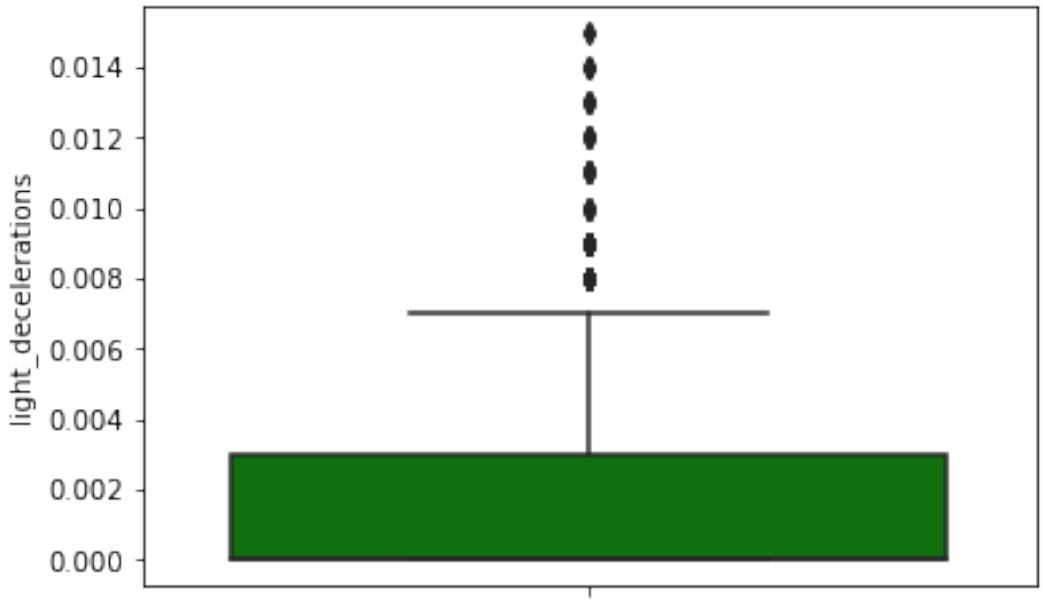
```
for i in range(0, len(nums )):
    sns.boxplot(y=df[nums[i]],color='green',orient='v')
    plt.show()
```

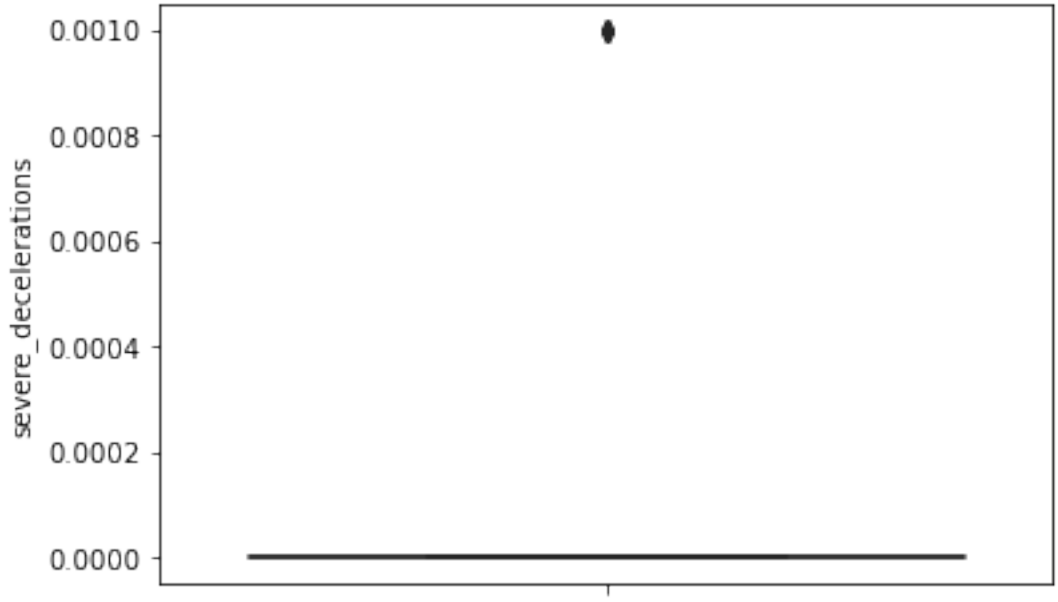

baseline value
160
150
140
130
120
110

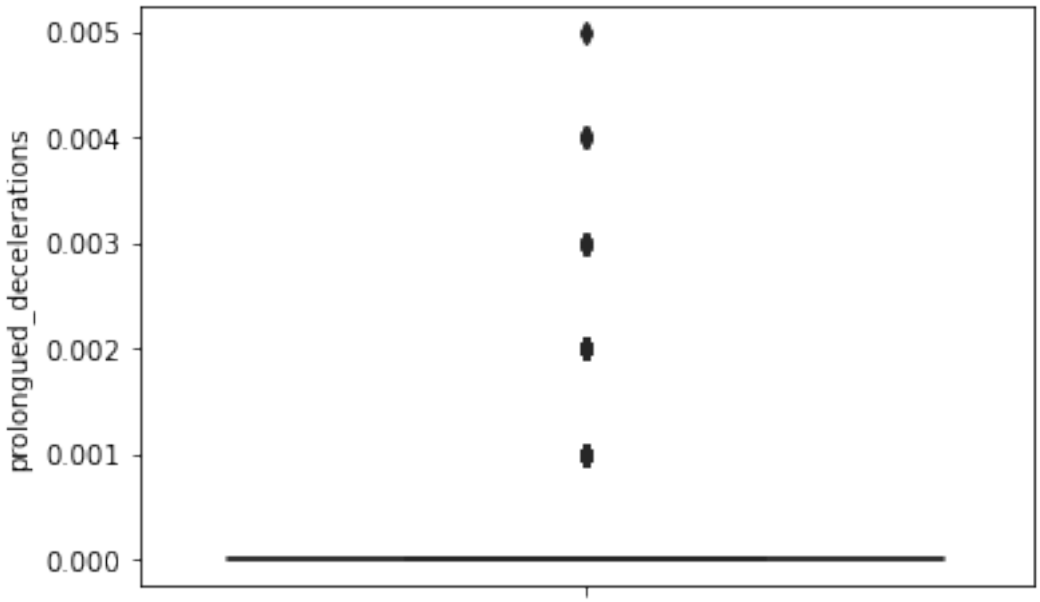

accelerations
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000

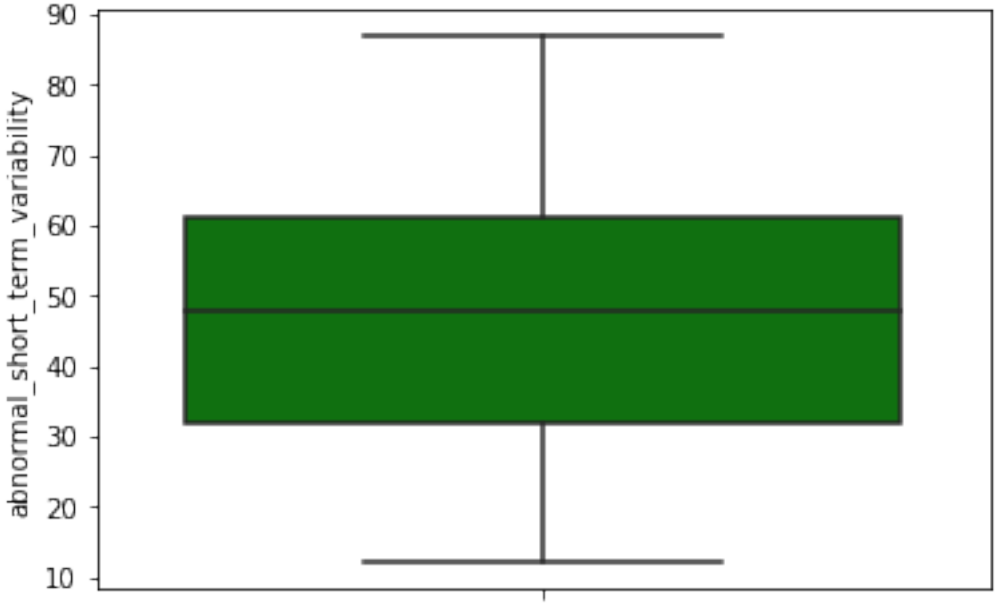

fetal_movement
0.5
0.4
0.3
0.2
0.1
0.0

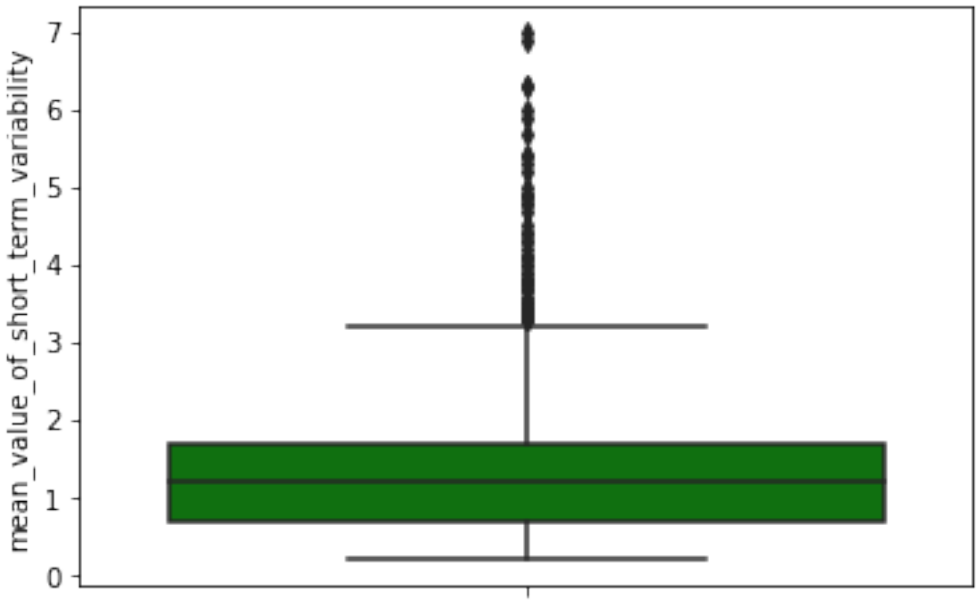

uterine_contractions
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000

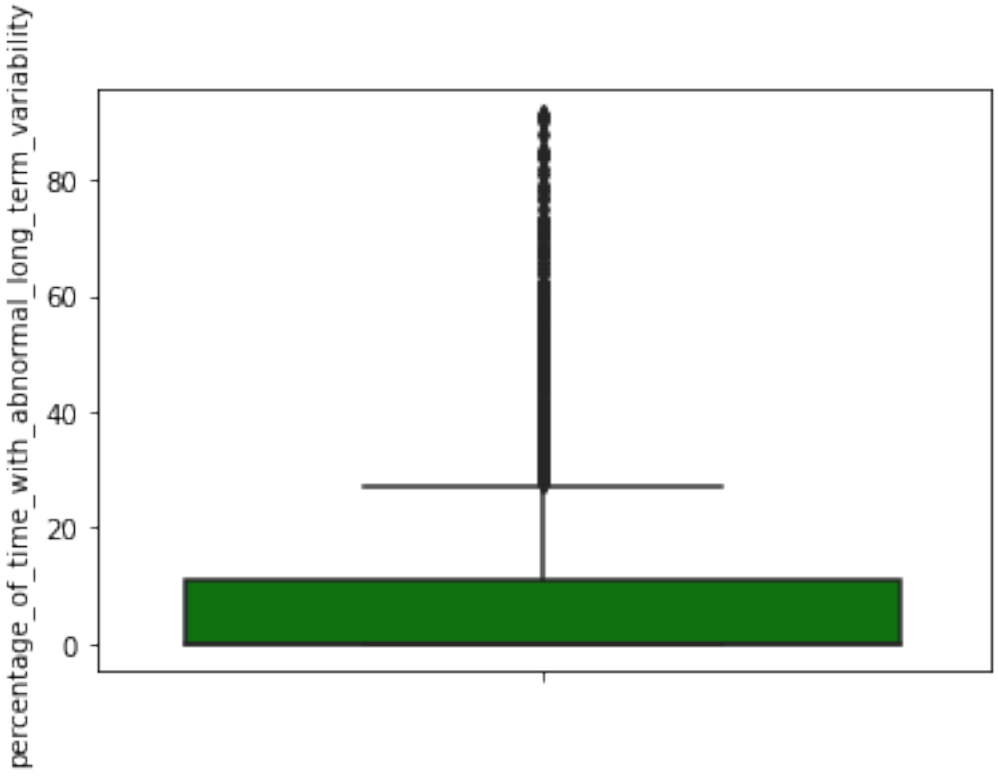

light_deceleration
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000

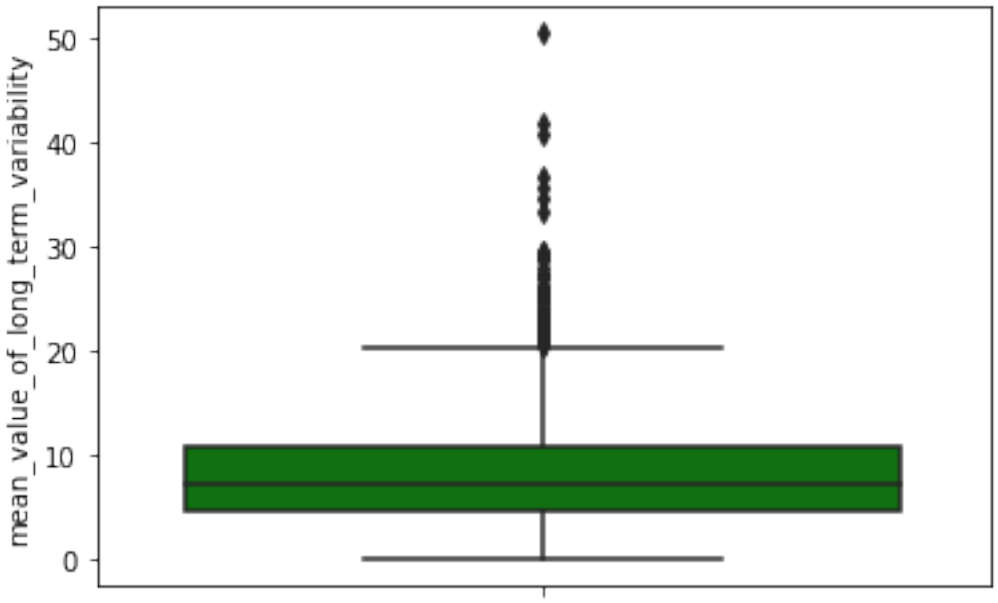

severe_decelerations
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000

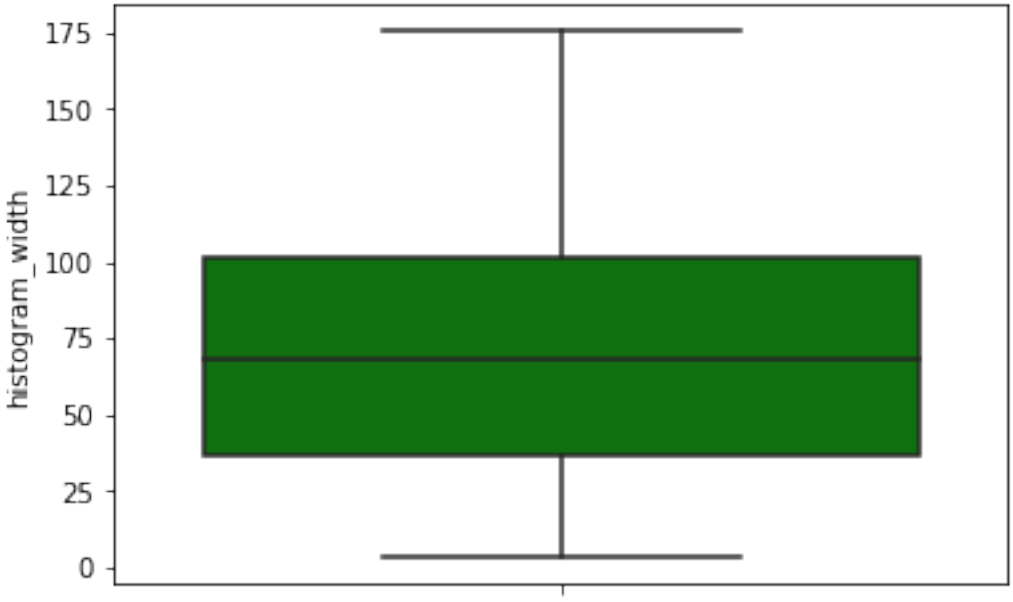

prolongued_decelerations
0.005
0.004
0.003
0.002
0.001
0.000

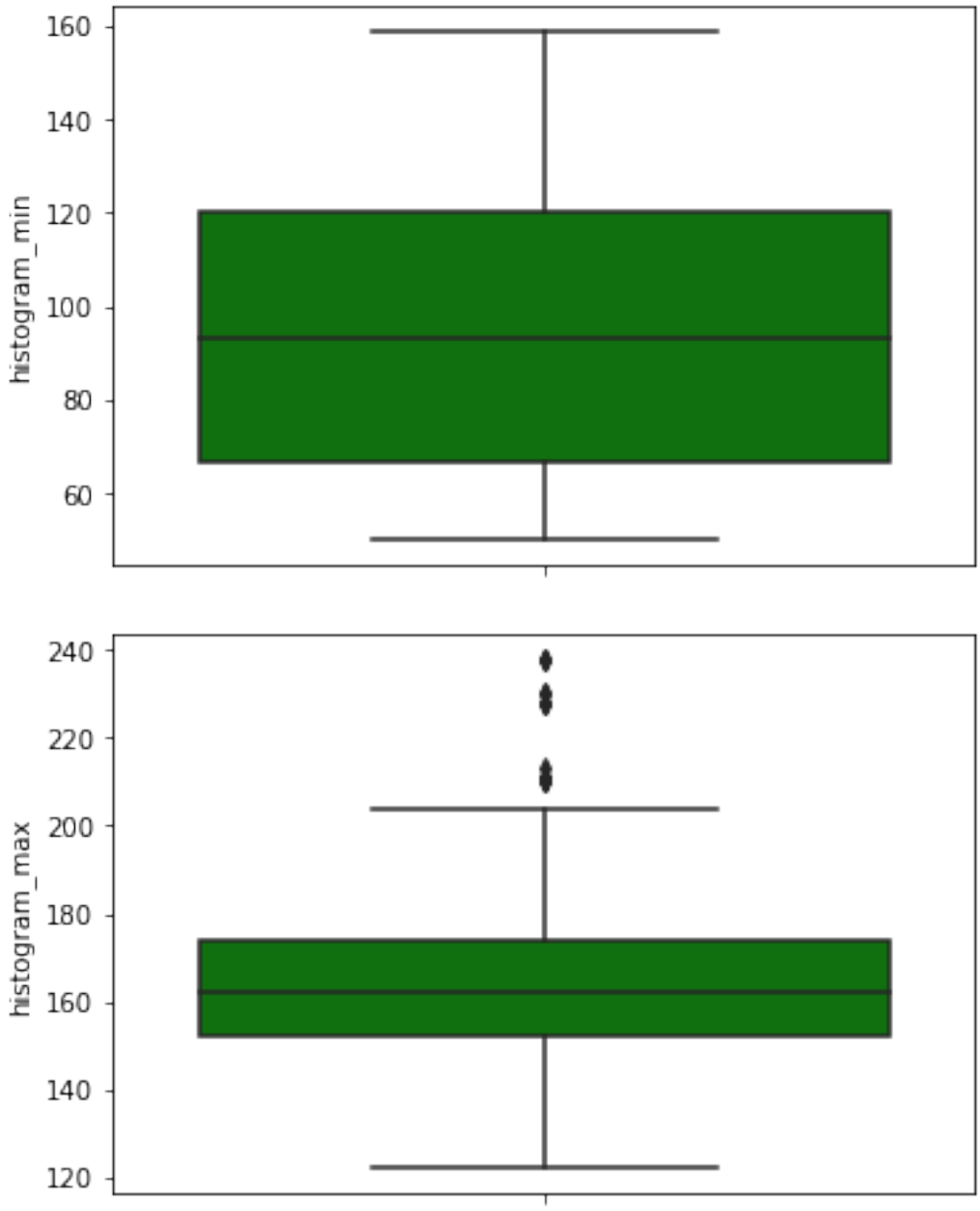

abnormal_short_term_variability
90
80
70
60
50
40
30
20
10

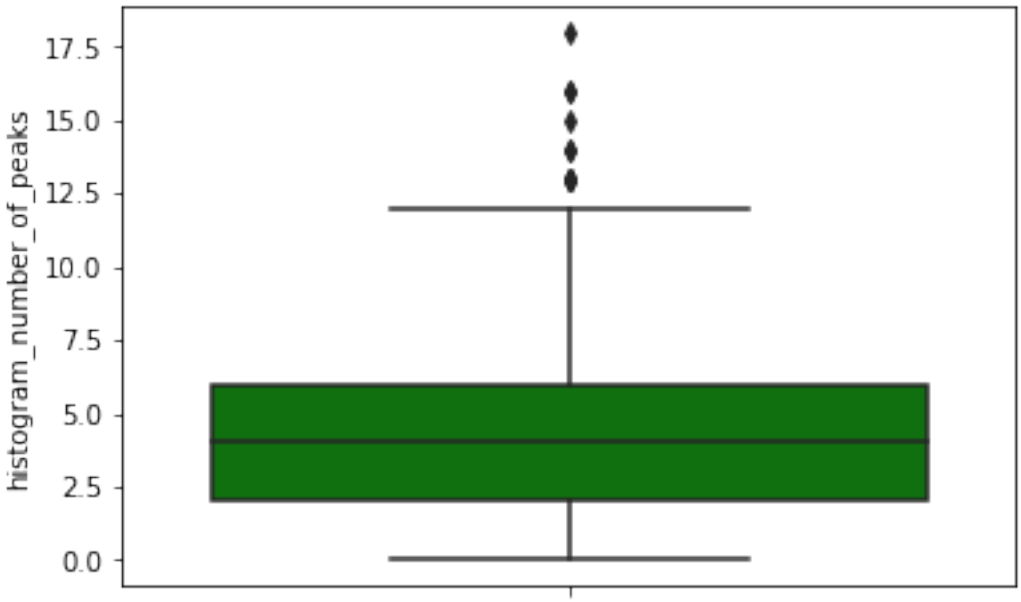

mean_value_of_short_term_variability
0
1
2
3
4
5
6
7


percentage_of_time_with_abnormal_long_term_variability
0
20
40
60
80

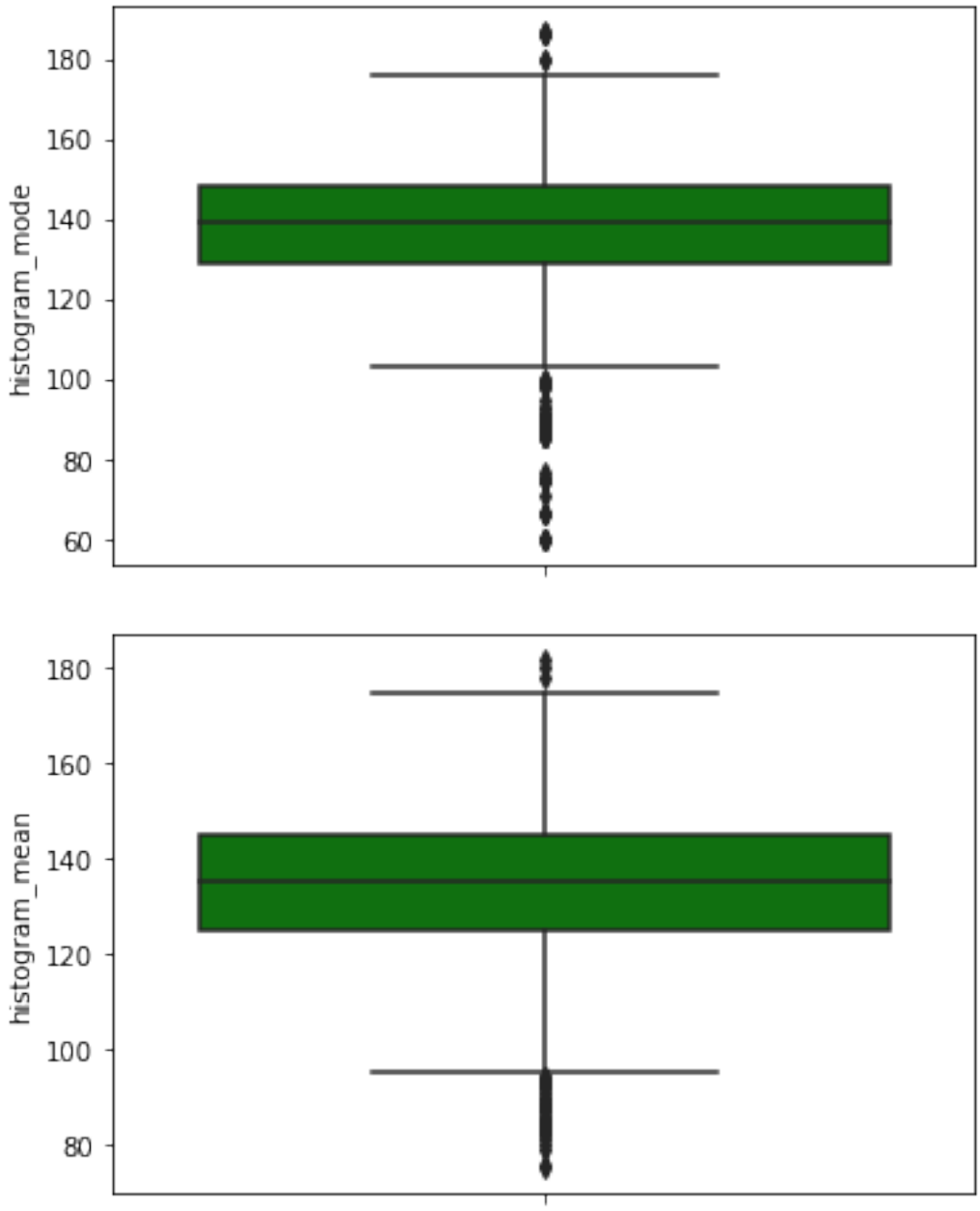

mean_value_of_long_term_variability
50
40
30
20
10
0

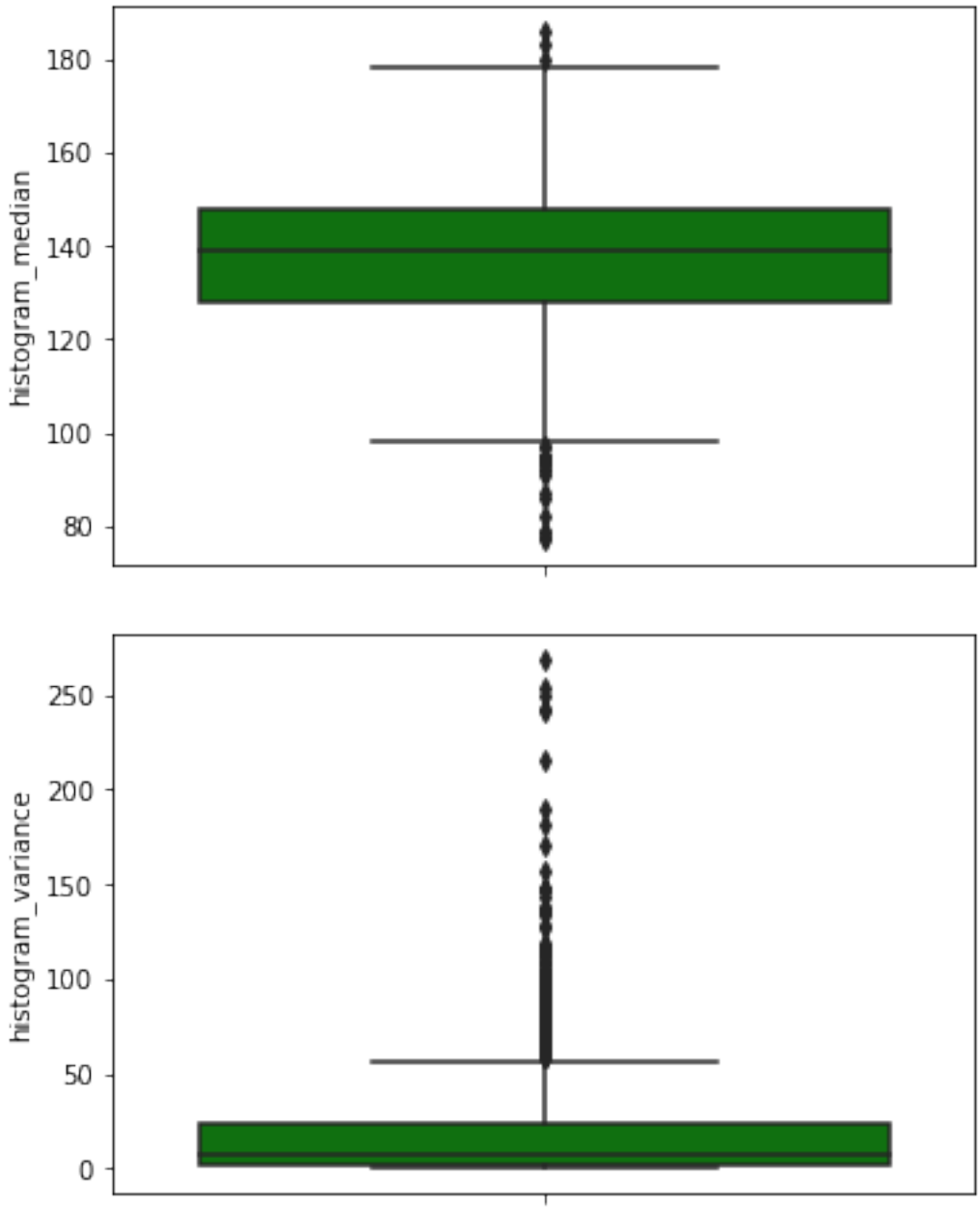

histogram_width
175
150
125
100
75
50
25
0

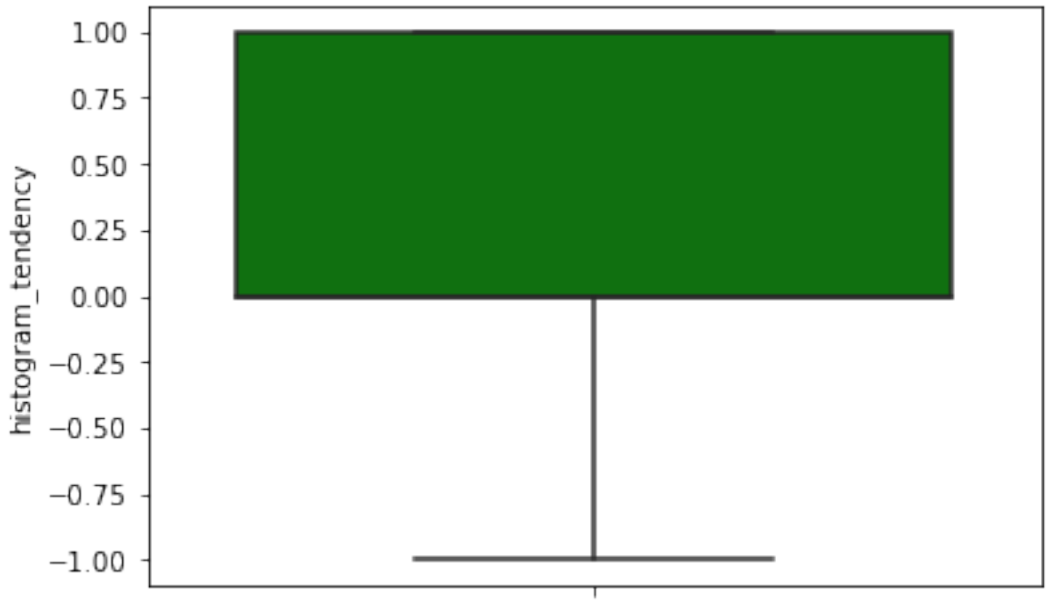

histogram_min
160
140
120
100
80
60
histogram_max
240
220
200
180
160
140
120

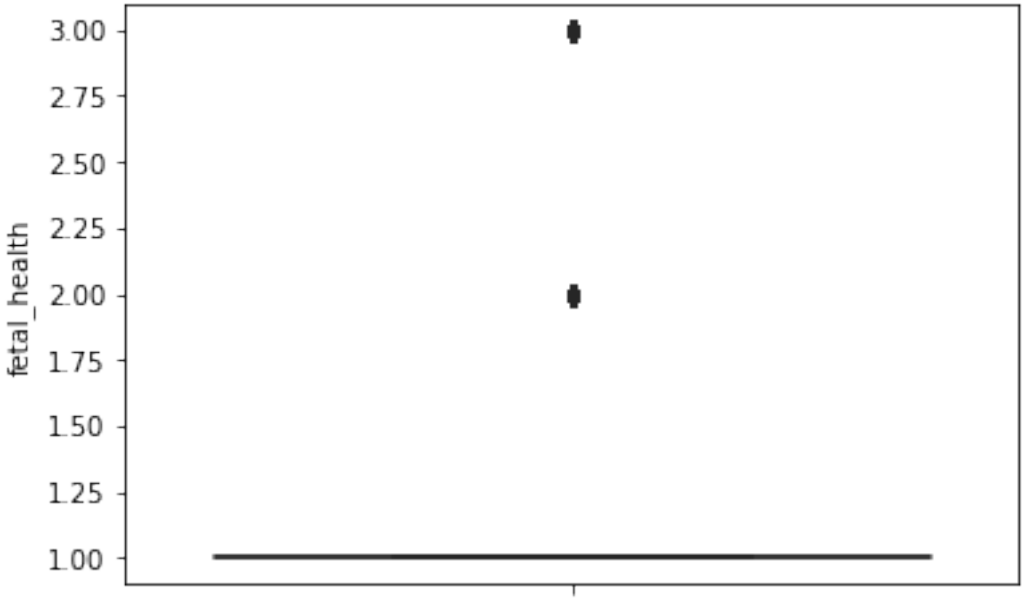

histogram_number_of_peaks
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


histogram_number_of_zeroes
10
8
6
4
2
0


histogram_mode
180
160
140
120
100
80
60
histogram_mean
180
160
140
120
100
80


histogram_median
180
160
140
120
100
80
histogram_variance
250
200
150
100
50
0

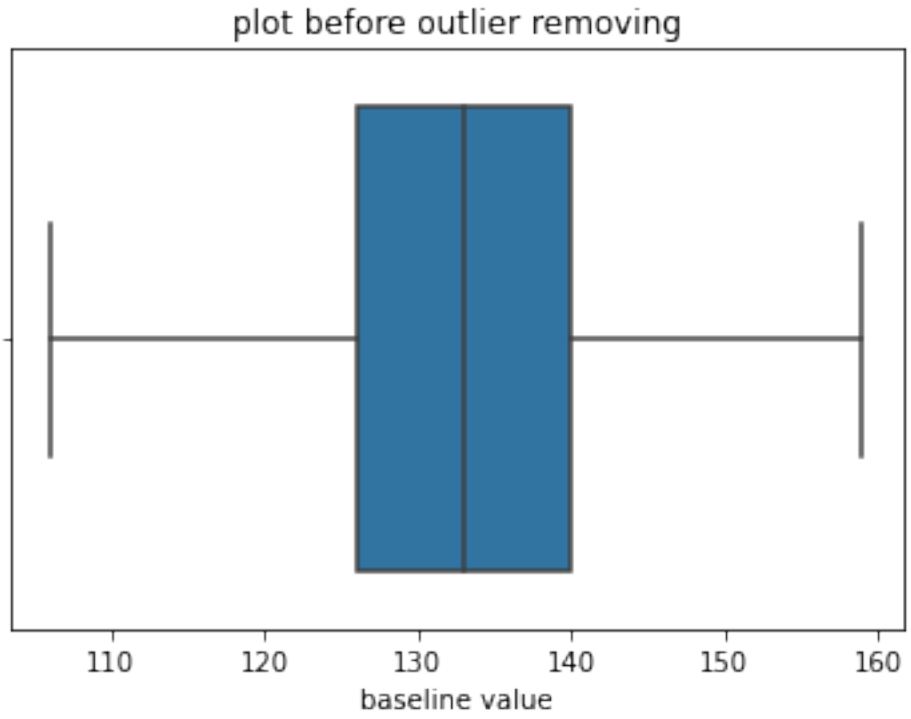


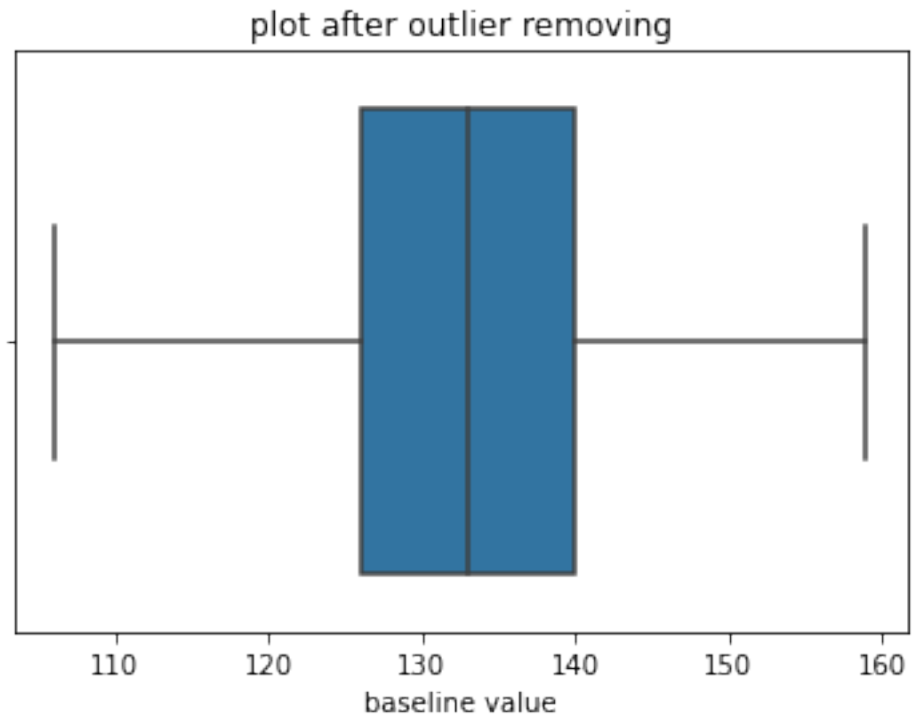


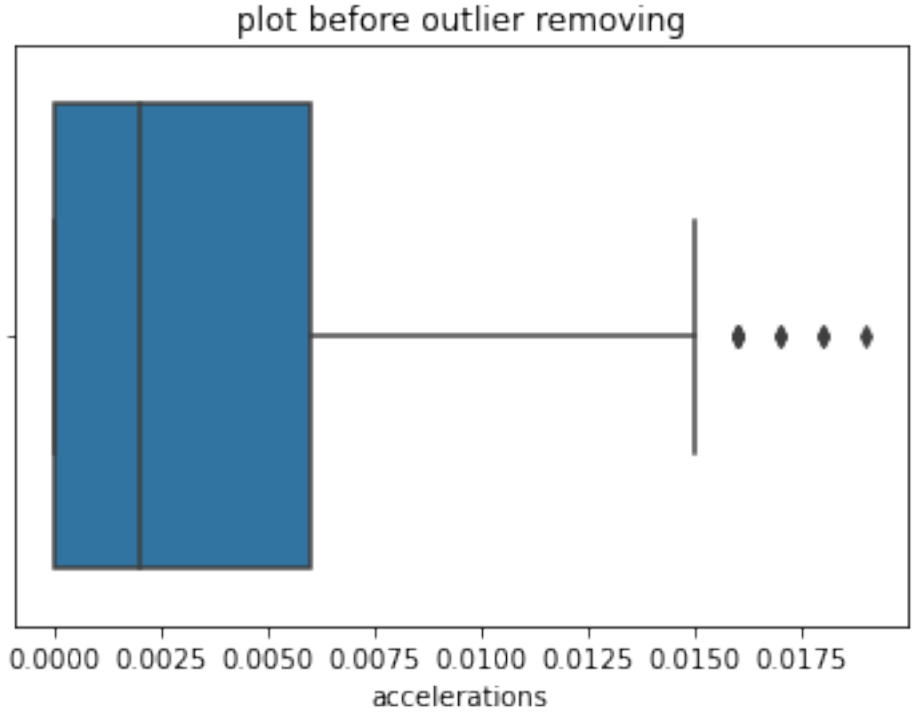
Removing the outliers

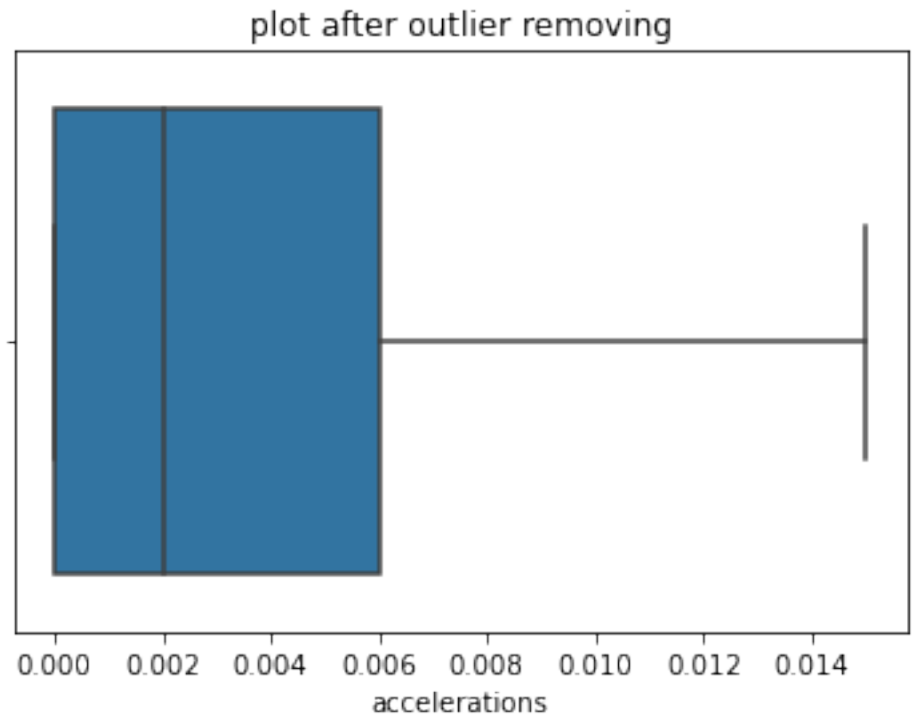
```
for i in range(len(nums)):
    sns.boxplot(df[nums[i]])
    plt.title(nums[i])
    plt.title("plot before outlier removing")
    plt.show()
    def drop_outliers(df, field_name):
        iqr = 1.5 * (np.percentile(df[field_name], 75) -
np.percentile(df[field_name], 25))
        df.drop(df[df[field_name] > (iqr +
np.percentile(df[field_name], 75))].index, inplace=True)
```

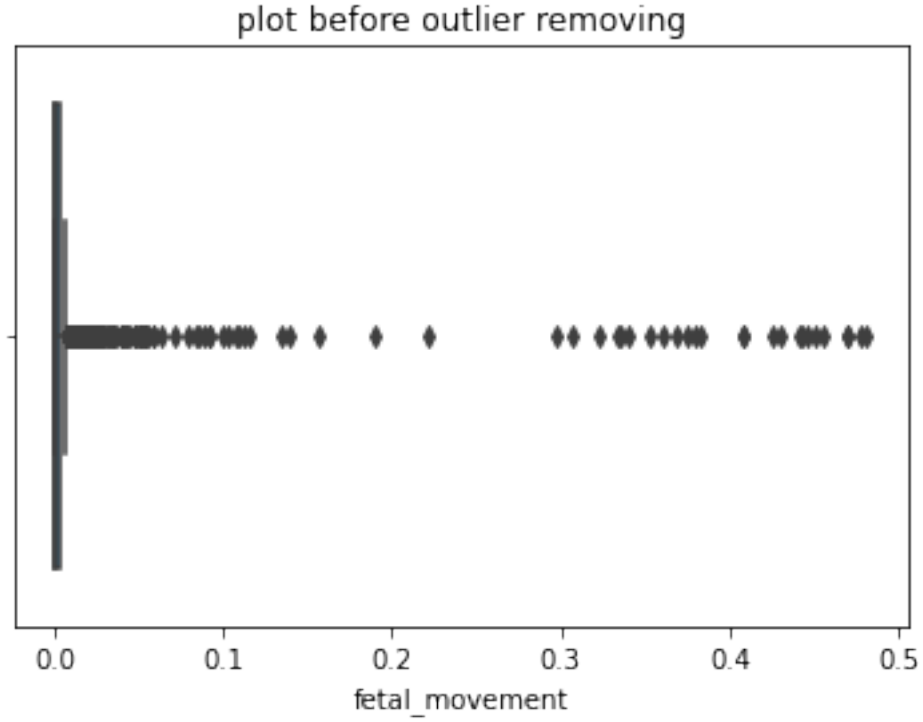
```
        df.drop(df[df[field_name] < (np.percentile(df[field_name], 25)
- iqr)].index, inplace=True)
        iqr = 1.5 * (np.percentile(df[field_name], 75) -
np.percentile(df[field_name], 25))
        df.drop(df[df[field_name] > (iqr +
np.percentile(df[field_name], 75))].index, inplace=True)
        df.drop(df[df[field_name] < (np.percentile(df[field_name], 25)
- iqr)].index, inplace=True)
    drop_outliers(df, nums[i])
    sns.boxplot(df[nums[i]])
    plt.title("plot after outlier removing")
    plt.show()
```

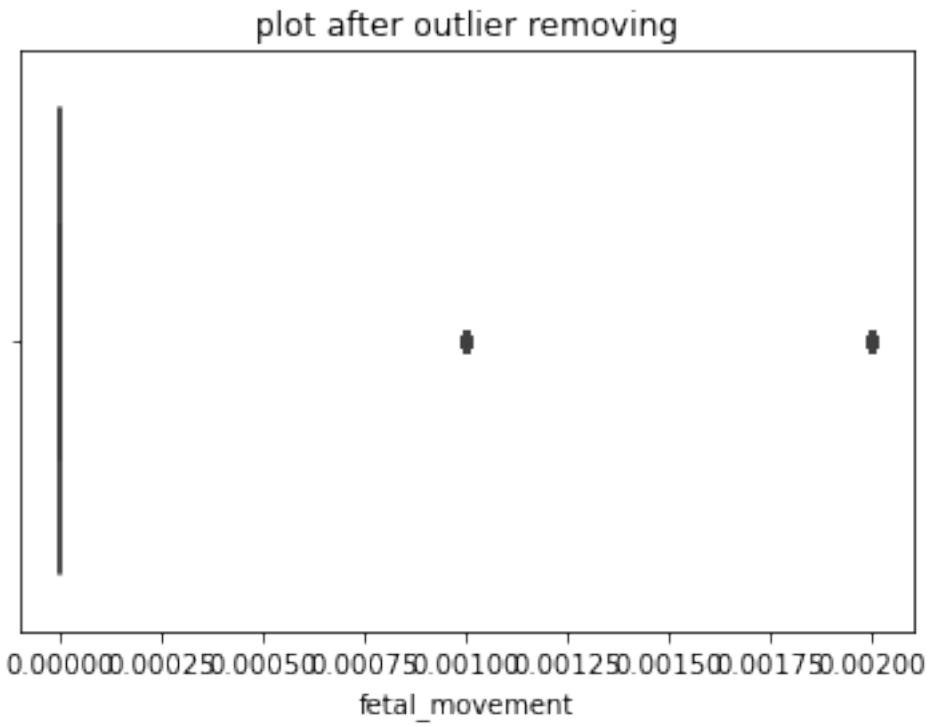


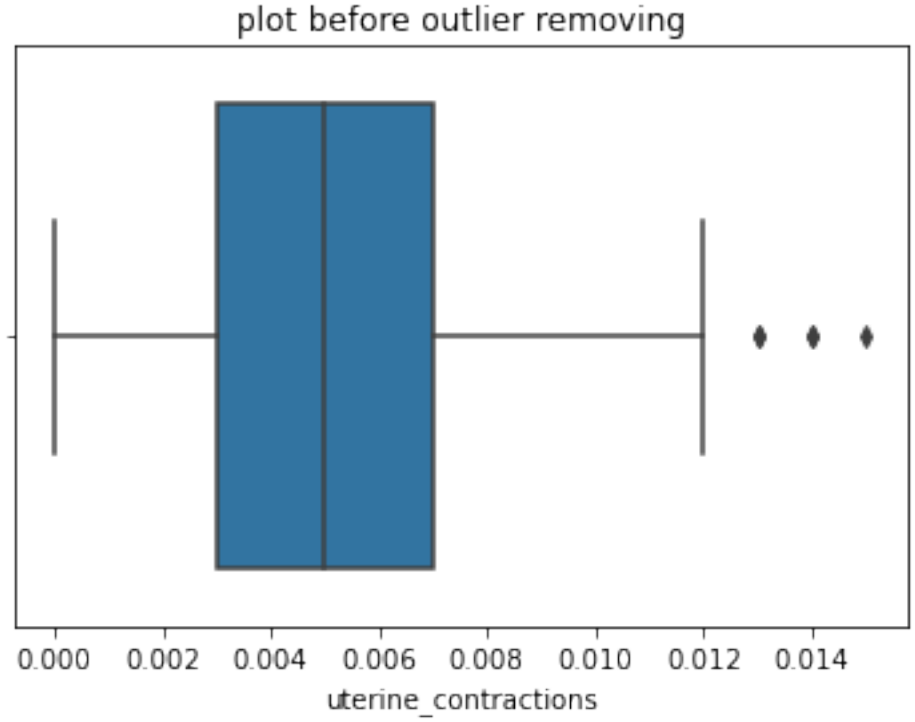

plot after outlier removing
110
120
130
140
150
160
baseline value

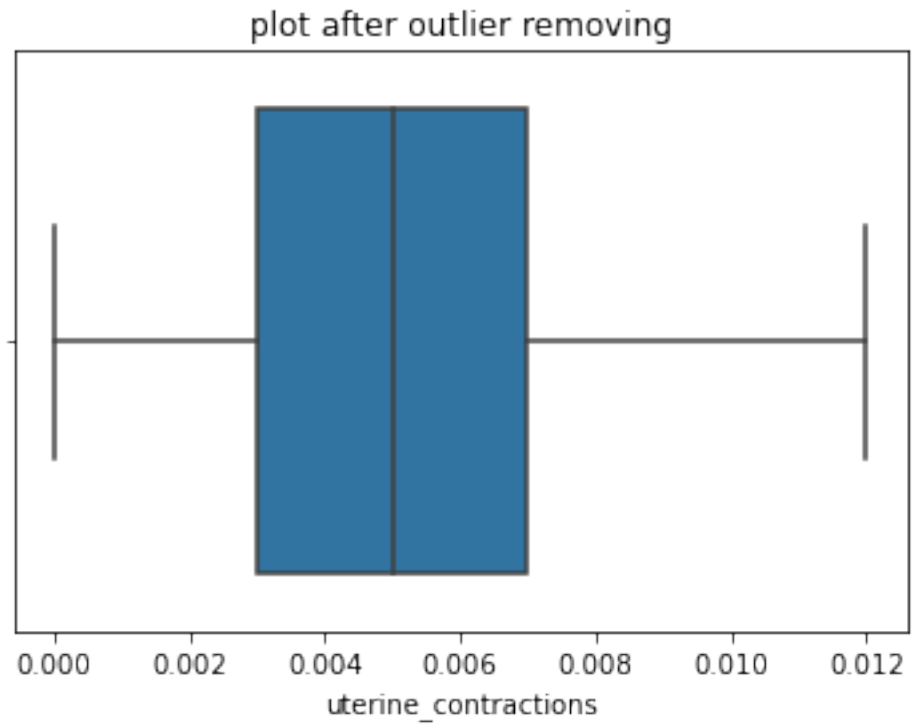

plot before outlier removing
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
accelerations

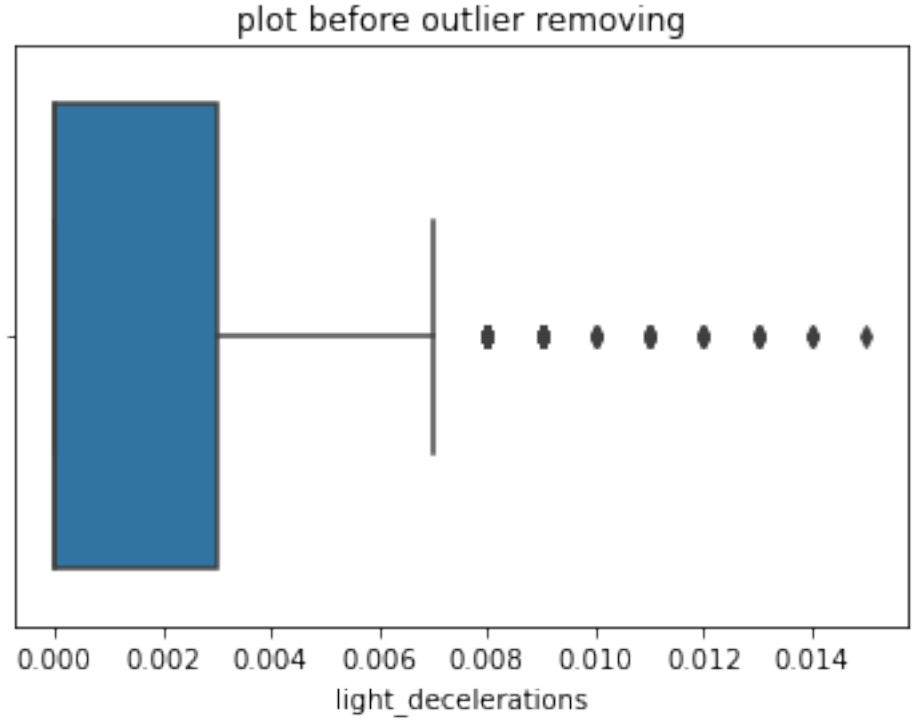

plot after outlier removing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
accelerations

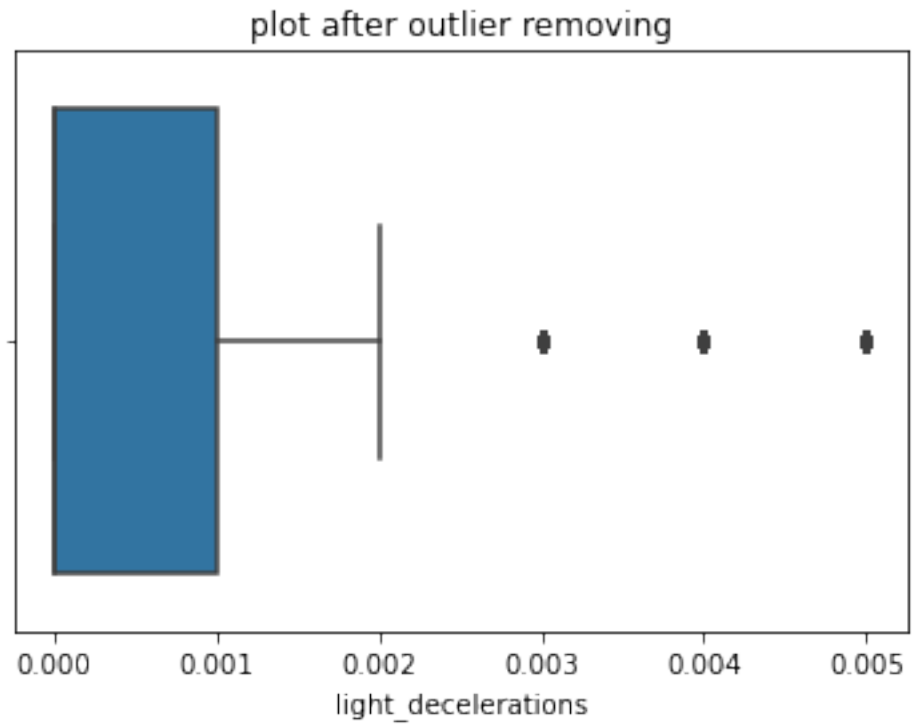

plot before outlier removing
0.0
0.1
0.2
0.3
0.4
0.5
fetal_movement

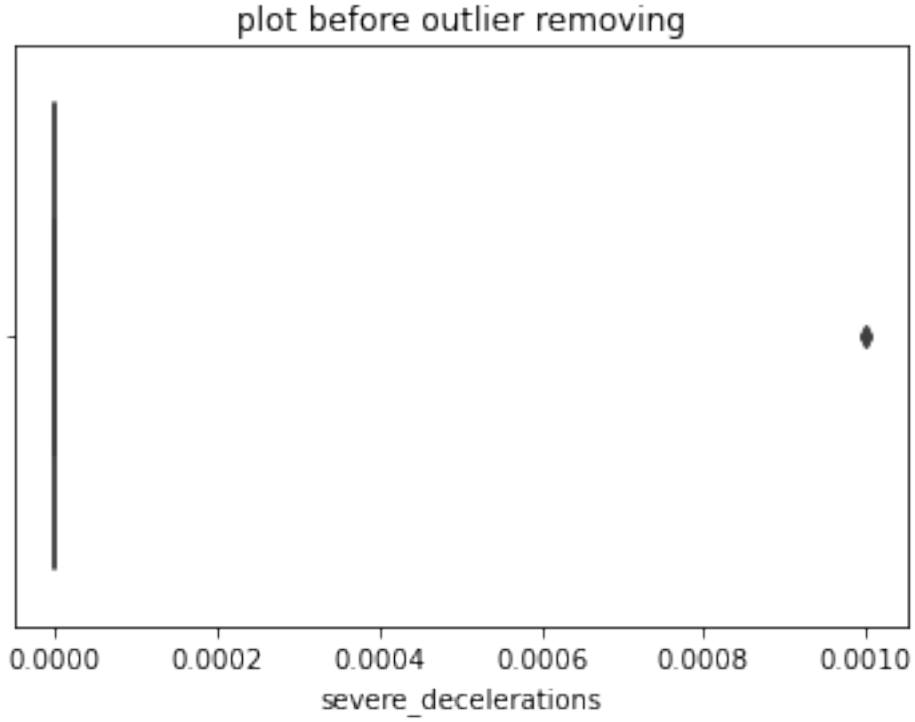

plot after outlier removing
0.00000
0.00025
0.00050
0.00075
0.00100
0.00125
0.00150
0.00175
0.00200
fetal_movement

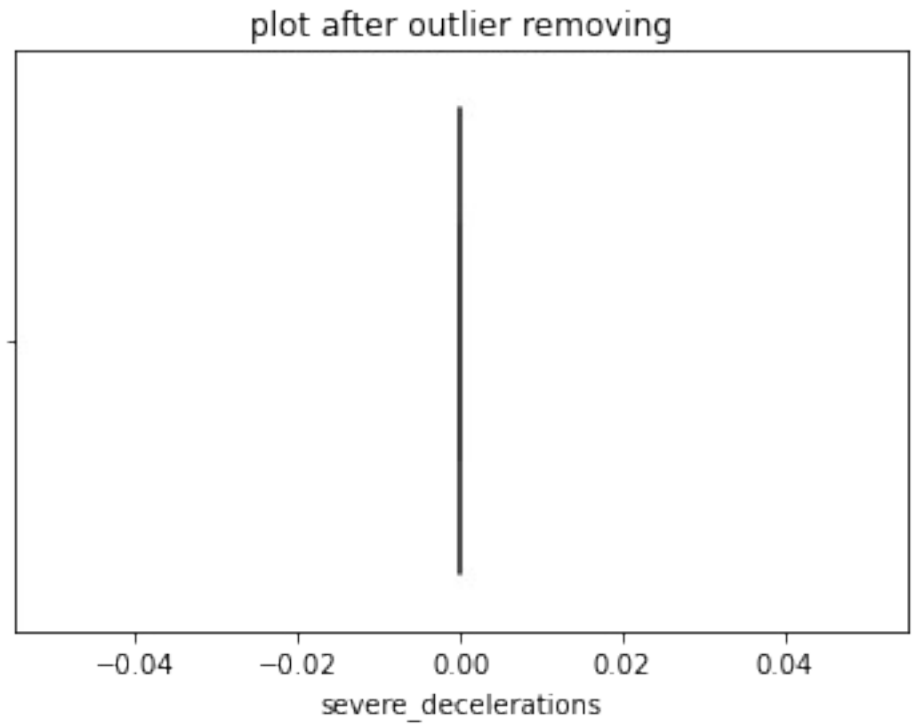

plot before outlier removing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
uterine_contractions

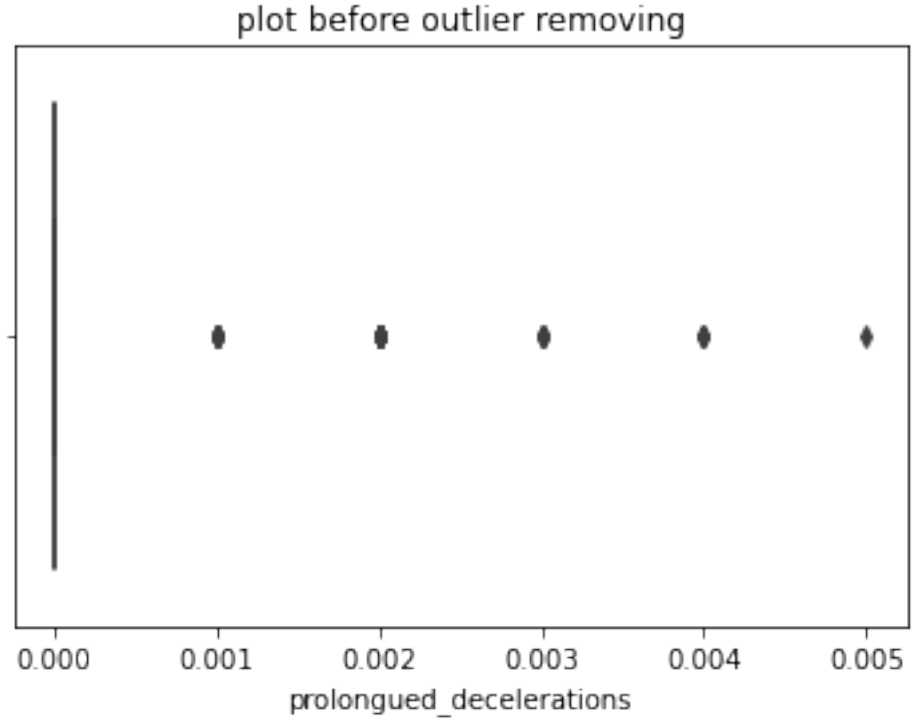

plot after outlier removing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
uterine_contractions

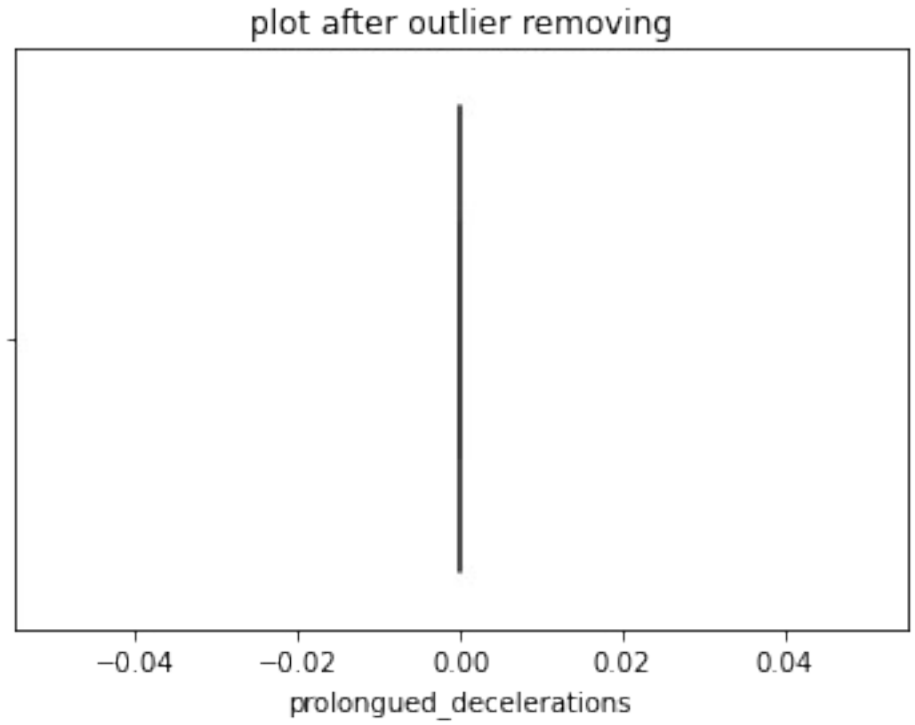

plot before outlier removing
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
light_decelerations

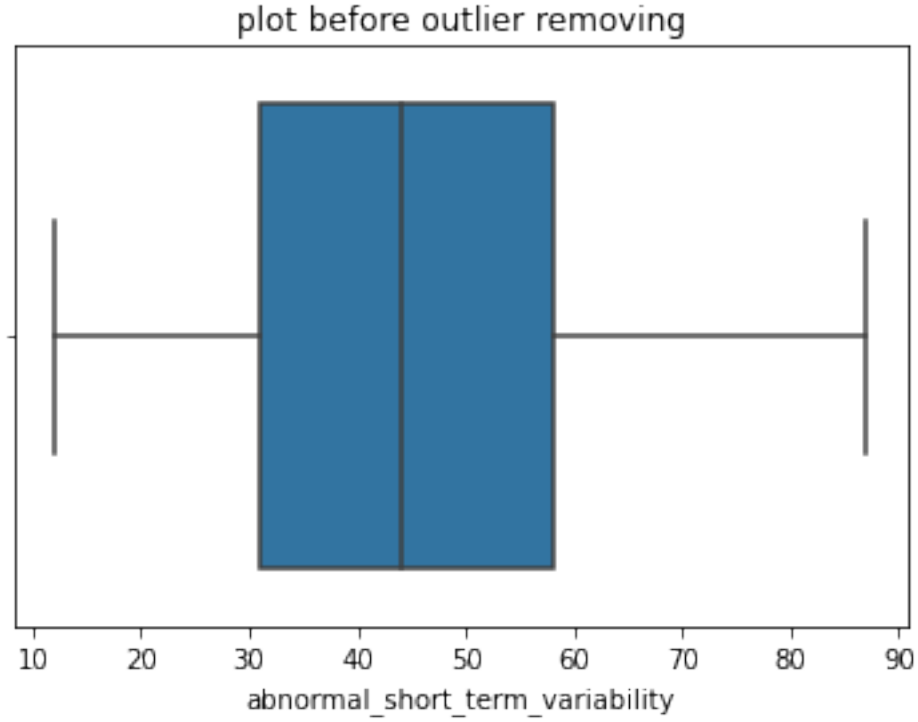

plot after outlier removing
0.000
0.001
0.002
0.003
0.004
0.005
light_decelerations

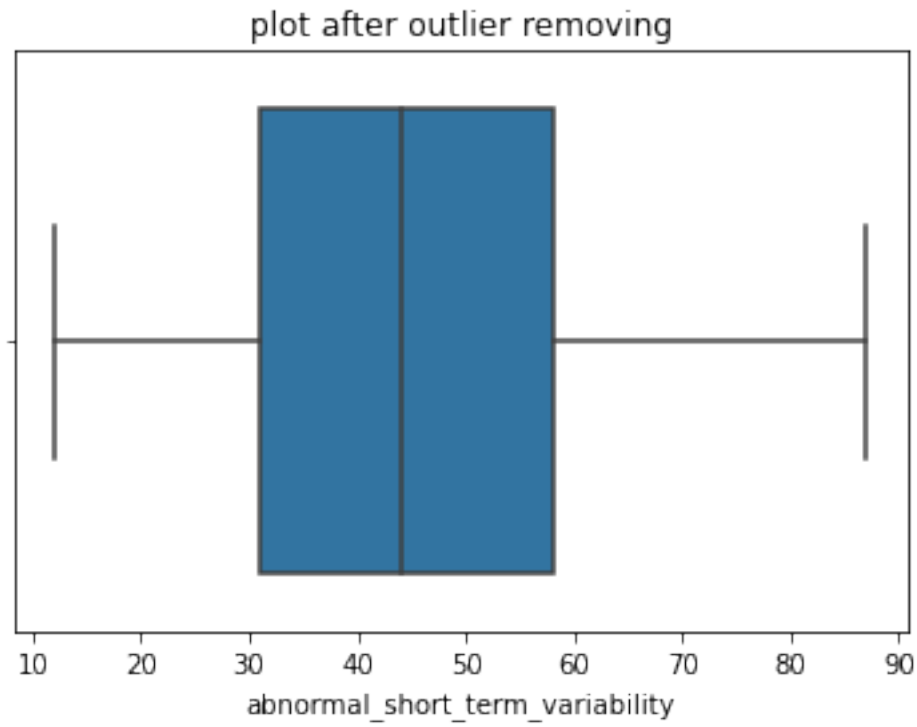

plot before outlier removing
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
severe_decelerations

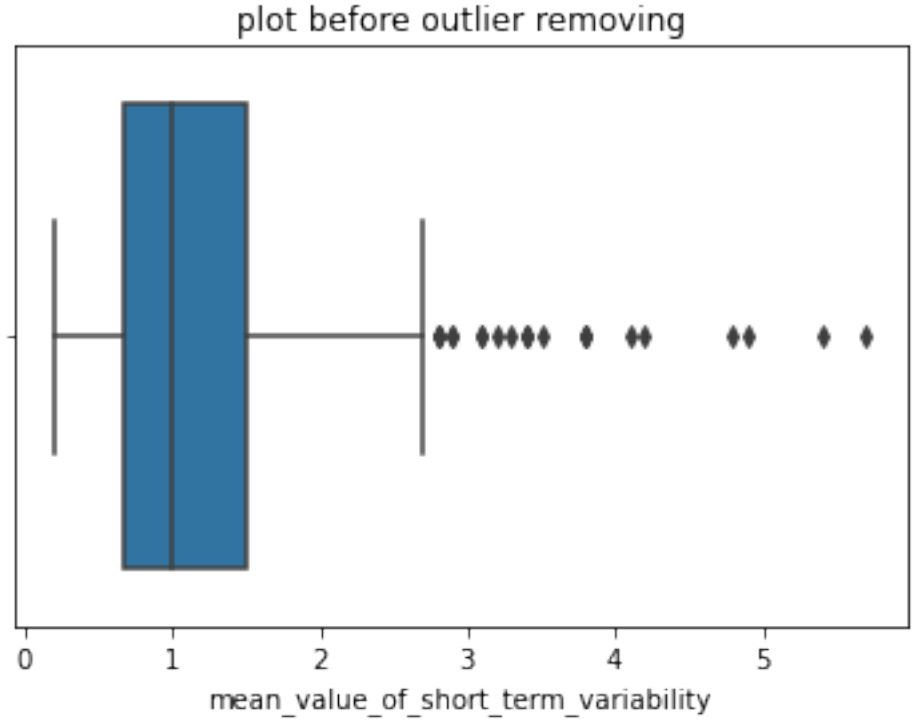

plot after outlier removing
−0.04
−0.02
0.00
0.02
0.04
severe_decelerations

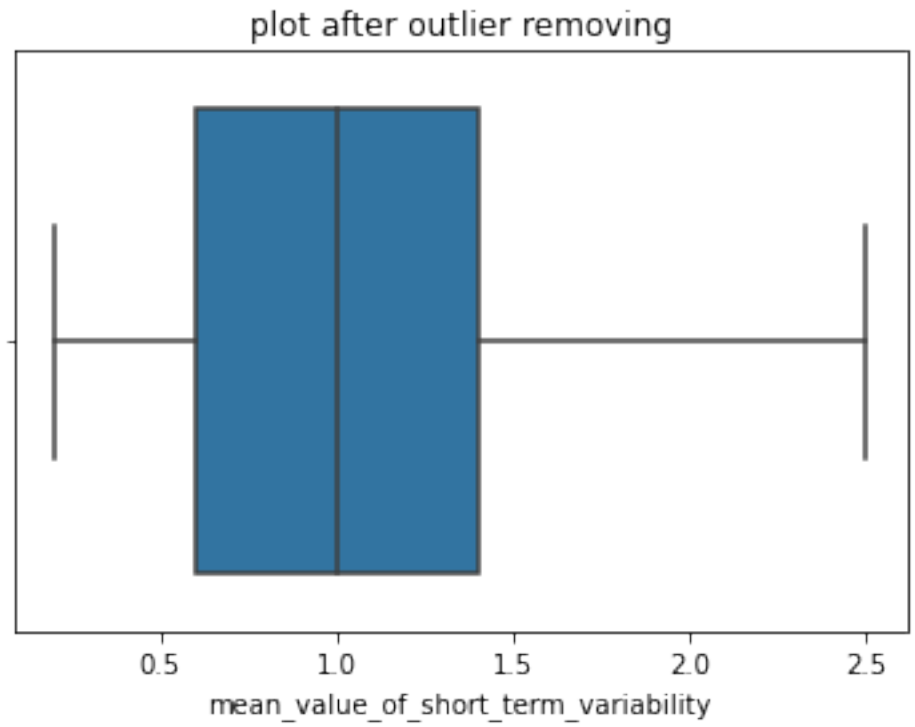

plot before outlier removing
0.000
0.001
0.002
0.003
0.004
0.005
prolongued_decelerations

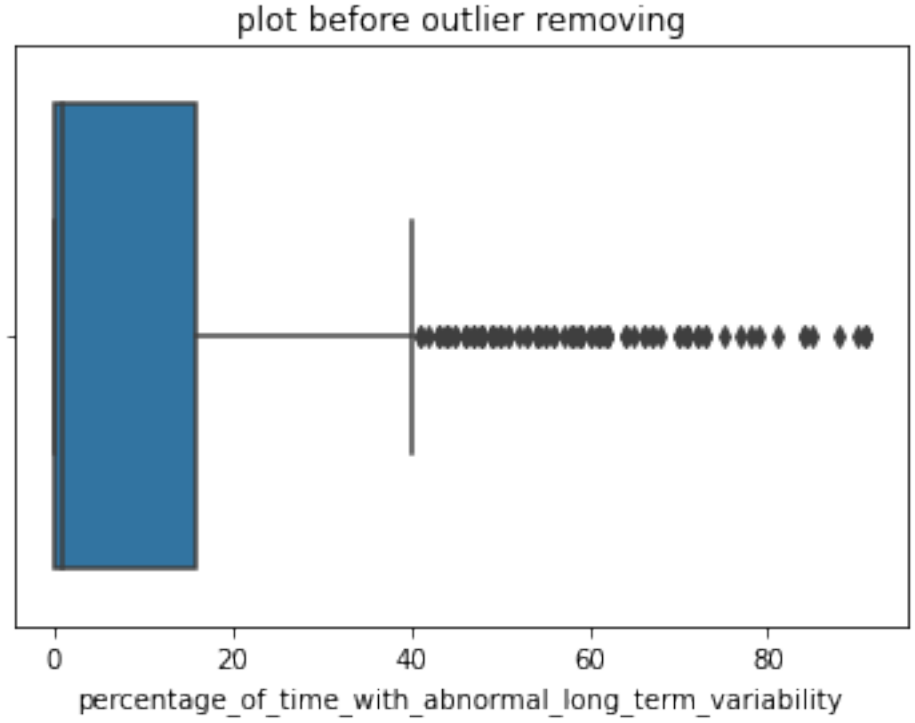

plot after outlier removing
−0.04
−0.02
0.00
0.02
0.04
prolongued_decelerations

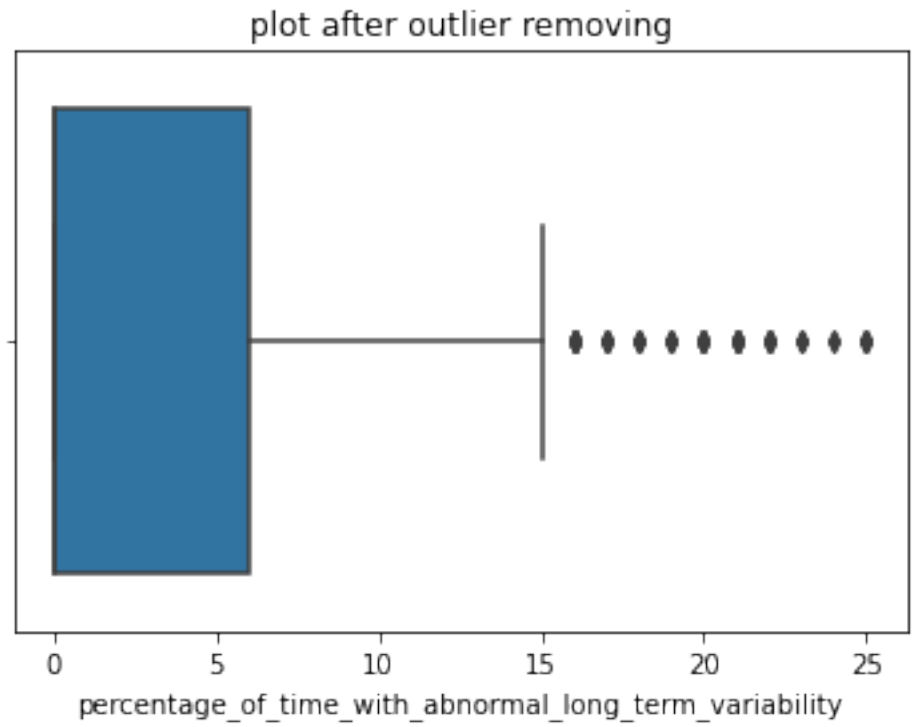

plot before outlier removing
10
20
30
40
50
60
70
80
90
abnormal_short_term_variability

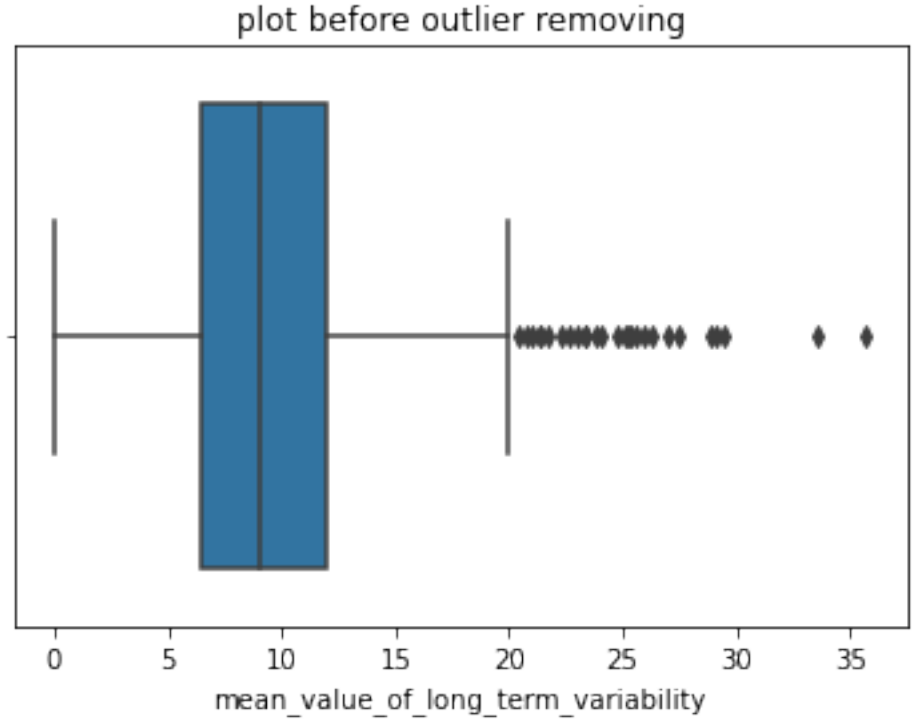

plot after outlier removing
10
20
30
40
50
60
70
80
90
abnormal_short_term_variability

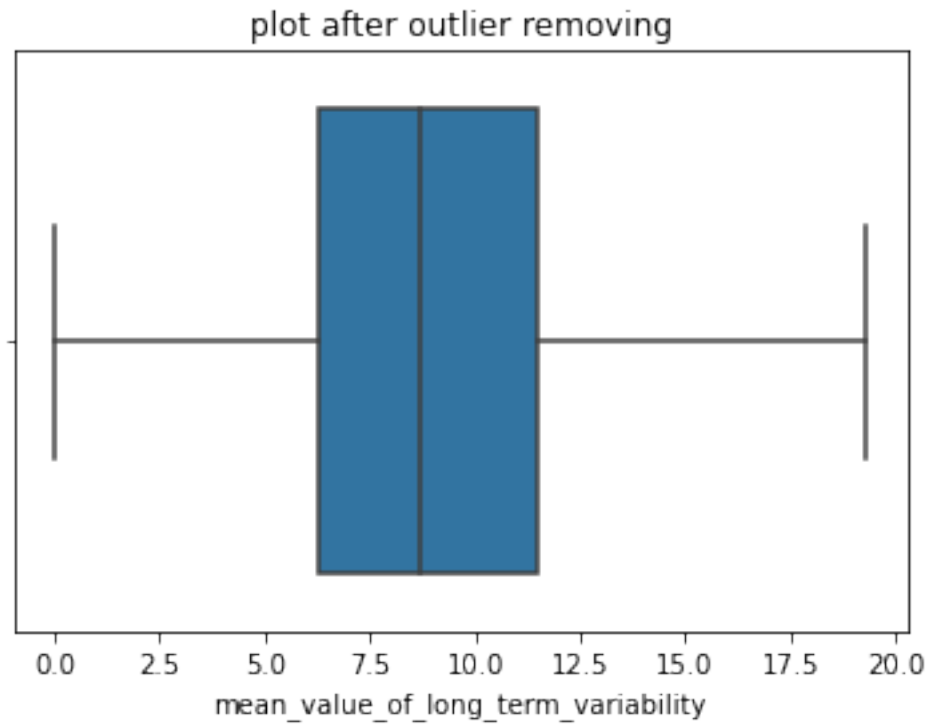

plot before outlier removing
0
1
2
3
4
5
mean_value_of_short_term_variability

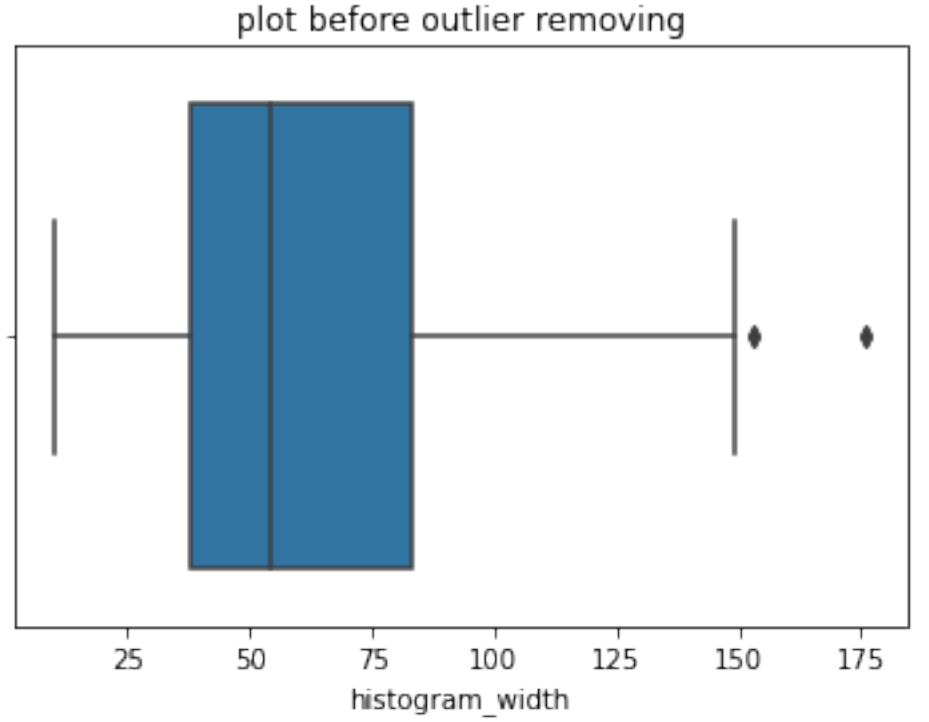

plot after outlier removing
0.5
1.0
1.5
2.0
2.5
mean_value_of_short_term_variability

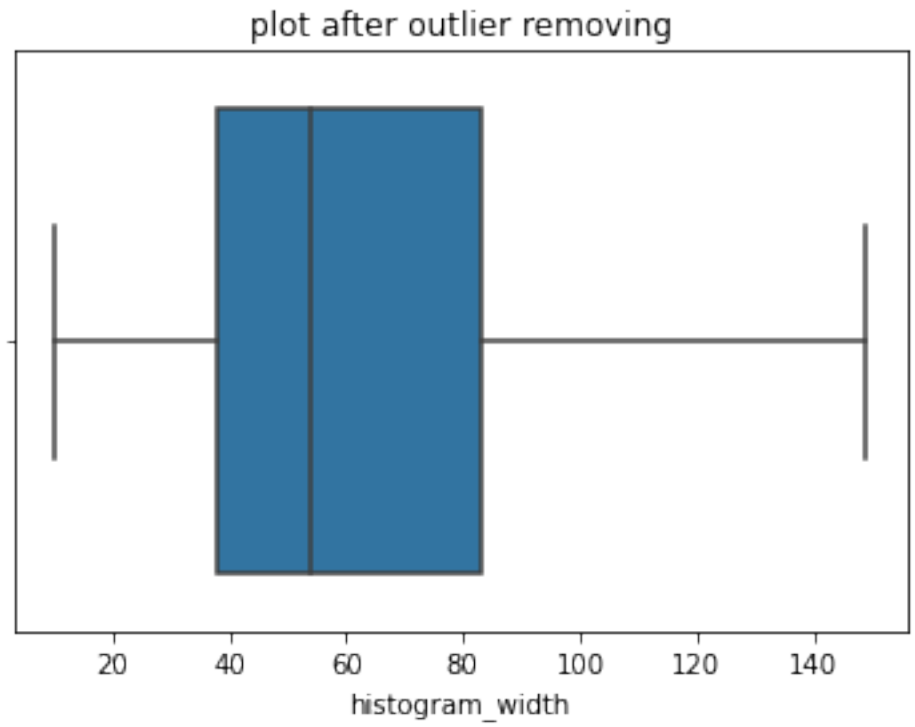

plot before outlier removing
0
20
40
60
80
percentage_of_time_with_abnormal_long_term_variability

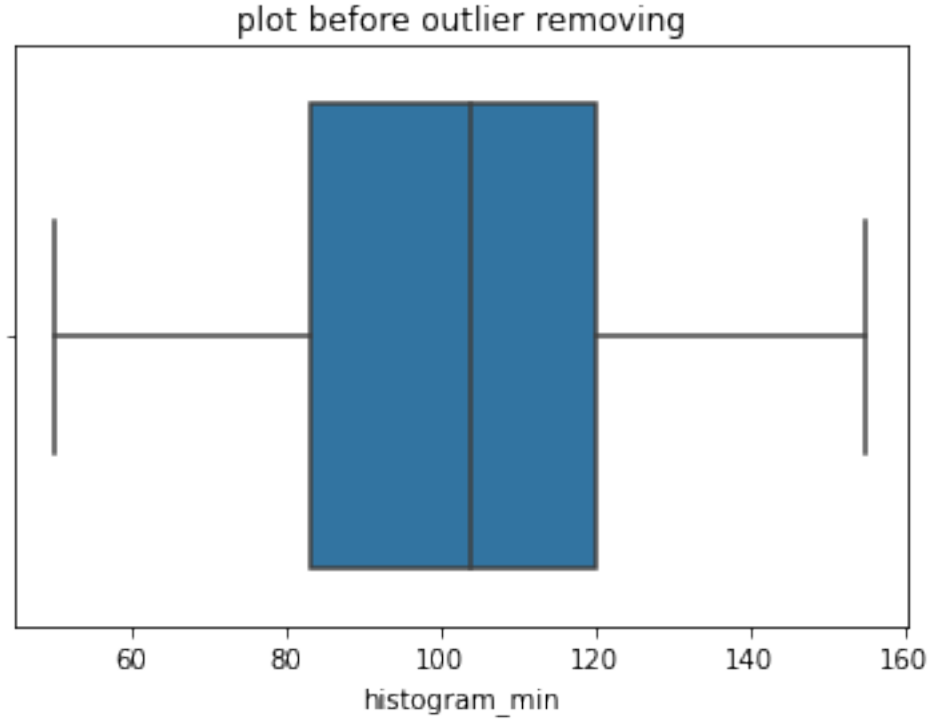

plot after outlier removing
0
5
10
15
20
25
percentage_of_time_with_abnormal_long_term_variability

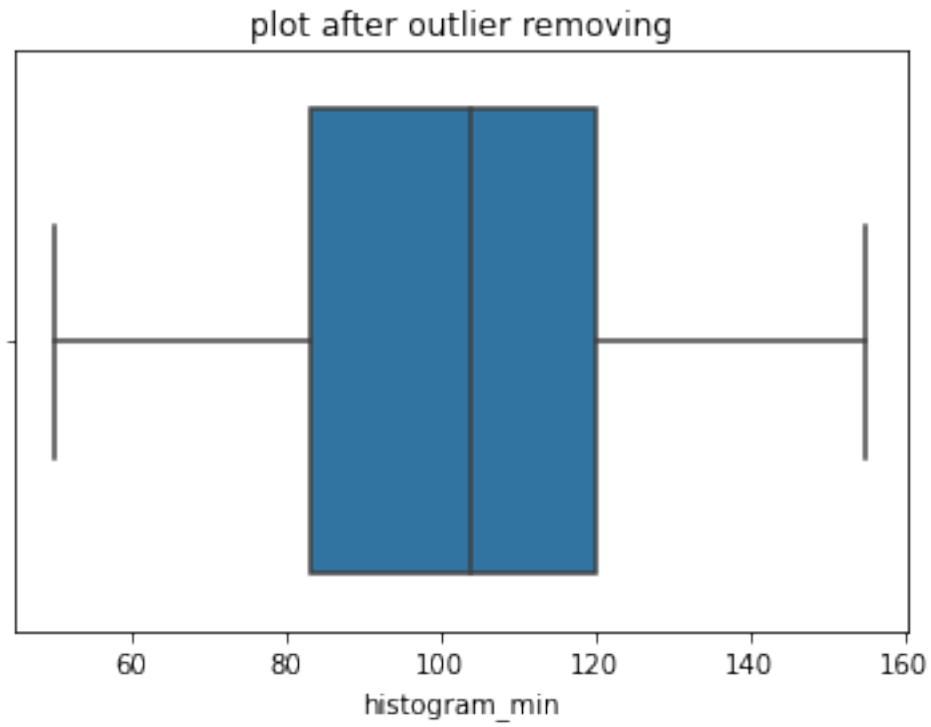

plot before outlier removing
0
5
10
15
20
25
30
35
mean_value_of_long_term_variability

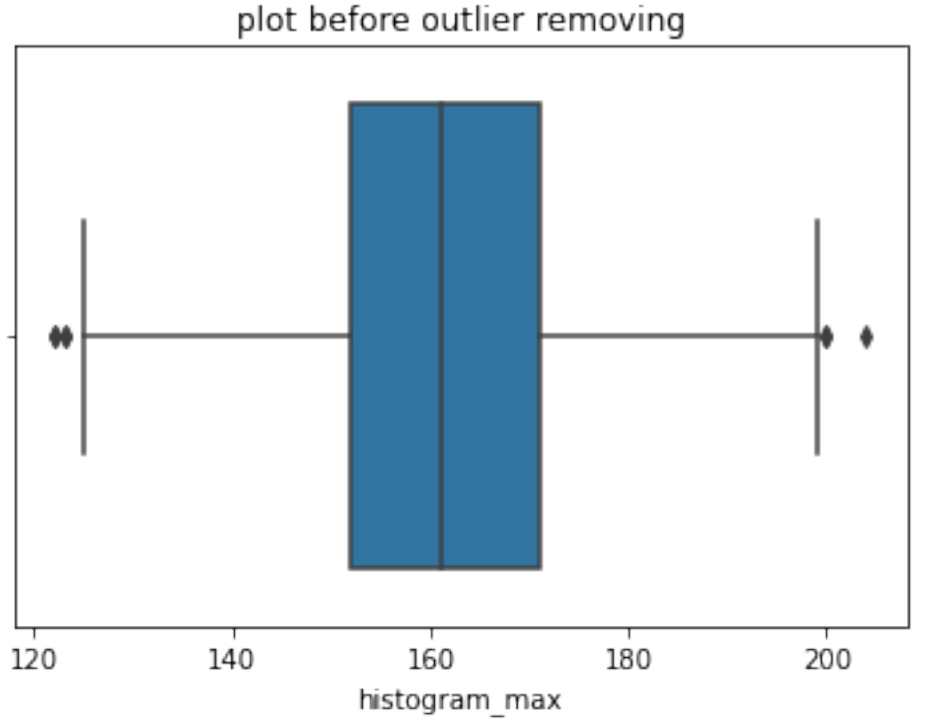

plot after outlier removing
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
mean_value_of_long_term_variability

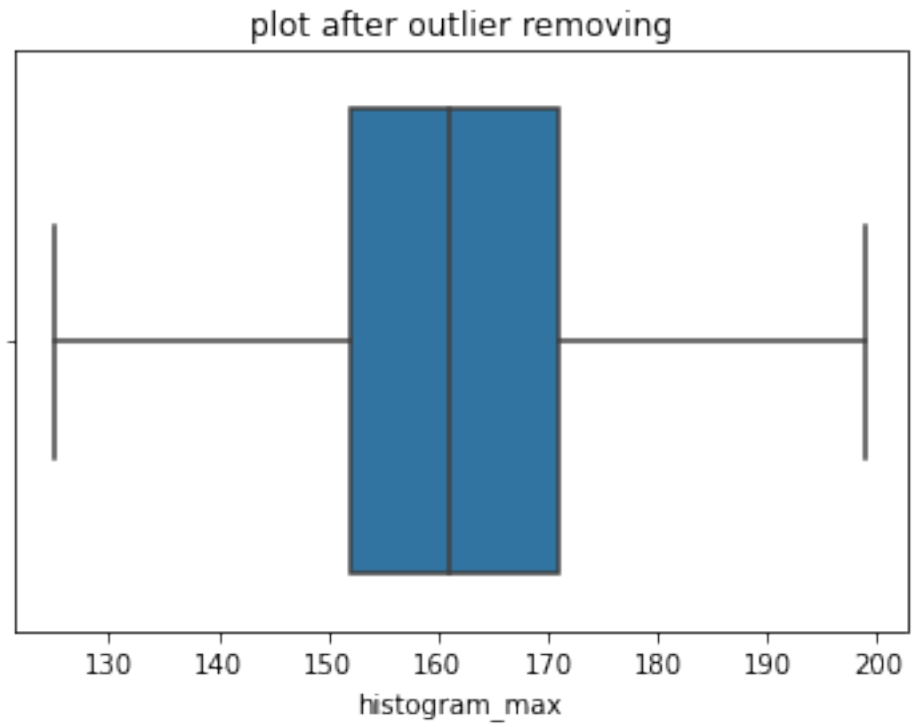

plot before outlier removing
25
50
75
100
125
150
175
histogram_width

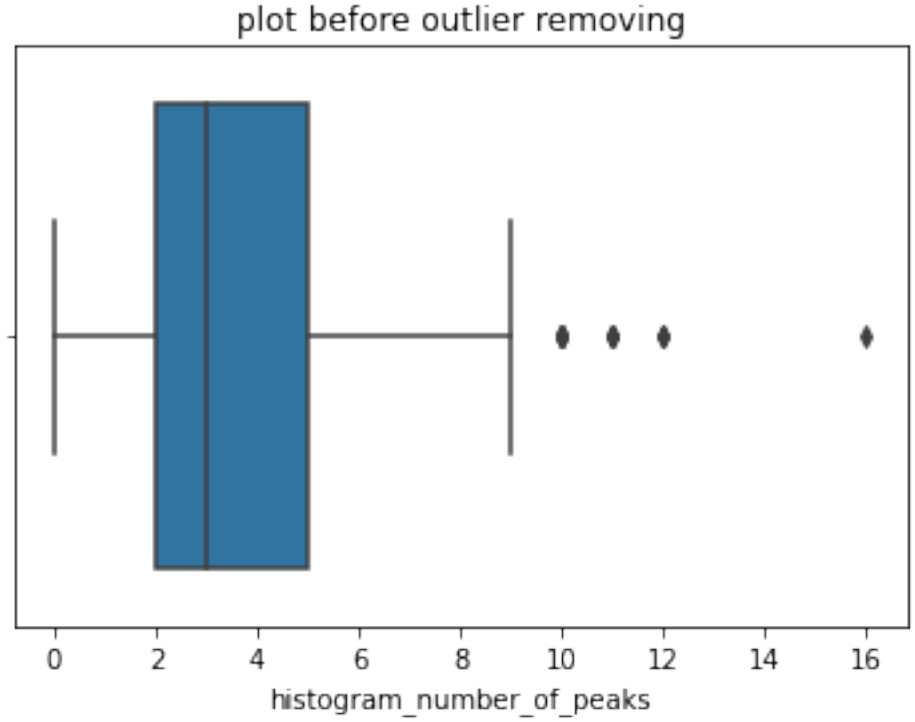

plot after outlier removing
20
40
60
80
100
120
140
histogram_width

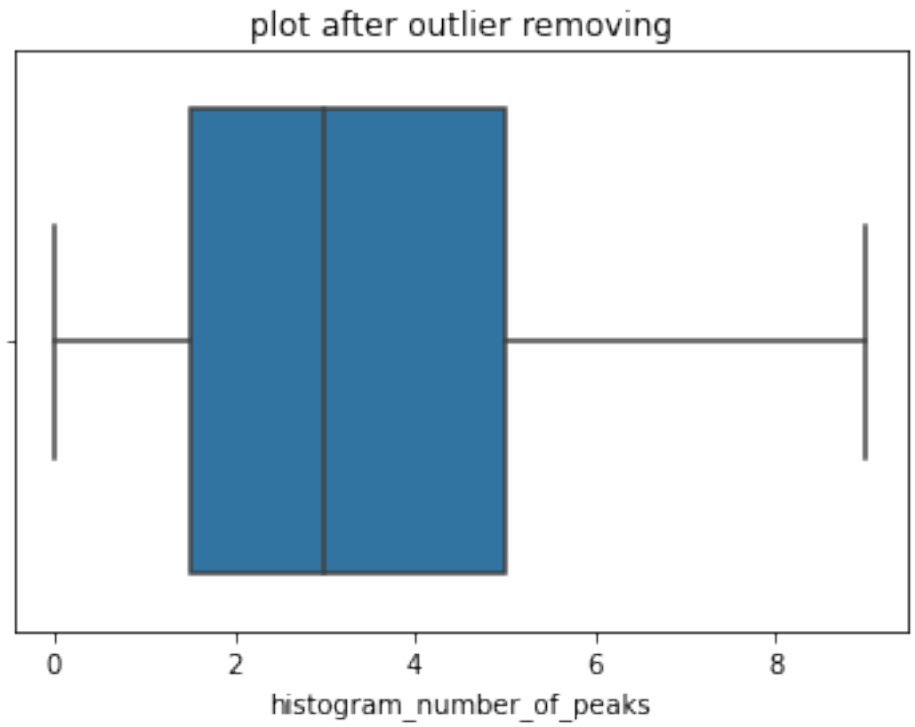

plot before outlier removing
60
80
100
120
140
160
histogram_min

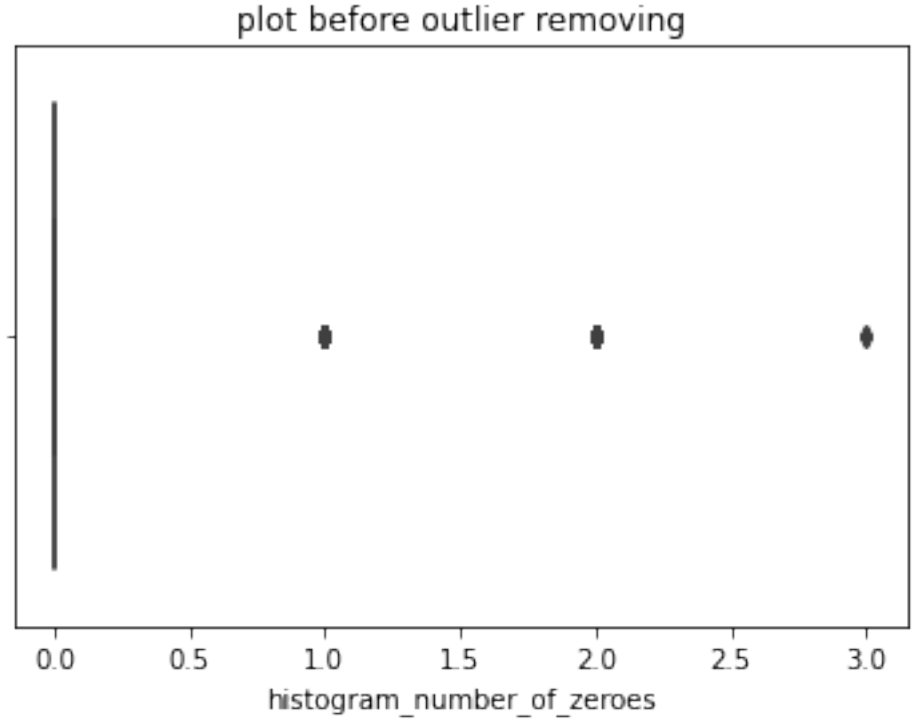

plot after outlier removing
60
80
100
120
140
160
histogram_min

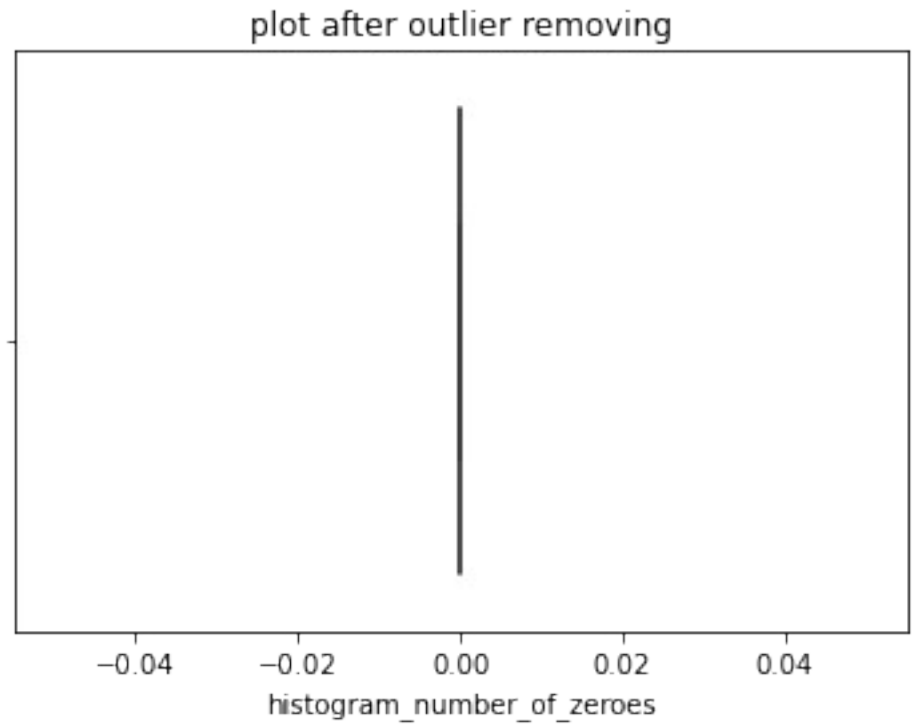

plot before outlier removing
120
140
160
180
200
histogram_max

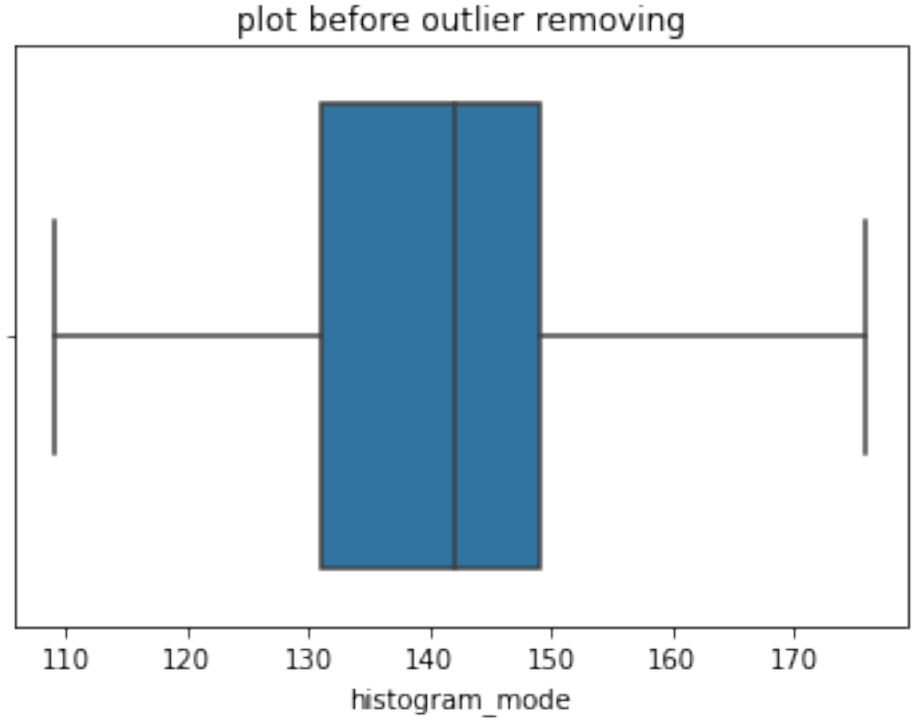

plot after outlier removing
130
140
150
160
170
180
190
200
histogram_max

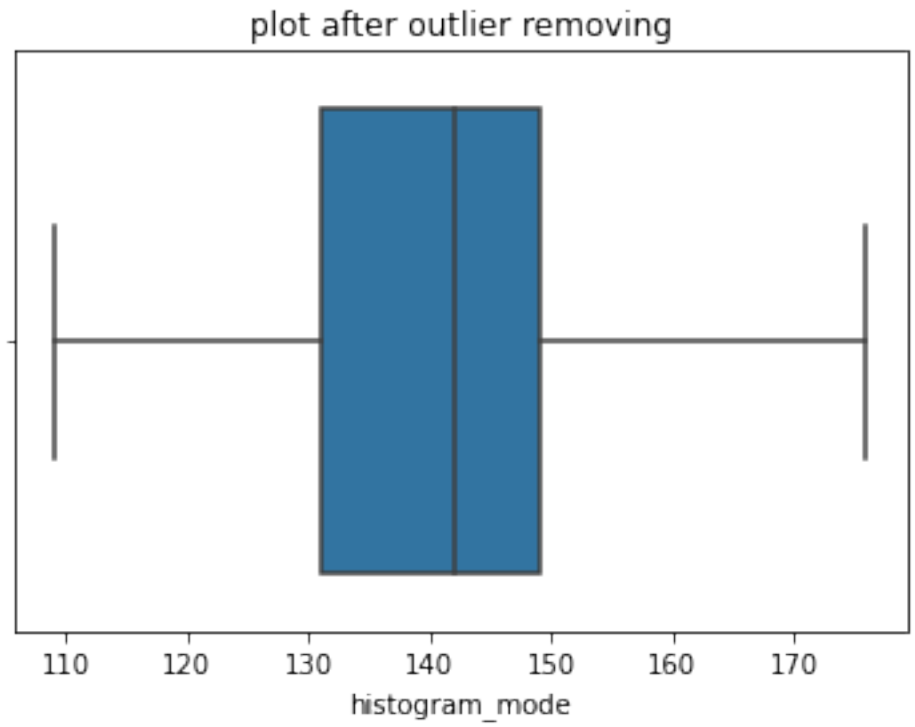

plot before outlier removing
0
2
4
6
8
10
12
14
16
histogram_number_of_peaks

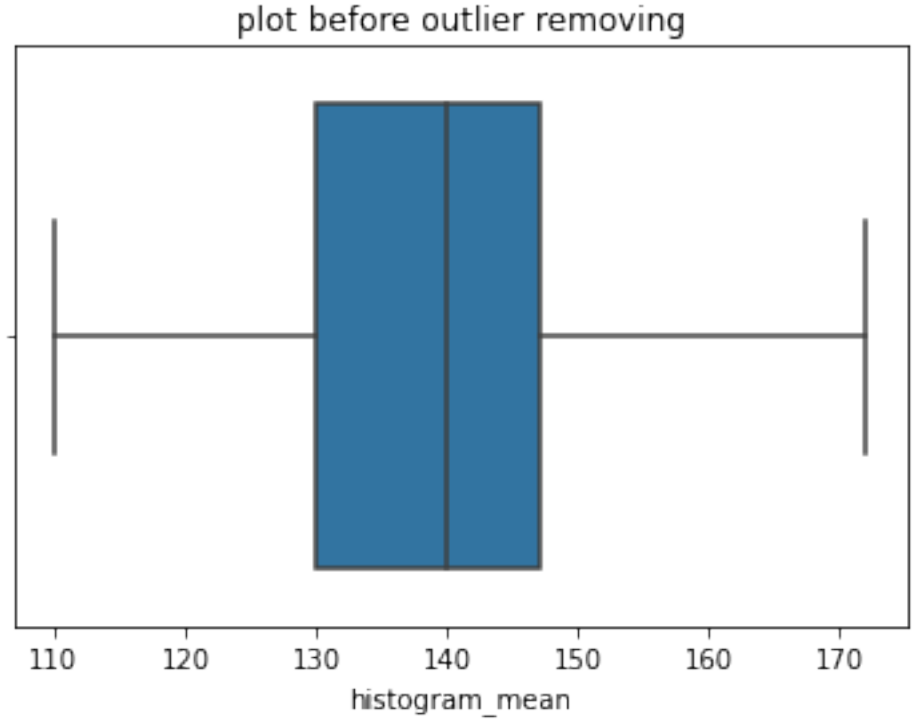

plot after outlier removing
0
2
4
6
8
histogram_number_of_peaks

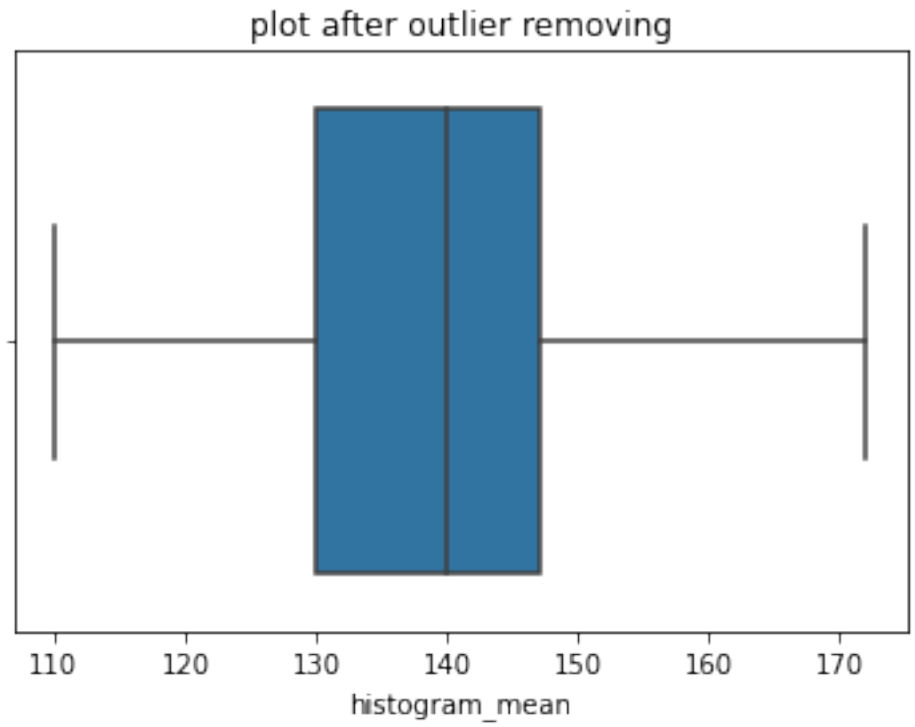

plot before outlier removing
0.0
0.5
1.0
1.5
2.0
2.5
3.0
histogram_number_of_zeroes

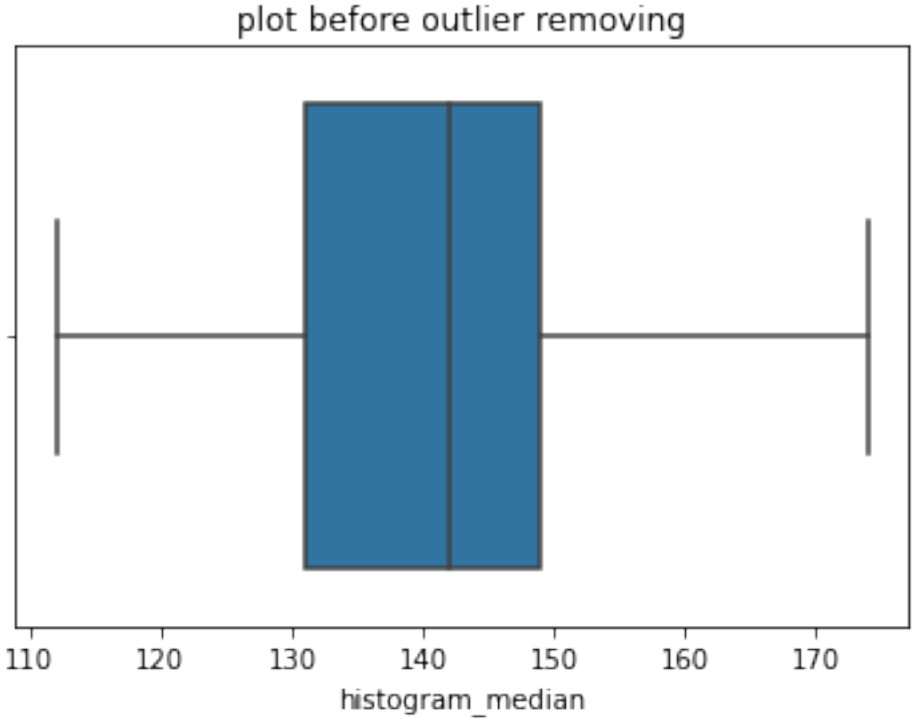

plot after outlier removing
−0.04
−0.02
0.00
0.02
0.04
histogram_number_of_zeroes

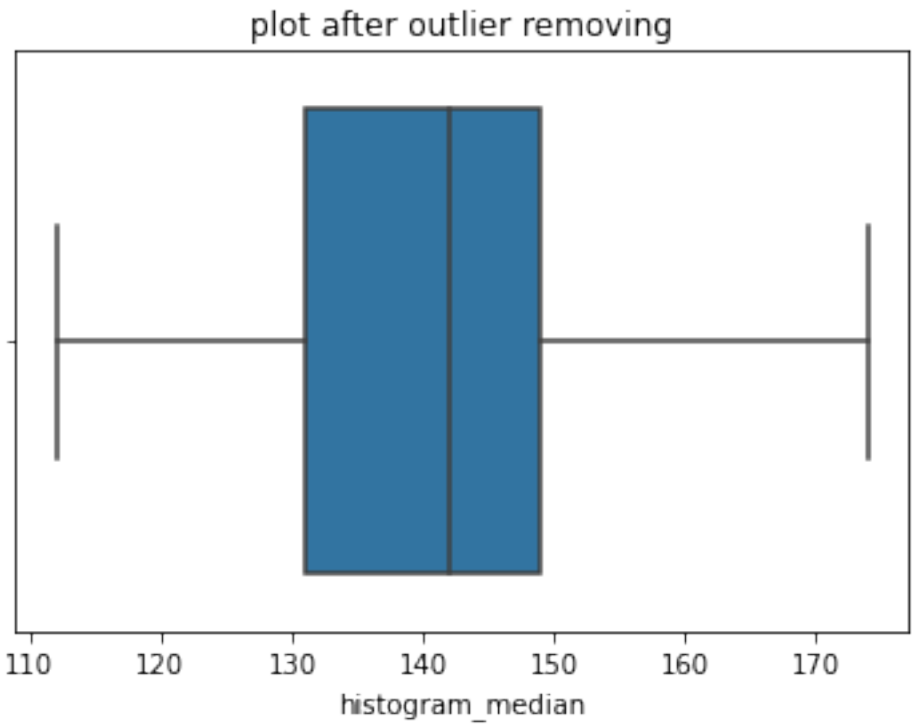

plot before outlier removing
110
120
130
140
150
160
170
histogram_mode

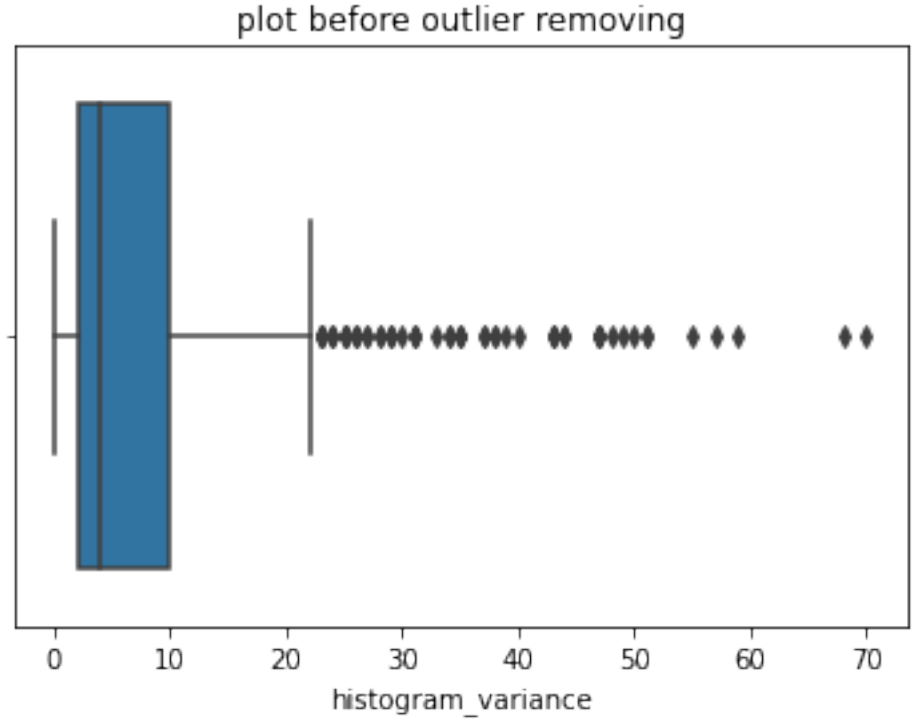

plot after outlier removing
110
120
130
140
150
160
170
histogram_mode

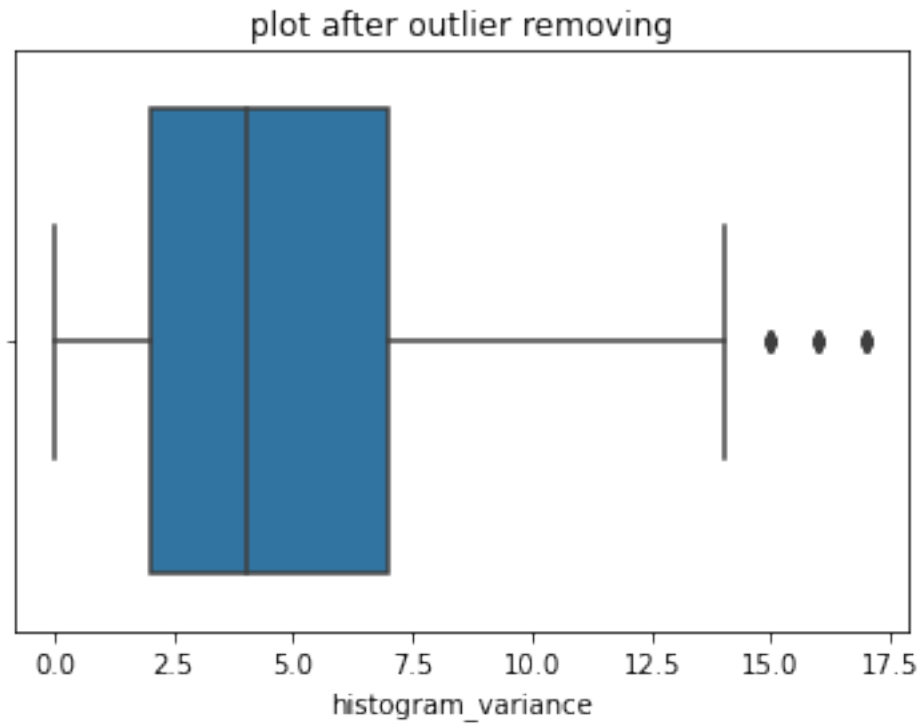

plot before outlier removing
110
120
130
140
150
160
170
histogram_mean

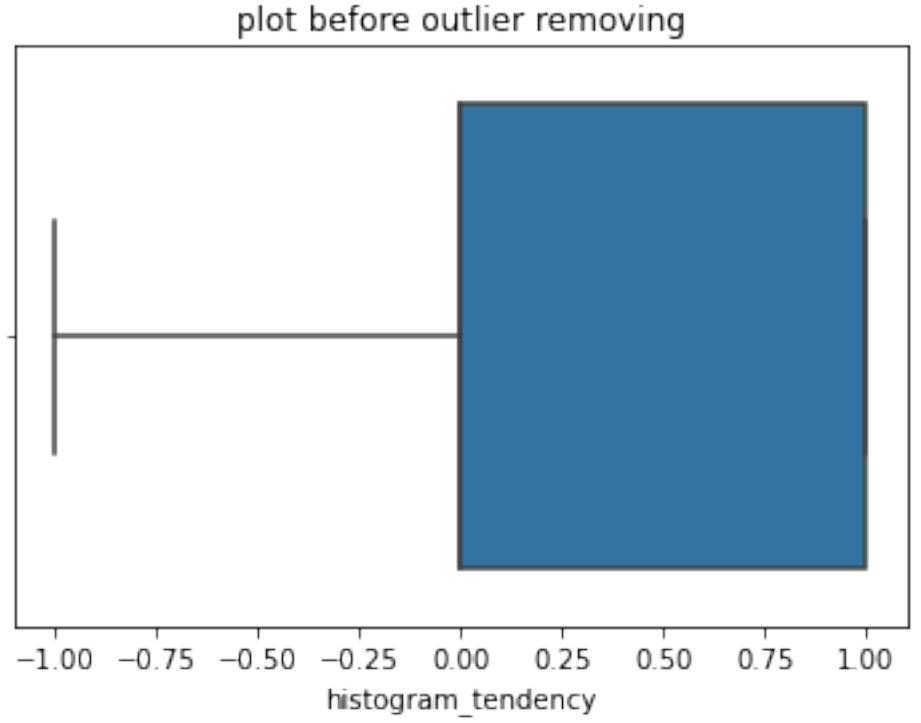

plot after outlier removing
110
120
130
140
150
160
170
histogram_mean

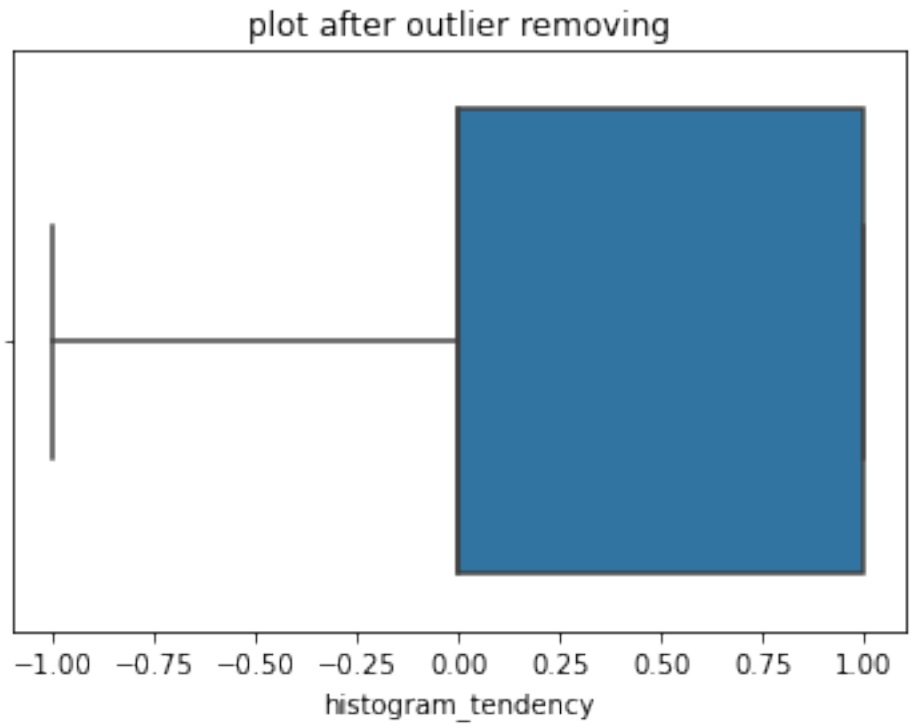

plot before outlier removing
110
120
130
140
150
160
170
histogram_median

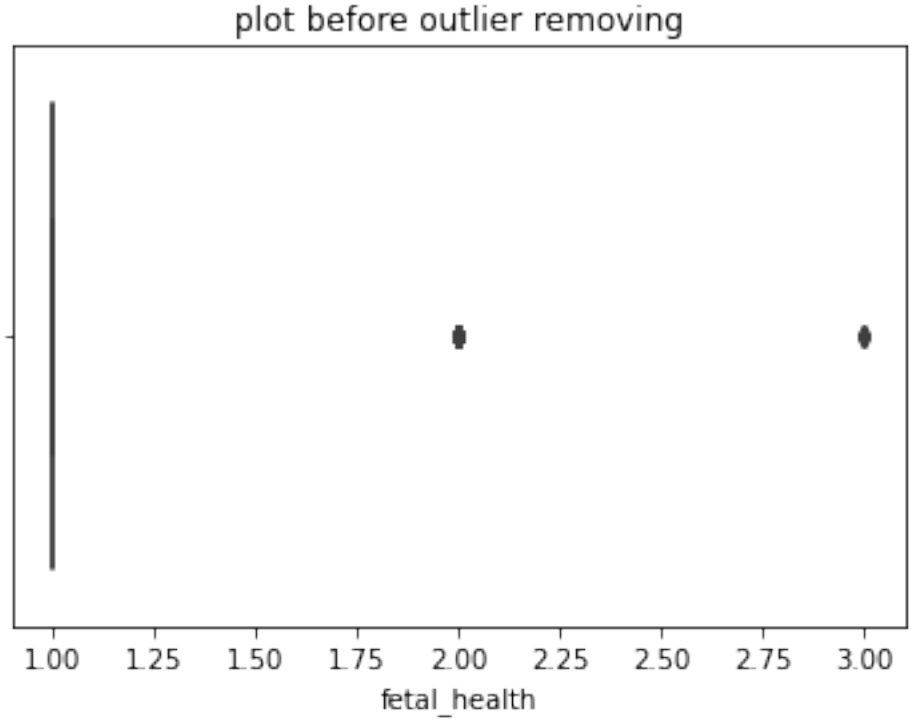

plot after outlier removing
110
120
130
140
150
160
170
histogram_median

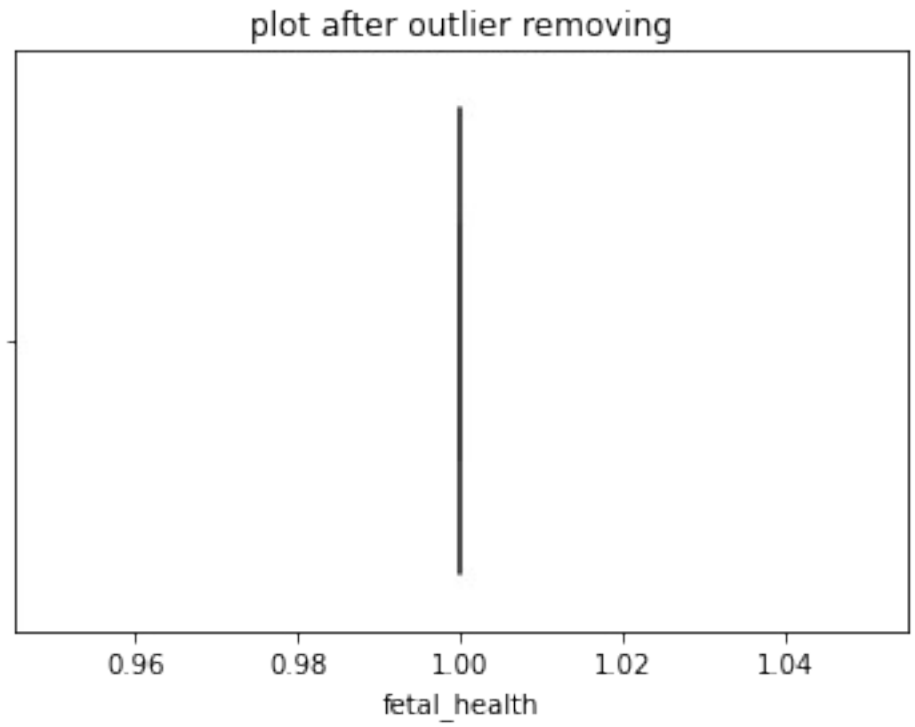

plot before outlier removing
0
10
20
30
40
50
60
70
histogram_variance


plot after outlier removing
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
histogram_variance


plot before outlier removing
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
histogram_tendency


plot after outlier removing
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
histogram_tendency


plot before outlier removing
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
fetal_health



Feature scaling

```
from sklearn.preprocessing import StandardScaler
SC=StandardScaler()
X_train=SC.fit_transform(X_train)
X_test=SC.fit_transform(X_test)

print(X_train)
```

```
[[-1.46513509 -0.83485626 -0.20820521 ... -1.18281773 -0.60623421
  -2.14821249]
 [ 0.16393359 -0.83485626 -0.0746156  ... -0.08025974 -0.53911612
   1.13129086]
 [-1.2615015   2.2251405  -0.00782079 ... -0.42480911 -0.37132092
  -2.14821249]
 ...
 [ 1.28391831  1.71514104 -0.20820521 ...  1.160118   -0.50555708
  -0.50846081]
 [ 1.08028472 -0.83485626  0.6823922  ...  0.74665876 -0.06928955
   1.13129086]
 [ 0.97846793 -0.3248568   0.94957142 ...  0.74665876  1.00459976
  -0.50846081]]
```

```
print(X_test)
```

```
[[ 0.92243937 -0.80741306  0.01293697 ...  0.39981121 -0.48257105
   1.06361291]
 [-1.28386413 -0.80741306 -0.20203067 ... -2.47496427  1.27487407
  -0.59523291]
```

```
 [-1.08329109 -0.80741306 -0.20203067 ... -0.83223542 -0.37919193
   1.06361291]
 ...
 [-1.18357761  0.24506147 -0.20203067 ... -0.76378839  0.34446195
   1.06361291]
 [-0.78243152  1.297536   -0.08627887 ... -0.010871    1.58501145
   1.06361291]
 [ 0.01986066 -0.80741306 -0.1854947  ... -0.010871   -0.65486959
  -0.59523291]]
```

Building the Model

```
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
classifier.fit(X_train,y_train)
```

```
KNeighborsClassifier()
```

```
y_pred=classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len
(y_test),1)),1))
```

```
[[1. 1.]
 [3. 3.]
 [1. 1.]
 ...
 [1. 1.]
 [1. 1.]
 [2. 1.]]
```

Cross validation

```
parameters ={ 'n_neighbors' : [5,7,9,11,13,15],
              'weights' : ['uniform','distance'],
              'metric' : ['minkowski','euclidean','manhattan']}
from sklearn.model_selection import RandomizedSearchCV
cv = RandomizedSearchCV(classifier,parameters ,cv=5)
```

```
cv.fit(X_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=KNeighborsClassifier(),
                   param_distributions={'metric': ['minkowski',
'euclidean',
                                                   'manhattan'],
                                        'n_neighbors': [5, 7, 9, 11,
13, 15],
                                        'weights': ['uniform',
'distance']})
```

```
y_pred = cv.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print('\n Hyperparametric tuned knn 
accuracy:',accuracy_score(y_pred,y_test))
```

```
 Hyperparametric tuned knn accuracy: 0.8960784313725491
```

```
test_set=pd.read_csv('test.csv')
y_pre =classifier.predict(test_set)
print(y_pre )
```

```
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 
1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Saving the file in csv format

```
y = pd.DataFrame(y_pred).astype(int)
y.to_csv('Result.csv')
```