Machine_Learning_Classification_Algorithms_Role_in_Fetal_Health_Prediction

By : VENKATESH_MUNGI, Email: venkateshmungi@gmail.com (mailto:venkateshmungi@gmail.com)

Exploratory Data Analysis

In [1]:
```
# importing the required libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
C:\Users\VENKATESH MUNGI\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\VENKATESH MUNGI\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\.libs\libopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-win_amd64.dll
C:\Users\VENKATESH MUNGI\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\.libs\libopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

In [2]:
```
# Loading the dataset

df = pd.read_csv(r"C:\\PYTHON\\PANDAS\\main_projects\\fetal_health.csv")
df
```

Out[2]:

	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
0	120	0.000	0.000	0.000	0.000	0.0	0.0	73	0.5	
1	132	0.006	0.000	0.006	0.003	0.0	0.0	17	2.1	
2	133	0.003	0.000	0.008	0.003	0.0	0.0	16	2.1	
3	134	0.003	0.000	0.008	0.003	0.0	0.0	16	2.4	
4	132	0.007	0.000	0.008	0.000	0.0	0.0	16	2.4	
...	...	...	...	...	...	...	...	...	...	
2121	140	0.000	0.000	0.007	0.000	0.0	0.0	79	0.2	
2122	140	0.001	0.000	0.007	0.000	0.0	0.0	78	0.4	
2123	140	0.001	0.000	0.007	0.000	0.0	0.0	79	0.4	
2124	140	0.001	0.000	0.006	0.000	0.0	0.0	78	0.4	
2125	142	0.002	0.002	0.008	0.000	0.0	0.0	74	0.4	

2126 rows × 22 columns

In [3]:
```
# Shallow copying the dataset for our future convenience

fetal = df.copy()
fetal
```

Out[3]:

	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
0	120	0.000	0.000	0.000	0.000	0.0	0.0	73	0.5	
1	132	0.006	0.000	0.006	0.003	0.0	0.0	17	2.1	
2	133	0.003	0.000	0.008	0.003	0.0	0.0	16	2.1	
3	134	0.003	0.000	0.008	0.003	0.0	0.0	16	2.4	
4	132	0.007	0.000	0.008	0.000	0.0	0.0	16	2.4	
...	...	...	...	...	...	...	...	...	...	
2121	140	0.000	0.000	0.007	0.000	0.0	0.0	79	0.2	
2122	140	0.001	0.000	0.007	0.000	0.0	0.0	78	0.4	
2123	140	0.001	0.000	0.007	0.000	0.0	0.0	79	0.4	
2124	140	0.001	0.000	0.006	0.000	0.0	0.0	78	0.4	
2125	142	0.002	0.002	0.008	0.000	0.0	0.0	74	0.4	

2126 rows × 22 columns

- Here i have done shallow copy of the dataset, because: if any changes made in the dataset that will not refelect in original dataset.

Data Understanding

- baseline value: Baseline Fetal Heart Rate (FHR) (beats per minute)
- accelerations: Number of accelerations per second
- fetal_movement: Number of fetal movements per second
- uterine_contractions: Number of uterine contractions per second
- light_decelerations: Number of light decelerations (LDs) per second
- severe_decelerations: Number of severe decelerations (SDs) per second
- prolongued_decelerations: Number of prolonged decelerations (PDs) per second
- abnormal_short_term_variability: Percentage of time with abnormal short term variability
- mean_value_of_short_term_variability: Mean value of short term variability
- percentage_of_time_with_abnormal_long_term_variability: Percentage of time with abnormal long term variability
- mean_value_of_long_term_variability: Mean value of long term variability
- histogram_width: Width of histogram made using all values from a record
- histogram_min: Histogram minimum value
- histogram_max: Histogram maximum value
- histogram_number_of_peaks: Number of peaks in the exam histogram
- histogram_number_of_zeroes: Number of zeros in the exam histogram
- histogram_mode: Histogram mode

- histogram_mean: Histogram mean
- histogram_median: Histogram median
- histogram_variance: Histogram variance
- histogram_tendency: Histogram tendency
- fetal_health: Encoded as 1-Normal; 2-Suspect; 3-Pathological. It is our very target column in the dataset.

In [4]:
```
# Metadata information

fetal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   baseline value                                          2126 non-null   int64  
 1   accelerations                                           2126 non-null   float64
 2   fetal_movement                                          2126 non-null   float64
 3   uterine_contractions                                    2126 non-null   float64
 4   light_decelerations                                     2126 non-null   float64
 5   severe_decelerations                                    2126 non-null   float64
 6   prolongued_decelerations                                2126 non-null   float64
 7   abnormal_short_term_variability                         2126 non-null   int64  
 8   mean_value_of_short_term_variability                    2126 non-null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  2126 non-null   int64  
 10  mean_value_of_long_term_variability                     2126 non-null   float64
 11  histogram_width                                         2126 non-null   int64  
 12  histogram_min                                           2126 non-null   int64  
 13  histogram_max                                           2126 non-null   int64  
 14  histogram_number_of_peaks                               2126 non-null   int64  
 15  histogram_number_of_zeroes                              2126 non-null   int64  
 16  histogram_mode                                          2126 non-null   int64  
 17  histogram_mean                                          2126 non-null   int64  
 18  histogram_median                                        2126 non-null   int64  
 19  histogram_variance                                      2126 non-null   int64  
 20  histogram_tendency                                      2126 non-null   int64  
 21  fetal_health                                            2126 non-null   int64  
dtypes: float64(8), int64(14)
memory usage: 365.5 KB
```

- By metadata information it is clear that our data frame has eight continuous values and fourteen discrete values

```
In [5]: # Statistics of the data

        fetal.describe().T.style.set_properties(**{'background-color': 'green','color': 'white','border-color': 'white'})

Out[5]:
```

	count	mean	std	min	25%	50%	75%	max
baseline value	2126.000000	133.303857	9.840844	106.000000	126.000000	133.000000	140.000000	160.000000
accelerations	2126.000000	0.003178	0.003866	0.000000	0.000000	0.002000	0.006000	0.019000
fetal_movement	2126.000000	0.009481	0.046666	0.000000	0.000000	0.000000	0.003000	0.481000
uterine_contractions	2126.000000	0.004366	0.002946	0.000000	0.002000	0.004000	0.007000	0.015000
light_decelerations	2126.000000	0.001889	0.002960	0.000000	0.000000	0.000000	0.003000	0.015000
severe_decelerations	2126.000000	0.000003	0.000057	0.000000	0.000000	0.000000	0.000000	0.001000
prolongued_decelerations	2126.000000	0.000159	0.000590	0.000000	0.000000	0.000000	0.000000	0.005000
abnormal_short_term_variability	2126.000000	46.990122	17.192814	12.000000	32.000000	49.000000	61.000000	87.000000
mean_value_of_short_term_variability	2126.000000	1.332785	0.883241	0.200000	0.700000	1.200000	1.700000	7.000000
percentage_of_time_with_abnormal_long_term_variability	2126.000000	9.846660	18.396880	0.000000	0.000000	0.000000	11.000000	91.000000
mean_value_of_long_term_variability	2126.000000	8.187629	5.628247	0.000000	4.600000	7.400000	10.800000	50.700000
histogram_width	2126.000000	70.445908	38.955693	3.000000	37.000000	67.500000	100.000000	180.000000
histogram_min	2126.000000	93.579492	29.560212	50.000000	67.000000	93.000000	120.000000	159.000000
histogram_max	2126.000000	164.025400	17.944183	122.000000	152.000000	162.000000	174.000000	238.000000
histogram_number_of_peaks	2126.000000	4.068203	2.949386	0.000000	2.000000	3.000000	6.000000	18.000000
histogram_number_of_zeroes	2126.000000	0.323612	0.706059	0.000000	0.000000	0.000000	0.000000	10.000000
histogram_mode	2126.000000	137.452023	16.381289	60.000000	129.000000	139.000000	148.000000	187.000000
histogram_mean	2126.000000	134.610536	15.593596	73.000000	125.000000	136.000000	145.000000	182.000000
histogram_median	2126.000000	138.090310	14.466589	77.000000	129.000000	139.000000	148.000000	186.000000
histogram_variance	2126.000000	18.808090	28.977636	0.000000	2.000000	7.000000	24.000000	269.000000
histogram_tendency	2126.000000	0.320320	0.610829	-1.000000	0.000000	0.000000	1.000000	1.000000
fetal_health	2126.000000	1.304327	0.614377	1.000000	1.000000	1.000000	1.000000	3.000000

In [6]:
```
# Getting Median of the dataset
fetal.median()
```

Out[6]:
```
baseline value                                            133.000
accelerations                                               0.002
fetal_movement                                              0.000
uterine_contractions                                        0.004
light_decelerations                                         0.000
severe_decelerations                                        0.000
prolongued_decelerations                                    0.000
abnormal_short_term_variability                            49.000
mean_value_of_short_term_variability                        1.200
percentage_of_time_with_abnormal_long_term_variability      0.000
mean_value_of_long_term_variability                         7.400
histogram_width                                            67.500
histogram_min                                              93.000
histogram_max                                             162.000
histogram_number_of_peaks                                   3.000
histogram_number_of_zeroes                                  0.000
histogram_mode                                            139.000
histogram_mean                                            136.000
histogram_median                                          139.000
histogram_variance                                          7.000
histogram_tendency                                          0.000
fetal_health                                                1.000
dtype: float64
```

In [7]:
```
# Getting Mode of the dataset
fetal.mode()
```

Out[7]:

	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_ab
0	133	0.0	0.0	0.0	0.0	0.0	0.0	60	0.8	

1 rows × 22 columns

In [8]:
```
# Range of the dataset
fetal_range = fetal.max() - fetal.min()

fetal_range
```

Out[8]:
```
baseline value                                             54.000
accelerations                                               0.019
fetal_movement                                              0.481
uterine_contractions                                        0.015
light_decelerations                                         0.015
severe_decelerations                                        0.001
prolongued_decelerations                                    0.005
abnormal_short_term_variability                            75.000
mean_value_of_short_term_variability                        6.800
percentage_of_time_with_abnormal_long_term_variability     91.000
mean_value_of_long_term_variability                        50.700
histogram_width                                           177.000
histogram_min                                             109.000
histogram_max                                             116.000
histogram_number_of_peaks                                  18.000
histogram_number_of_zeroes                                 10.000
histogram_mode                                            127.000
histogram_mean                                            109.000
histogram_median                                          109.000
histogram_variance                                        269.000
histogram_tendency                                          2.000
fetal_health                                                2.000
dtype: float64
```

In [9]:
```
# getting variance of the dataset
fetal.var()
```

Out[9]:
```
baseline value                                            9.684222e+01
accelerations                                             1.494279e-05
fetal_movement                                            2.177701e-03
uterine_contractions                                      8.679323e-06
light_decelerations                                       8.762835e-06
severe_decelerations                                      3.283272e-09
prolongued_decelerations                                  3.480381e-07
abnormal_short_term_variability                           2.955928e+02
mean_value_of_short_term_variability                      7.801153e-01
percentage_of_time_with_abnormal_long_term_variability    3.384452e+02
mean_value_of_long_term_variability                       3.167716e+01
histogram_width                                           1.517546e+03
histogram_min                                             8.738061e+02
histogram_max                                             3.219937e+02
histogram_number_of_peaks                                 8.698876e+00
histogram_number_of_zeroes                                4.985198e-01
histogram_mode                                            2.683466e+02
histogram_mean                                            2.431602e+02
histogram_median                                          2.092822e+02
histogram_variance                                        8.397034e+02
histogram_tendency                                        3.731116e-01
fetal_health                                              3.774589e-01
dtype: float64
```

In [10]:
```
# Knowing Skewness and Kurtosis of the data
from scipy.stats import skew
from scipy.stats import kurtosis
```

In [11]:
```
fetal.skew()
```

Out[11]:
```
baseline value                                             0.020312
accelerations                                              1.204392
fetal_movement                                             7.811477
uterine_contractions                                       0.159315
light_decelerations                                        1.718437
severe_decelerations                                      17.353457
prolongued_decelerations                                   4.323965
abnormal_short_term_variability                           -0.011829
mean_value_of_short_term_variability                       1.657339
percentage_of_time_with_abnormal_long_term_variability     2.195075
mean_value_of_long_term_variability                        1.331998
histogram_width                                            0.314235
histogram_min                                              0.115784
histogram_max                                              0.577862
histogram_number_of_peaks                                  0.892886
histogram_number_of_zeroes                                 3.920287
histogram_mode                                            -0.995178
histogram_mean                                            -0.651019
histogram_median                                          -0.478414
histogram_variance                                         3.219974
histogram_tendency                                        -0.311632
fetal_health                                               1.849934
dtype: float64
```

In [12]:
```
fetal.kurtosis()
```

Out[12]:
```
baseline value                                           -0.292943
accelerations                                             0.767648
fetal_movement                                           64.260821
uterine_contractions                                     -0.635071
light_decelerations                                       2.517461
severe_decelerations                                    299.424142
prolongued_decelerations                                 20.515918
abnormal_short_term_variability                          -1.051030
mean_value_of_short_term_variability                      4.700756
percentage_of_time_with_abnormal_long_term_variability    4.252998
mean_value_of_long_term_variability                       4.131254
histogram_width                                          -0.902287
histogram_min                                            -1.290422
histogram_max                                             0.632769
histogram_number_of_peaks                                 0.504211
histogram_number_of_zeroes                               30.365084
histogram_mode                                            3.009531
histogram_mean                                            0.933427
histogram_median                                          0.667259
histogram_variance                                       15.131589
histogram_tendency                                       -0.652639
fetal_health                                              2.091215
dtype: float64
```

In [13]:
```
# Checkng for null values in the dataset
fetal.isnull().sum()
```

Out[13]:
```
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
```

- From the above informatio it is clear that features in this dataset doesn't has any null values

Look into Important Features

In [14]:
```
# Let's Look into Important Features in our dataset using SelectKBest
"""
ANOVA f-test Feature Selection

ANOVA is an acronym for “analysis of variance” and is a parametric statistical hypothesis test for determining whether the means from two or more samples of data (often three or more) 

An F-statistic, or F-test, is a class of statistical tests that calculate the ratio between variances values, such as the variance from two different samples or the explained and unexp

Importantly, ANOVA is used when one variable is numeric and one is categorical, such as numerical input variables and a classification target variable in a classification task.

The results of this test can be used for feature selection where those features that are independent of the target variable can be removed from the dataset.

When the outcome is numeric, and […] the predictor has more than two levels, the traditional ANOVA F-statistic can be calculated.

The scikit-learn machine library provides an implementation of the ANOVA f-test in the f_classif() function. This function can be used in a feature selection strategy, such as selectin

"""
```

Out[14]:
```
'\nANOVA f-test Feature Selection\n\nANOVA is an acronym for “analysis of variance” and is a parametric statistical hypothesis test for determining whether the means from two or more
samples of data (often three or more) come from the same distribution or not.\n\nAn F-statistic, or F-test, is a class of statistical tests that calculate the ratio between variances
values, such as the variance from two different samples or the explained and unexplained variance by a statistical test, like ANOVA. The ANOVA method is a type of F-statistic referred
to here as an ANOVA f-test.\n\nImportantly, ANOVA is used when one variable is numeric and one is categorical, such as numerical input variables and a classification target variable i
n a classification task.\n\nThe results of this test can be used for feature selection where those features that are independent of the target variable can be removed from the datase
t.\n\nWhen the outcome is numeric, and […] the predictor has more than two levels, the traditional ANOVA F-statistic can be calculated.\n\nThe scikit-learn machine library provides an
implementation of the ANOVA f-test in the f_classif() function. This function can be used in a feature selection strategy, such as selecting the top k most relevant features (largest
values) via the SelectKBest class.\n\n'
```

In [15]:
```
from sklearn.feature_selection import SelectKBest, f_classif
```

In [16]:
```
select = SelectKBest(score_func=f_classif, k=21) # K = Number of features to select
```

In [17]:
```
fitting = select.fit(fetal.drop('fetal_health',axis=1),fetal['fetal_health'])
```

In [18]:
```
x = pd.DataFrame(fitting.scores_)
```

In [19]:
```
columns = pd.DataFrame(fetal.drop('fetal_health',axis=1).columns)
```

In [20]:
```
fscores = pd.concat([columns,x],axis=1)
```

In [21]:
```
fscores.columns = ['Attribute','Score']
```

In [22]:
```
fscores.sort_values(by='Score',ascending=False)
```

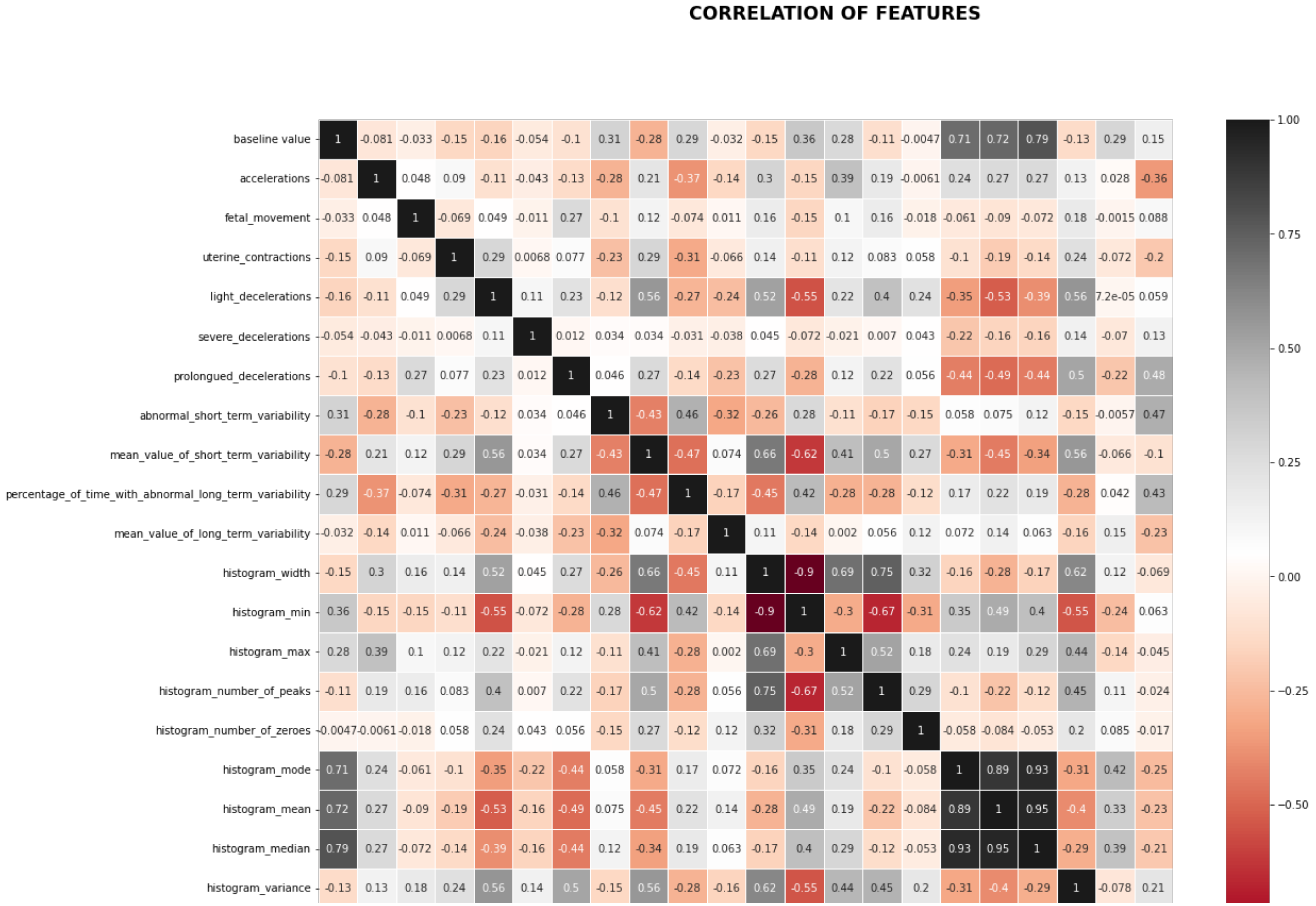
Out[22]:

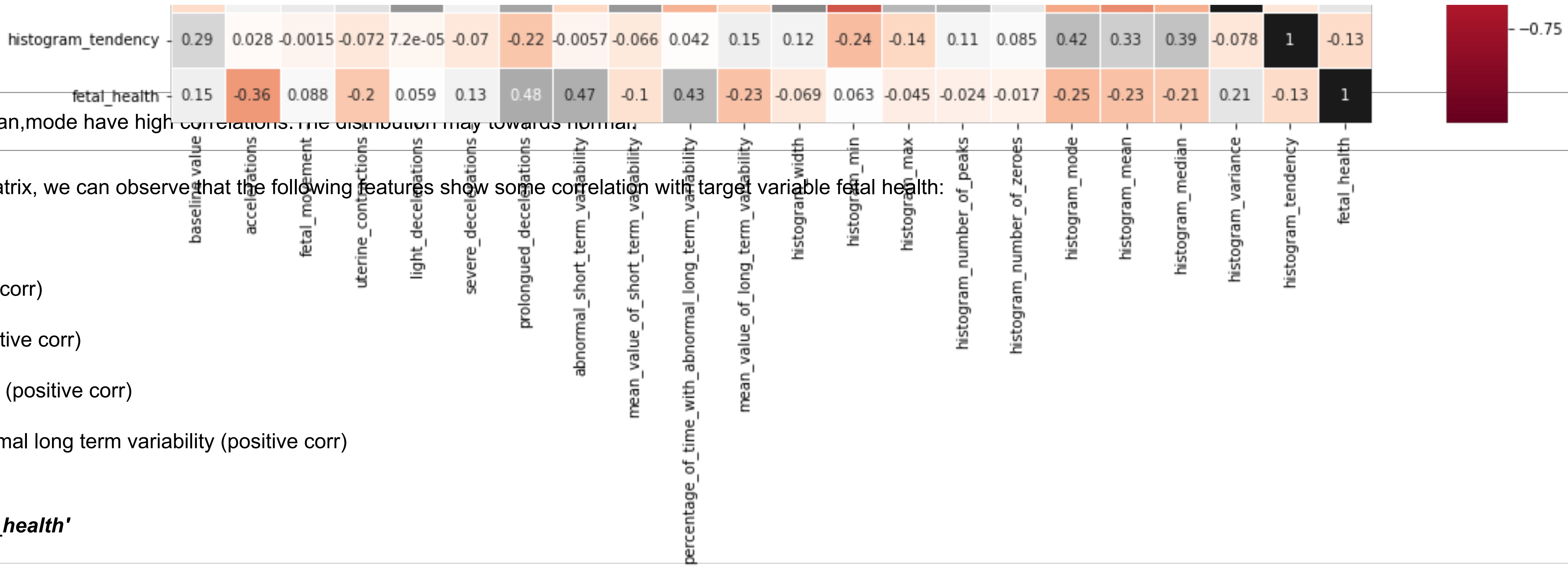
	Attribute	Score
6	prolongued_decelerations	505.853206
9	percentage_of_time_with_abnormal_long_term_var...	345.156385
7	abnormal_short_term_variability	343.820419
17	histogram_mean	297.625497
16	histogram_mode	275.117696
18	histogram_median	248.772237
1	accelerations	196.027523
19	histogram_variance	150.796849
0	baseline value	140.621076
8	mean_value_of_short_term_variability	119.882006
3	uterine_contractions	93.715743
12	histogram_min	87.340503
10	mean_value_of_long_term_variability	70.174093
4	light_decelerations	66.864754
11	histogram_width	55.088241
20	histogram_tendency	44.542294
5	severe_decelerations	28.448156
14	histogram_number_of_peaks	12.104834
2	fetal_movement	11.679797
13	histogram_max	2.464923
15	histogram_number_of_zeroes	2.196373

- Fromtheaboveinformation it is clear that feature 'prolongued_decelerations' has highest correlaation and 'histogram_number_of_zeroes' has lowest correlation.
- First i want to use all features to build models. If the model accuracy is good its okay wtih all features, otherwise i would like to drop some features based on particular thresh-hold value.

Let's Check Correlation of Variables with output variable "fetal_health"

In [23]:
```
plt.figure(figsize = (18,15))
plt.suptitle("CORRELATION OF FEATURES", fontsize = 'xx-large', weight = 'extra bold' , ha = 'center')
corr = fetal.corr()
ax = sns.heatmap(corr, cmap='RdGy', annot=True, linewidths= 1.0)
plt.show()
```





- Observation:

histogram mean,median,mode have high correlations.The distribution may towards normal.

From the above correlation matrix, we can observe that the following features show some correlation with target variable fetal health:

accelerations (negative corr)

uterine contractions (negative corr)

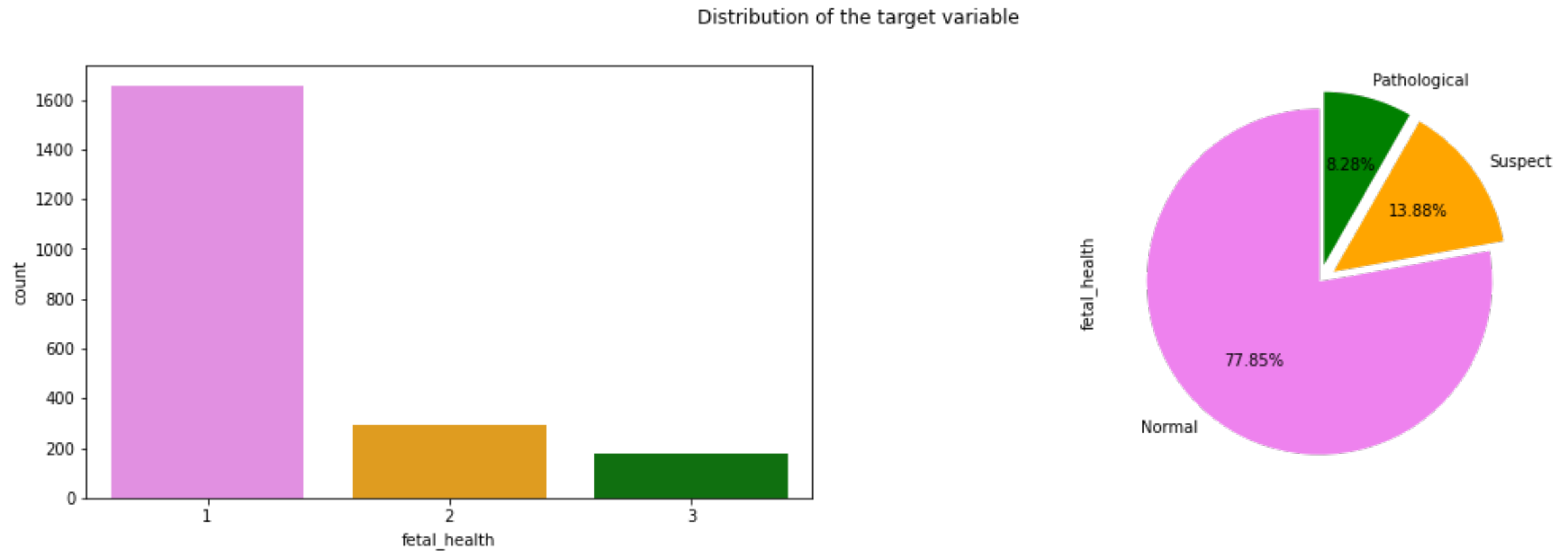
prolonged_decelerations (positive corr)

abnormal short term variability (positive corr)

percentage of time with abnormal long term variability (positive corr)

Let's look into out put 'fetal_health'

In [24]:
```
plt.figure(figsize=(18,5))
plt.subplot(1,2,1)
sns.countplot(x=fetal['fetal_health'], palette=['violet','orange','green'])
plt.subplot(1,2,2)
df['fetal_health'].value_counts().plot(kind='pie', autopct='%.2f%%', explode=[0,0.1,0.1], startangle=90, colors=['violet','orange','green'],labels = ['Normal','Suspect','Pathological']
plt.suptitle('Distribution of the target variable')
plt.show()
```



Observation:

- As class 3 "Pathological" has very little frequency it means that we are facing a “rare diseases” dataset. Also, this means that we can not remove any class and we have to study all of them.

- An highly Imbalanced Dataset. Which is obvious as Normal would be dominant.

Solution:

- Oversampling of the minority classes. To make better predictions.

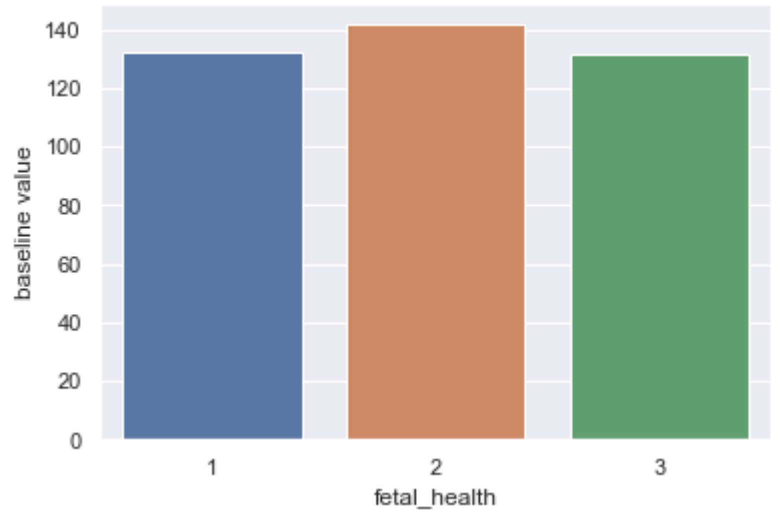
In [25]:
```
grouped = fetal.groupby(by='fetal_health').mean()
grouped
```

Out[25]:

	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_
fetal_health										
1	131.981873	0.003992	0.007963	0.004781	0.001941	6.042296e-07	0.000051	42.465861	1.430634	
2	141.684746	0.000275	0.008332	0.002390	0.000536	0.000000e+00	0.000095	61.901695	0.638983	
3	131.687500	0.000392	0.025676	0.003784	0.003670	3.409091e-05	0.001273	64.539773	1.575568	

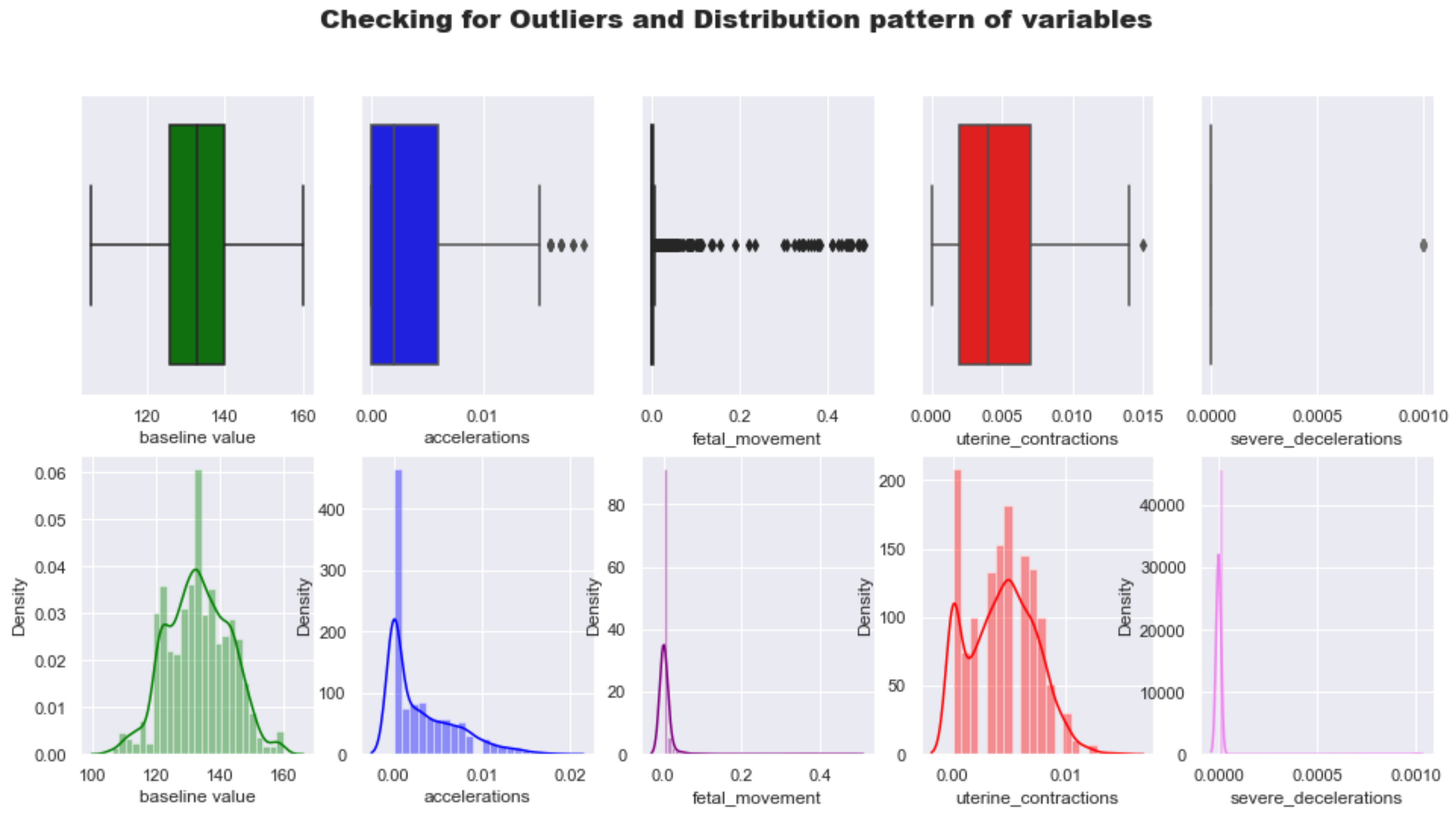
3 rows × 21 columns

In [26]:
```
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
sns.set()
for index,i in enumerate(grouped.columns,start=1):
    plt.figure(figsize=(6,4))
    sns.barplot(data=grouped,x=grouped.index,y=grouped[i])
    plt.show()
```



Checking for outliers in our data

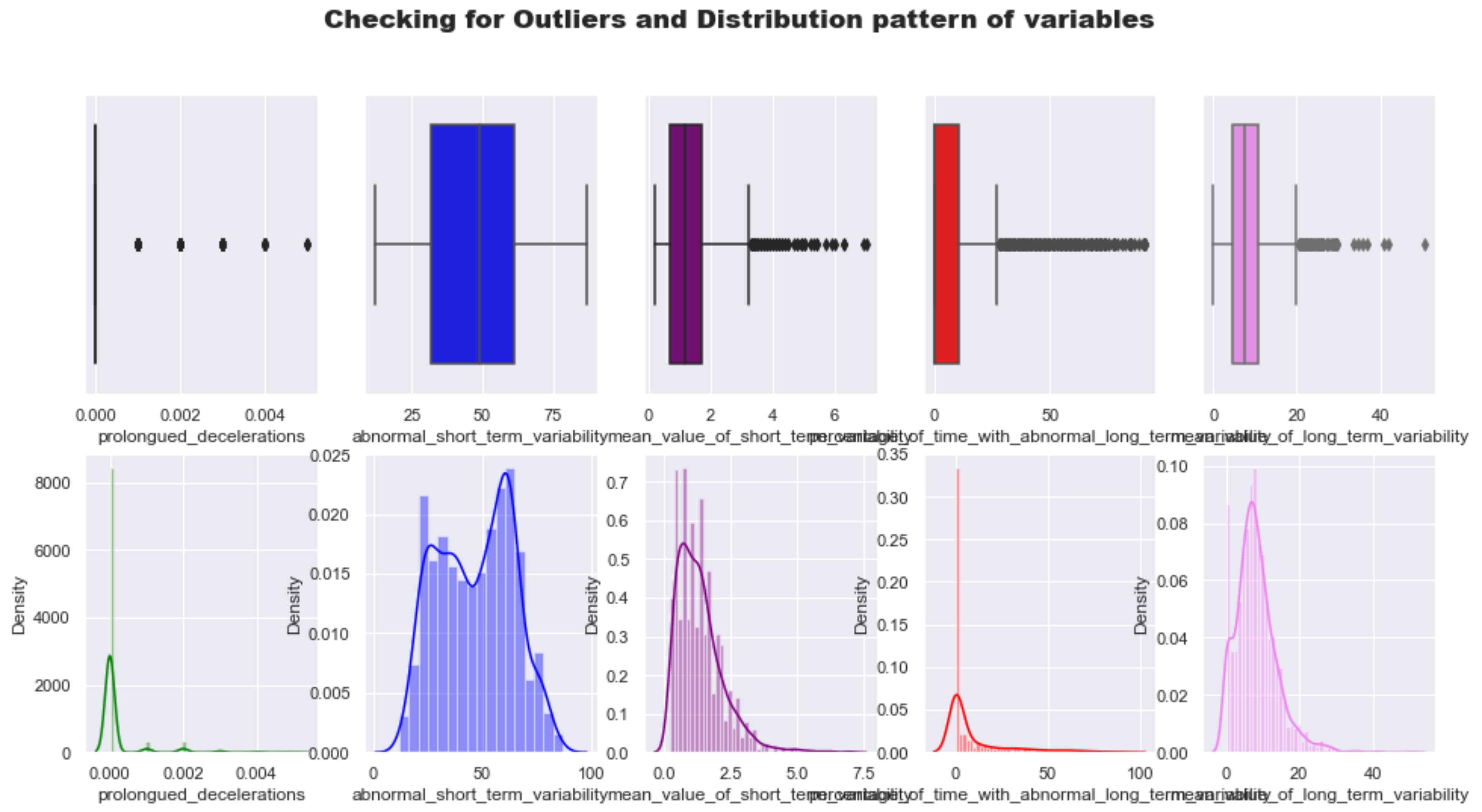
```
In [27]: plt.figure(figsize = (16,8))
         plt.suptitle("Checking for Outliers and Distribution pattern of variables", fontsize = 'x-large',weight = 'extra bold',ha = "center")
         plt.subplot(2,5,1)
         sns.boxplot(fetal['baseline value'], color = 'green')
         plt.subplot(2,5,2)
         sns.boxplot(fetal['accelerations'], color = 'blue')
         plt.subplot(2,5,3)
         sns.boxplot(fetal['fetal_movement'], color = 'purple')
         plt.subplot(2,5,4)
         sns.boxplot(fetal['uterine_contractions'], color = 'red')
         plt.subplot(2,5,5)
         sns.boxplot(fetal['severe_decelerations'], color = 'violet')
         plt.subplot(2,5,6)
         sns.distplot(fetal['baseline value'], color = 'green')
         plt.subplot(2,5,7)
         sns.distplot(fetal['accelerations'], color = 'blue')
         plt.subplot(2,5,8)
         sns.distplot(fetal['fetal_movement'], color = 'purple')
         plt.subplot(2,5,9)
         sns.distplot(fetal['uterine_contractions'], color = 'red')
         plt.subplot(2,5,10)
         sns.distplot(fetal['severe_decelerations'], color = 'violet')
         plt.show()
```



- From the above graph it is clear that all features has outliers except 'baseline value.'

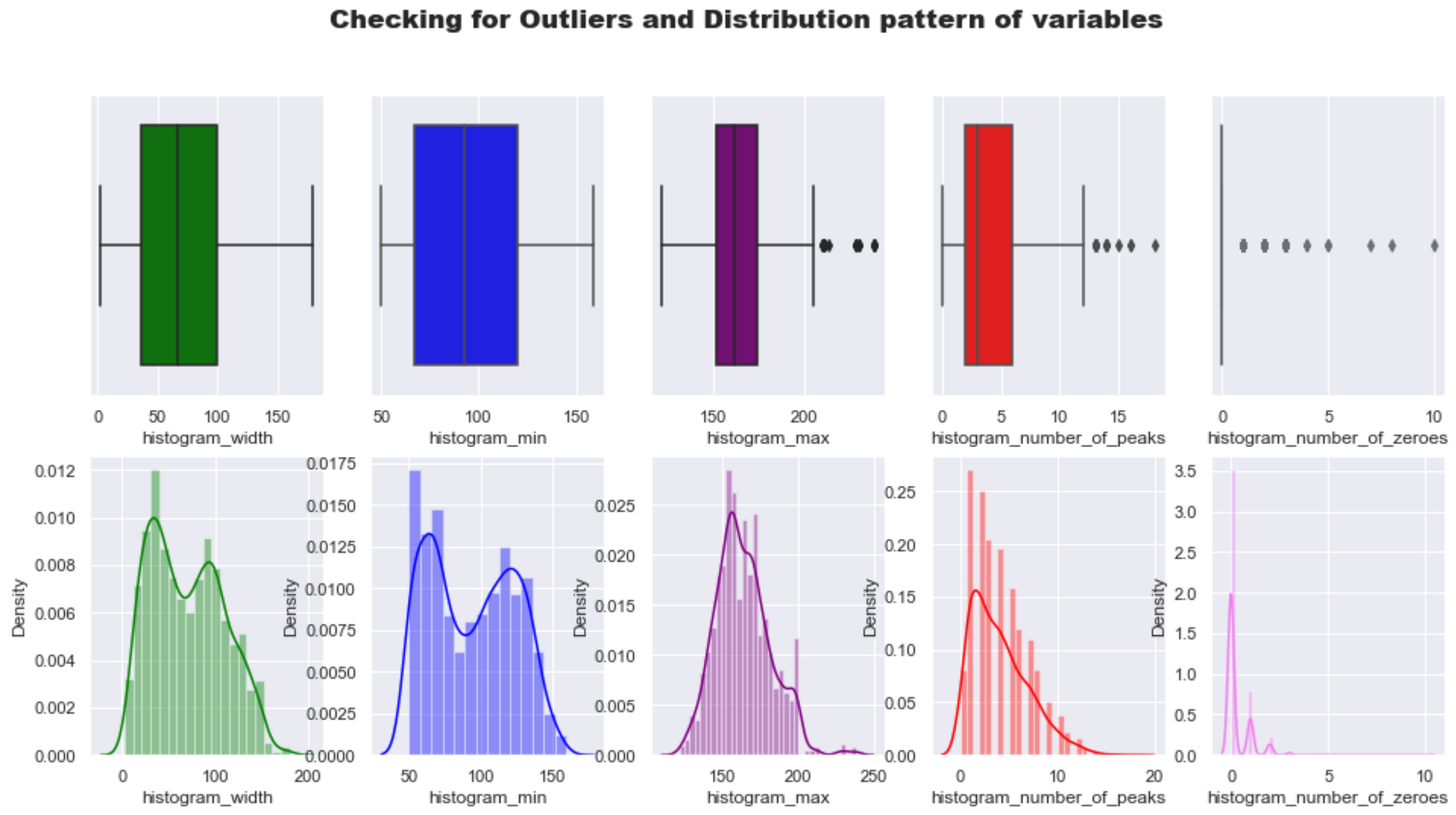
- All features are right skewed except 'baseline value'

```
In [28]: plt.figure(figsize = (16,8))
         plt.suptitle("Checking for Outliers and Distribution pattern of variables", fontsize = 'x-large',weight = 'extra bold',ha = "center")
         plt.subplot(2,5,1)
         sns.boxplot(fetal['prolongued_decelerations'], color = 'green')
         plt.subplot(2,5,2)
         sns.boxplot(fetal['abnormal_short_term_variability'], color = 'blue')
         plt.subplot(2,5,3)
         sns.boxplot(fetal['mean_value_of_short_term_variability'], color = 'purple')
         plt.subplot(2,5,4)
         sns.boxplot(fetal['percentage_of_time_with_abnormal_long_term_variability'], color = 'red')
         plt.subplot(2,5,5)
         sns.boxplot(fetal['mean_value_of_long_term_variability'], color = 'violet')
         plt.subplot(2,5,6)
         sns.distplot(fetal['prolongued_decelerations'], color = 'green')
         plt.subplot(2,5,7)
         sns.distplot(fetal['abnormal_short_term_variability'], color = 'blue')
         plt.subplot(2,5,8)
         sns.distplot(fetal['mean_value_of_short_term_variability'], color = 'purple')
         plt.subplot(2,5,9)
         sns.distplot(fetal['percentage_of_time_with_abnormal_long_term_variability'], color = 'red')
         plt.subplot(2,5,10)
         sns.distplot(fetal['mean_value_of_long_term_variability'], color = 'violet')
         plt.show()
```



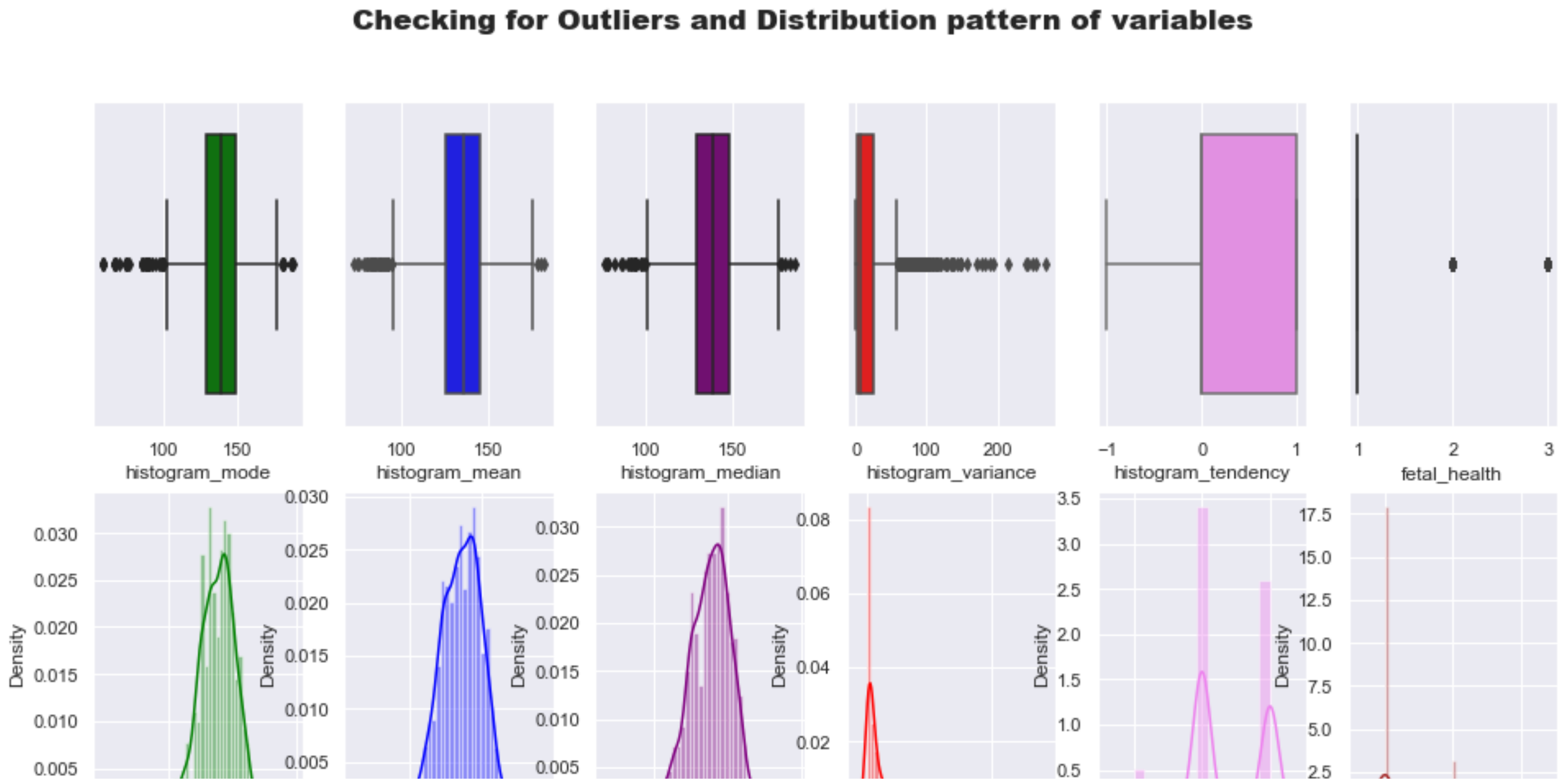
- From the above picture it is clear that all the featureas are right skewed and has outliers except 'abnormal_short_term_variability'

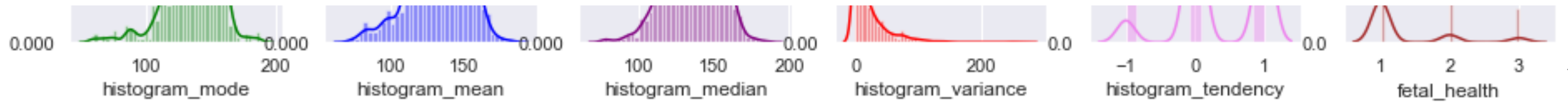
In [29]:
```
plt.figure(figsize = (16,8))
plt.suptitle("Checking for Outliers and Distribution pattern of variables", fontsize = 'x-large',weight = 'extra bold',ha = "center")
plt.subplot(2,5,1)
sns.boxplot(fetal['histogram_width'], color = 'green')
plt.subplot(2,5,2)
sns.boxplot(fetal['histogram_min'], color = 'blue')
plt.subplot(2,5,3)
sns.boxplot(fetal['histogram_max'], color = 'purple')
plt.subplot(2,5,4)
sns.boxplot(fetal['histogram_number_of_peaks'], color = 'red')
plt.subplot(2,5,5)
sns.boxplot(fetal['histogram_number_of_zeroes'], color = 'violet')
plt.subplot(2,5,6)
sns.distplot(fetal['histogram_width'], color = 'green')
plt.subplot(2,5,7)
sns.distplot(fetal['histogram_min'], color = 'blue')
plt.subplot(2,5,8)
sns.distplot(fetal['histogram_max'], color = 'purple')
plt.subplot(2,5,9)
sns.distplot(fetal['histogram_number_of_peaks'], color = 'red')
plt.subplot(2,5,10)
sns.distplot(fetal['histogram_number_of_zeroes'], color = 'violet')
plt.show()
```



- From the above visual it is clear that the features 'histigram width', 'histogram min' has no outliers remaining features has outliers and right skewness.

```
In [30]: plt.figure(figsize = (16,8))
         plt.suptitle("Checking for Outliers and Distribution pattern of variables", fontsize = 'x-large',weight = 'extra bold',ha = "center")
         plt.subplot(2,6,1)
         sns.boxplot(fetal['histogram_mode'], color = 'green')
         plt.subplot(2,6,2)
         sns.boxplot(fetal['histogram_mean'], color = 'blue')
         plt.subplot(2,6,3)
         sns.boxplot(fetal['histogram_median'], color = 'purple')
         plt.subplot(2,6,4)
         sns.boxplot(fetal['histogram_variance'], color = 'red')
         plt.subplot(2,6,5)
         sns.boxplot(fetal['histogram_tendency'], color = 'violet')
         plt.subplot(2,6,6)
         sns.boxplot(fetal['fetal_health'], color = 'brown')
         plt.subplot(2,6,7)
         sns.distplot(fetal['histogram_mode'], color = 'green')
         plt.subplot(2,6,8)
         sns.distplot(fetal['histogram_mean'], color = 'blue')
         plt.subplot(2,6,9)
         sns.distplot(fetal['histogram_median'], color = 'purple')
         plt.subplot(2,6,10)
         sns.distplot(fetal['histogram_variance'], color = 'red')
         plt.subplot(2,6,11)
         sns.distplot(fetal['histogram_tendency'], color = 'violet')
         plt.subplot(2,6,12)
         sns.distplot(fetal['fetal_health'], color = 'brown')
         plt.show()
```





Even though , outliers capping is important , I don't want to capping outliers, because each and evry value in the health data set is import tant to analise patient's health.

If we change the value to new value, then there is no meaning to that particular record.

In this project, I would like to continue without outlier capping of data

Minority Class Oversampling

```
In [31]: x = fetal.iloc[:,0:21]
         y = fetal.iloc[:,-1]
```

```
In [32]: from imblearn.over_sampling import RandomOverSampler
         oversample = RandomOverSampler(sampling_strategy='not majority')
         x_over, y_over = oversample.fit_resample(x, y)
```

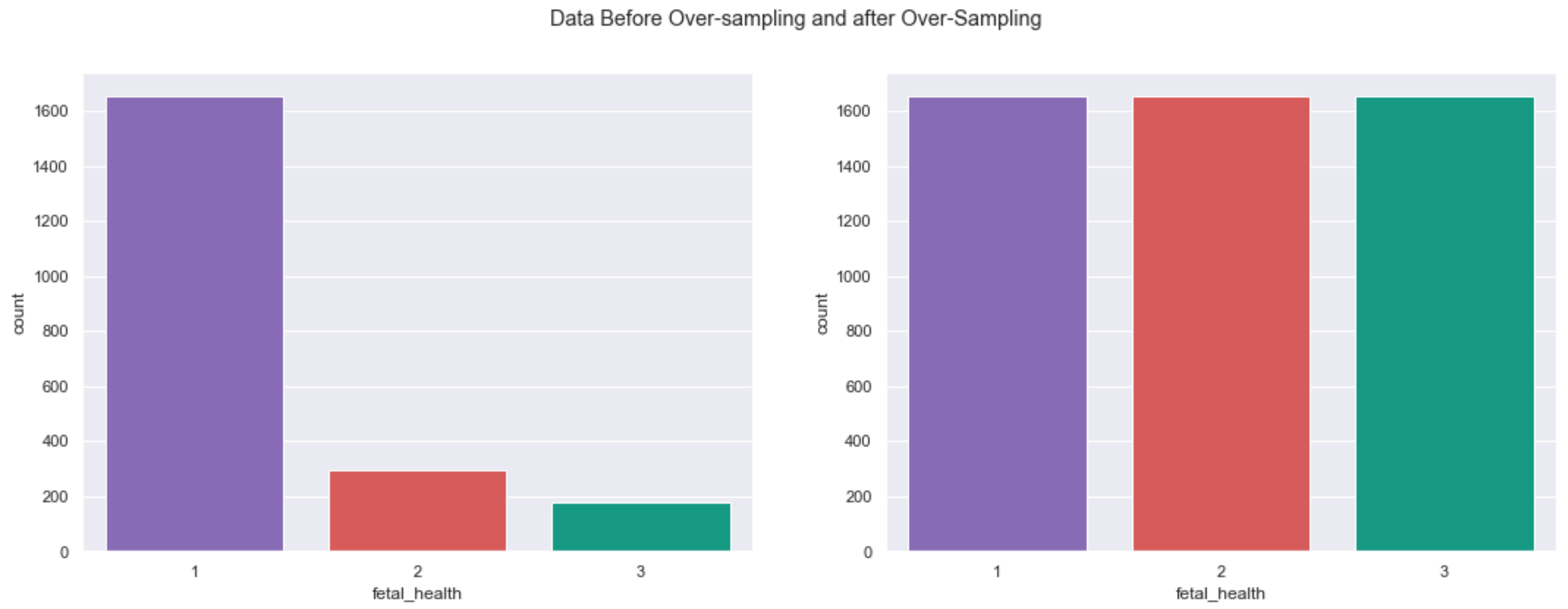
```
In [33]: x_over.shape

Out[33]: (4965, 21)
```

```
In [34]: y_over.shape

Out[34]: (4965,)
```

In [35]:
```
plt.figure(figsize=(18,6))
plt.suptitle('Data Before Over-sampling and after Over-Sampling')
plt.subplot(1,2,1)
sns.countplot(x=fetal['fetal_health'], palette=['#845ec2','#ec4646','#00af91'])
plt.subplot(1,2,2)
sns.countplot(data=pd.DataFrame(y_over),x='fetal_health', palette=['#845ec2','#ec4646','#00af91'] )
plt.show()
```



- From te above picture we can see that all the classes in out put feture are balanced.

Feature Scalling

In [36]:
```
from sklearn.preprocessing import StandardScaler
sdd = StandardScaler()
scaled_fetal = sdd.fit_transform(x_over)
```

In [37]:
```
scaled_fetal
```

Out[37]:
```
array([[-1.52261655, -0.52042455, -0.23379762, ..., -0.77138339,
         1.13918796,  1.14402939],
       [-0.32021627,  1.4995686 , -0.23379762, ...,  0.26415691,
        -0.30603768, -0.3865839 ],
       [-0.22001624,  0.48957202, -0.23379762, ...,  0.15515266,
        -0.28234545, -0.3865839 ],
       ...,
       [-0.22001624, -0.52042455, -0.23379762, ..., -1.75242157,
         1.66041688, -0.3865839 ],
       [ 1.28298411, -0.52042455, -0.15181638, ...,  0.80917812,
        -0.59034436,  1.14402939],
       [ 1.48338415, -0.52042455, -0.16821263, ...,  1.08168872,
        -0.56665214, -0.3865839 ]])
```

In [38]:
```
new_fetal = pd.DataFrame(scaled_fetal, columns = x_over.columns)
```

In [39]:
```
new_fetal
```

Out[39]:

	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_wit
0	-1.522617	-0.520425	-0.233798	-1.131941	-0.615206	-0.096703	-0.466968	0.982911	-0.707679	
1	-0.320216	1.499569	-0.233798	0.731792	0.315003	-0.096703	-0.466968	-2.294205	0.895893	
2	-0.220016	0.489572	-0.233798	1.353037	0.315003	-0.096703	-0.466968	-2.352725	0.895893	
3	-0.119816	0.489572	-0.233798	1.353037	0.315003	-0.096703	-0.466968	-2.352725	1.196563	
4	-0.320216	1.836234	-0.233798	1.353037	-0.615206	-0.096703	-0.466968	-2.352725	1.196563	
...	...	...	...	...	...	...	...	...	...	
4960	-1.522617	-0.520425	-0.233798	-0.821319	1.555282	-0.096703	1.509584	0.456232	-0.106340	
4961	-0.019616	-0.520425	-0.020646	-0.821319	-0.615206	-0.096703	-0.466968	0.748831	-0.908126	
4962	-0.220016	-0.520425	-0.233798	1.974281	0.315003	-0.096703	1.509584	0.280672	1.597456	
4963	1.282984	-0.520425	-0.151816	-0.510697	-0.615206	-0.096703	-0.466968	0.924391	-0.908126	
4964	1.483384	-0.520425	-0.168213	-1.131941	-0.615206	-0.096703	-0.466968	1.509590	-0.807902	

4965 rows × 21 columns

In [40]:
```
import pickle
import requests
import json
import joblib
```

In [41]:
```
joblib.dump(sdd, "sdd.joblib")  # Dumping standard Scaling for model deployment
```

Out[41]:
```
['sdd.joblib']
```

Train_Test_Split

In [42]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_over, y_over,train_size = 0.70, shuffle = True, stratify = y_over)
```

Model Building

Logistic Regression

In [43]:
```
from sklearn.linear_model import LogisticRegression
```

In [44]:
```
model1 = LogisticRegression(multi_class='multinomial', solver = 'newton-cg', random_state = 0)
model1.fit(x_train, y_train)
```

Out[44]:
```
▾                     LogisticRegression

LogisticRegression(multi_class='multinomial', random_state=0,
                   solver='newton-cg')
```

In [45]:
```
lr_y_pred = model1.predict(x_test)
```

In [46]:
```
from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(y_test, lr_y_pred)                     # Confusion Matrix
cm1.ravel()
```

Out[46]:
```
array([394,  88,  14,  50, 384,  63,   5,  46, 446], dtype=int64)
```

In [47]:
```
from sklearn.metrics import log_loss

pred1 = model1.predict_proba(x_test)
logloss1 = log_loss(y_test,pred1)                           # Log_loss Score
```

In [48]:
```
from sklearn.metrics import accuracy_score                  # Accuracy_Score

acc1 = accuracy_score(y_test, lr_y_pred)
```

- When LogisticRegression(solver = 'lbfgs', random_state = 0) the accuray score is 75%
- When LogisticRegression(multi_class='multinomial', solver = 'lbfgs', random_state = 0) the accuray score is 77%
- When LogisticRegression(multi_class='multinomial', solver = 'newton-cg', random_state = 0) the accuray score is 83%
- When LogisticRegression(multi_class='ovr', solver = 'newton-cg', random_state = 0) the accuray score is 81%

> Among all parameter tuning multi_class='multinomial', solver = 'newton-cg' gives best accuracy score. So, I choosen third one for this model.

In [49]:
```
from sklearn import metrics                                  # Precision_Score

precision_positive1 = metrics.precision_score(y_test, lr_y_pred, pos_label = 'positive', average='micro')

precision_negative1 = metrics.precision_score(y_test, lr_y_pred, pos_label = 'negative', average='micro')
```

In [50]:
```
recall_sensitivity1 = metrics.recall_score(y_test, lr_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

recall_specificity1 = metrics.recall_score(y_test, lr_y_pred, pos_label = 'negative', average='micro')
```

In [51]:
```
f1_positive1 = metrics.f1_score(y_test, lr_y_pred, pos_label = 'positive', average='micro')            # f1_Score

f1_negative1 = metrics.f1_score(y_test, lr_y_pred, pos_label = 'negative', average='micro')
```

- Use micro-averaging score when there is a need to weight each instance or prediction equally.
- Use macro-averaging score when all classes need to be treated equally to evaluate the overall performance of the classifier with regard to the most frequent class labels.

In [52]:
```
#roc auc score
from sklearn.metrics import roc_curve, roc_auc_score
rocscore1 = roc_auc_score(y_test, pred1, multi_class='ovr', average= 'weighted')
```

- For Ploting ROC curves for all models is time consuming and will take too much space. Sinc, I would like to plot multiclass auc curve for best model later on.

Note:

OvR and OvO strategies can (and should) be used to adapt any binary classification metric to the multiclass classification task.

Evaluating OvO and OvR results also can help understanding which classes the model is struggling to describe, and which features you can add or remove to improve the result of the model.

Cross Validation of Logistic Regression Model Using Stratified K_fold method

- Stratified kfold cross validation is an extension of regular kfold cross validation but specifically for classification problems where rather than the splits being completely random, the ratio between the target classes is the same in each fold as it is in the full dataset.
- Stratified kfold cross validation is typically useful when we have imbalanced data and where the data size is on the small side. Sometimes we will over or under sample our data to deal with class imbalance but other times we want to maintain the class imbalance when it’s representative of or contains some information about what we are trying to classify.
- When the data is large enough we can still use regular kfold cross validation as this will often preserve the class ratios but this becomes less likely with fewer training examples.
- Roughly and succinctly speaking (depending of course on the test, training and validation scheme you employ), things like std. dev. can be used as a rudimentary measure of classifier stability. If you you are using stratified k-fold cross validation (depending on how it is implemented), then the mean *could* represent the mean of the aggregated correct classification results over k models (there are other ways to interpret it of course). The std. dev would then give an indication as to how these varied across the different models. If you are not interested in outright classification performance, but merely want to compare the stability of two different learners (again in a rather rudimentary fashion), you could use something like relative std. dev. = std dev/correct classification percentage.

> In this project, I have choosen Stratified kfold cross validation technique, because the ratio between the target classes is the same in each fold as it is in the full dataset after performing the "RandomOversampling" technique.

```
In [53]: from sklearn.model_selection import StratifiedKFold
         from sklearn.model_selection import cross_val_score
```

```
In [54]: skf = StratifiedKFold(n_splits = 10, shuffle = True, random_state = None)
```

```
In [55]: lr_cvscr = cross_val_score(model1, x, y, cv = skf)
```

```
In [56]: lr_cvscr
```

```
Out[56]: array([0.85915493, 0.87793427, 0.85915493, 0.87323944, 0.92957746,
                0.89671362, 0.89150943, 0.85377358, 0.88679245, 0.86792453])
```

```
In [57]: # Check the Accuracy
         lr_cvscr_Accuracy =  lr_cvscr * 100.0
         lr_cvscr_Accuracy
```

```
Out[57]: array([85.91549296, 87.79342723, 85.91549296, 87.32394366, 92.95774648,
                89.6713615 , 89.1509434 , 85.37735849, 88.67924528, 86.79245283])
```

```
In [58]: lr_cvscr_Accuracy_mean = lr_cvscr_Accuracy.mean()          # Average of Crossvalidation score
```

```
In [59]: lr_cvscr_Accuracy_std = lr_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

KNeighbours Clssifier

```
In [60]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [61]: model2 = KNeighborsClassifier(n_neighbors = 3)
```

```
In [62]: model2.fit(x_train, y_train)
```

```
Out[62]: ▾       KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=3)
```

```
In [63]: kn_y_pred = model2.predict(x_test)
```

```
In [64]: cm2 = confusion_matrix(y_test, kn_y_pred)
         cm2.ravel()
```

```
Out[64]: array([442,  48,   6,   8, 485,   4,   0,   2, 495], dtype=int64)
```

```
In [65]: pred2 = model2.predict_proba(x_test)
         logloss2 = log_loss(kn_y_pred, pred2)             #logloss score
```

```
In [66]: acc2 = accuracy_score(y_test, kn_y_pred)
```

- When Parameter is set to n_neighbors = 3 gives 95%, n_neighbors = 5 gives 92%, n_neighbors = 7 gives 91%. So, i kept first parameter for best accuracy score

In [67]:
```
precision_positive2 = metrics.precision_score(y_test, kn_y_pred, pos_label = 'positive', average='micro')

precision_negative2 = metrics.precision_score(y_test, kn_y_pred, pos_label = 'negative', average='micro')
```

In [68]:
```
recall_sensitivity2 = metrics.recall_score(y_test, kn_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

recall_specificity2 = metrics.recall_score(y_test, kn_y_pred, pos_label = 'negative', average='micro')
```

In [69]:
```
f1_positive2 = metrics.f1_score(y_test, kn_y_pred, pos_label = 'positive', average='micro')            # f1_Score

f1_negative2 = metrics.f1_score(y_test, kn_y_pred, pos_label = 'negative', average='micro')
```

In [70]:
```
rocscore2 = roc_auc_score(y_test, pred2, multi_class='ovr', average= 'weighted')
```

Crossvalidation of KNeighborsClassifier Model

In [71]:
```
kn_cvscr = cross_val_score(model2, x, y, cv = skf)
```

In [72]:
```
# Check the Accuracy
kn_cvscr_Accuracy =  kn_cvscr * 100.0
kn_cvscr_Accuracy
```

Out[72]:
```
array([94.83568075, 88.73239437, 92.01877934, 91.54929577, 90.14084507,
       92.48826291, 87.73584906, 89.1509434 , 88.20754717, 92.45283019])
```

In [73]:
```
kn_cvscr_Accuracy_mean = kn_cvscr_Accuracy.mean()          # Average of Crossvalidation score
```

In [74]:
```
kn_cvscr_Accuracy_std = kn_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

SupportVectorMachine

In [75]:
```
from sklearn.svm import SVC
model3 = SVC(kernel = 'rbf', decision_function_shape='ovr', gamma ='auto',  probability = True)
```

In [76]:
```
model3.fit(x_train, y_train)
```

Out[76]:
```
▾                SVC
SVC(gamma='auto', probability=True)
```

In [77]:
```
svc_y_pred = model3.predict(x_test)
```

In [78]:
```
cm3 = confusion_matrix(y_test, svc_y_pred)
cm3.ravel()
```

Out[78]:
```
array([492,   3,   1,   0, 494,   3,   0,   0, 497], dtype=int64)
```

In [79]:
```
acc3 = accuracy_score(y_test, svc_y_pred)
```

- 1 When SVC(kernel = 'rbf', decision_function_shape='ovr'), the accuracy score is 81%
- 2 When svc(kernel = 'linear', decision_function_shape='ovr'), the accuracy score is 84%
- 3 When svc(kernel = 'poly', decision_function_shape='ovr'), the accuracy score is 82%
- 4 When svc(kernel = 'sigmoid', decision_function_shape='ovr'), the accuracy score is 32% means very poor.
- 5 When svc(kernel = 'rbf', decision_function_shape='ovr', gamma='scale'), the accuracy score is 81%
- 6 When svc(kernel = 'rbf', decision_function_shape='ovr', gamma='auto'), the accuracy score is 98%
- 7 When svc(kernel = 'linear', decision_function_shape='ovr', gamma='scale'), the accuracy score is 84%
- 8 When svc(kernel = 'linear', decision_function_shape='ovr', gamma='auto'), the accuracy score is 84%
- 9 When svc(kernel = 'poly', decision_function_shape='ovr', gamma='scale'), the accuracy score is 82%
- 10 When svc(kernel = 'poly', decision_function_shape='ovr', gamma='auto'), the accuracy score is 90%
- 11 When svc(kernel = 'sigmoid', decision_function_shape='ovr', gamma='scale'), the accuracy score is 31%
- 12 When svc(kernel = 'sigmoid', decision_function_shape='ovr', gamma='auto'), the accuracy score is 33%

> Among all above parameter tuning 6th one gives best accuracy score, For this model i selected 6th one.

In [80]:
```
pred3 = model3.predict_proba(x_test)
logloss3 = log_loss(svc_y_pred, pred3)            #logloss score
```

In [81]:
```
precision_positive3 = metrics.precision_score(y_test, svc_y_pred, pos_label = 'positive', average='micro')

precision_negative3 = metrics.precision_score(y_test, svc_y_pred, pos_label = 'negative', average='micro')
```

In [82]:
```
recall_sensitivity3 = metrics.recall_score(y_test, svc_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

recall_specificity3 = metrics.recall_score(y_test, svc_y_pred, pos_label = 'negative', average='micro')
```

In [83]:
```
f1_positive3 = metrics.f1_score(y_test, svc_y_pred, pos_label = 'positive', average='micro')           # f1_Score

f1_negative3 = metrics.f1_score(y_test, svc_y_pred, pos_label = 'negative', average='micro')
```

In [84]:
```
rocscore3 = roc_auc_score(y_test, pred3, multi_class='ovr', average= 'weighted')
```

Cross Validation of SVM

In [85]:
```
svc_cvscr = cross_val_score(model3, x, y, cv = skf)
```

In [86]:
```
# Check the Accuracy
svc_cvscr_Accuracy =  svc_cvscr * 100.0
svc_cvscr_Accuracy
```

Out[86]:
```
array([79.342723  , 79.342723  , 81.22065728, 78.40375587, 79.342723  ,
       81.22065728, 82.0754717 , 80.66037736, 80.66037736, 79.24528302])
```

In [87]:
```
svc_cvscr_Accuracy_mean = svc_cvscr_Accuracy.mean()          # Average of Crossvalidation score
```

In [88]:
```
svc_cvscr_Accuracy_std = svc_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

Decision Tree

In [89]:
```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion = 'log_loss')
```

In [90]:
```
model4.fit(x_train,y_train)
```

Out[90]:
```
▾           DecisionTreeClassifier
DecisionTreeClassifier(criterion='log_loss')
```

In [91]:
```
dt_y_pred = model4.predict(x_test)
```

In [92]:
```
cm4 = confusion_matrix(y_test, dt_y_pred)
cm4.ravel()
```

Out[92]:
```
array([472,  18,   6,   0, 497,   0,   0,   0, 497], dtype=int64)
```

In [93]:
```
acc4 = accuracy_score(y_test, dt_y_pred)
```

- When DecisionTreeClassifier(criterion = 'entropy'), the accuracy is 98%
- When DecisionTreeClassifier(criterion = 'gini'), the accuracy is 97%
- When DecisionTreeClassifier(criterion = 'log_loss'), the accuracy is 98%

> Among all above three, I've Choosen last one.

In [94]:
```
pred4 = model4.predict_proba(x_test)
logloss4 = log_loss(dt_y_pred, pred4)             #logloss score
```

In [95]:
```
precision_positive4 = metrics.precision_score(y_test, dt_y_pred, pos_label = 'positive', average='micro')

precision_negative4 = metrics.precision_score(y_test, dt_y_pred, pos_label = 'negative', average='micro')
```

```
In [96]: recall_sensitivity4 = metrics.recall_score(y_test, dt_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

         recall_specificity4 = metrics.recall_score(y_test, dt_y_pred, pos_label = 'negative', average='micro')
```

```
In [97]: f1_positive4 = metrics.f1_score(y_test, dt_y_pred, pos_label = 'positive', average='micro')          # f1_Score

         f1_negative4 = metrics.f1_score(y_test, dt_y_pred, pos_label = 'negative', average='micro')
```

```
In [98]: rocscore4 = roc_auc_score(y_test, pred4, multi_class='ovr', average= 'weighted')
```

Crossvalidation of DecisionTreeClasssifier

```
In [99]: dt_cvscr = cross_val_score(model4, x, y, cv = skf)
```

```
In [100]: # Check the Accuracy
          dt_cvscr_Accuracy =  dt_cvscr * 100.0
          dt_cvscr_Accuracy

Out[100]: array([90.14084507, 92.95774648, 93.42723005, 93.89671362, 91.07981221,
                 95.77464789, 94.81132075, 94.33962264, 93.86792453, 92.45283019])
```

```
In [101]: dt_cvscr_Accuracy_mean = dt_cvscr_Accuracy.mean()          # Average of Crossvalidation score
```

```
In [102]: dt_cvscr_Accuracy_std = dt_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

NaiveBayes

```
In [103]: from sklearn.naive_bayes import GaussianNB
          model5 = GaussianNB()
```

```
In [104]: model5.fit(x_train, y_train)

Out[104]: ▾ GaussianNB
          GaussianNB()
```

```
In [105]: nb_y_pred = model5.predict(x_test)
```

```
In [106]: cm5 = confusion_matrix(y_test, nb_y_pred)
          cm5.ravel()

Out[106]: array([362, 113,  21,  30, 435,  32,  21, 142, 334], dtype=int64)
```

```
In [107]: acc5 = accuracy_score(y_test, nb_y_pred)
```

```
In [108]: pred5 = model5.predict_proba(x_test)
          logloss5 = log_loss(nb_y_pred, pred5)            #logloss score
```

In [109]:
```
precision_positive5 = metrics.precision_score(y_test, nb_y_pred, pos_label = 'positive', average='micro')

precision_negative5 = metrics.precision_score(y_test, nb_y_pred, pos_label = 'negative', average='micro')

recall_sensitivity5 = metrics.recall_score(y_test, nb_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

recall_specificity5 = metrics.recall_score(y_test, nb_y_pred, pos_label = 'negative', average='micro')

f1_positive5 = metrics.f1_score(y_test, nb_y_pred, pos_label = 'positive', average='micro')            # f1_Score

f1_negative5 = metrics.f1_score(y_test, nb_y_pred, pos_label = 'negative', average='micro')

rocscore5 = roc_auc_score(y_test, pred5, multi_class='ovr', average= 'weighted')
```

Crossvalidation of Naive Bayes

In [110]:
```
nb_cvscr = cross_val_score(model5, x, y, cv = skf)
```

In [111]:
```
# Check the Accuracy
nb_cvscr_Accuracy =  nb_cvscr * 100.0
nb_cvscr_Accuracy
```

Out[111]:
```
array([80.75117371, 82.62910798, 82.62910798, 80.28169014, 83.09859155,
       76.99530516, 80.66037736, 82.0754717 , 78.30188679, 85.37735849])
```

In [112]:
```
nb_cvscr_Accuracy_mean = nb_cvscr_Accuracy.mean()          # Average of Crossvalidation score
nb_cvscr_Accuracy_std = nb_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

Randomforesrt

In [113]:
```
from sklearn.ensemble import RandomForestClassifier
model6 = RandomForestClassifier(n_estimators = 100, criterion = 'log_loss')
```

In [114]:
```
model6.fit(x_train, y_train)
```

Out[114]:
```
▾            RandomForestClassifier

RandomForestClassifier(criterion='log_loss')
```

In [115]:
```
rf_y_pred = model6.predict(x_test)
```

In [116]:
```
cm6 = confusion_matrix(y_test, rf_y_pred)
cm6.ravel()
```

Out[116]:
```
array([478,  13,   5,   0, 497,   0,   0,   0, 497], dtype=int64)
```

In [117]:
```
acc6 = accuracy_score(y_test, rf_y_pred)
```

- When RandomForestClassifier(n_estimators = 100, criterion = 'gini'), the accuracy is 98.5%

- When RandomForestClassifier(n_estimators = 100, criterion = 'enropy'), the accuracy is 98.7%
- When RandomForestClassifier(n_estimators = 100, criterion = 'log_loss'), the accuracy is 98.6%

> Among all of the above, I have choosen second one.

In [118]:
```
pred6 = model6.predict_proba(x_test)
logloss6 = log_loss(rf_y_pred, pred6)            #logloss score
```

In [119]:
```
precision_positive6 = metrics.precision_score(y_test, rf_y_pred, pos_label = 'positive', average='micro')

precision_negative6 = metrics.precision_score(y_test, rf_y_pred, pos_label = 'negative', average='micro')

recall_sensitivity6 = metrics.recall_score(y_test, rf_y_pred, pos_label = 'positive', average='micro')  # Recall_Score

recall_specificity6 = metrics.recall_score(y_test, rf_y_pred, pos_label = 'negative', average='micro')

f1_positive6 = metrics.f1_score(y_test, rf_y_pred, pos_label = 'positive', average='micro')           # f1_Score

f1_negative6 = metrics.f1_score(y_test, rf_y_pred, pos_label = 'negative', average='micro')

rocscore6 = roc_auc_score(y_test, pred6, multi_class='ovr', average= 'weighted')
```

Crossvalidation of Randomforest

In [120]:
```
rf_cvscr = cross_val_score(model6, x, y, cv = skf)
```

In [121]:
```
# Check the Accuracy
rf_cvscr_Accuracy =  kn_cvscr * 100.0
rf_cvscr_Accuracy
```

Out[121]:
```
array([94.83568075, 88.73239437, 92.01877934, 91.54929577, 90.14084507,
       92.48826291, 87.73584906, 89.1509434 , 88.20754717, 92.45283019])
```

In [122]:
```
rf_cvscr_Accuracy_mean = rf_cvscr_Accuracy.mean()          # Average of Crossvalidation score

rf_cvscr_Accuracy_std = rf_cvscr_Accuracy.std()            # Standard deviation of crossvalidation score.
```

Display of All scores

In [123]:
```
data = {"Accuracy_Score":[acc1,acc2,acc3,acc4,acc5,acc6],
        "log_loss_Score":[logloss1,logloss2,logloss3,logloss4,logloss5,logloss6],
        "F1_Score_Positive":[f1_positive1,f1_positive2,f1_positive3,f1_positive4,f1_positive5,f1_positive6],
        "F1_Score_Nagative":[f1_negative1,f1_negative2,f1_negative3,f1_negative4,f1_negative5,f1_negative6],
        "Precision_Positive":[precision_positive1,precision_positive2,precision_positive3,precision_positive4,precision_positive5,precision_positive6],
        "Precision_Nagative":[precision_negative1,precision_negative2,precision_negative3,precision_negative4,precision_negative5,precision_negative6],
        "Recall_sensitivity":[recall_sensitivity1,recall_sensitivity2,recall_sensitivity3,recall_sensitivity4,recall_sensitivity5,recall_sensitivity6],
        "Recall_specificity":[recall_specificity1,recall_specificity2,recall_specificity3,recall_specificity4,recall_specificity5,recall_specificity6],
        "ROC_Score":[rocscore1,rocscore2,rocscore3,rocscore4,rocscore5,rocscore6],
        "Cross_Val Score":[lr_cvscr_Accuracy_mean,kn_cvscr_Accuracy_mean,svc_cvscr_Accuracy_mean,dt_cvscr_Accuracy_mean,nb_cvscr_Accuracy_mean,rf_cvscr_Accuracy_mean],
        "Cross_Val Std":[lr_cvscr_Accuracy_std,kn_cvscr_Accuracy_std,svc_cvscr_Accuracy_std,dt_cvscr_Accuracy_std,nb_cvscr_Accuracy_std,rf_cvscr_Accuracy_std]}
All_scores = pd.DataFrame(data, index = ["LR","KNN","SVM","DT","NB","RF"])
```

In [124]:
```
All_scores
```

Out[124]:

	Accuracy_Score	log_loss_Score	F1_Score_Positive	F1_Score_Nagative	Precision_Positive	Precision_Nagative	Recall_sensitivity	Recall_specificity	ROC_Score	Cross_Val Score	Cross_Val Std
LR	0.821477	4.445686e-01	0.821477	0.821477	0.821477	0.821477	0.821477	0.821477	0.943090	87.957746	2.160263
KNN	0.954362	2.013719e-02	0.954362	0.954362	0.954362	0.954362	0.954362	0.954362	0.981083	90.731243	2.177317
SVM	0.995302	1.276138e-02	0.995302	0.995302	0.995302	0.995302	0.995302	0.995302	0.998256	80.151475	1.111034
DT	0.983893	2.109424e-15	0.983893	0.983893	0.983893	0.983893	0.983893	0.983893	0.987915	93.274869	1.610018
NB	0.759060	5.232484e-02	0.759060	0.759060	0.759060	0.759060	0.759060	0.759060	0.887387	81.280007	2.307738
RF	0.987919	3.457891e-02	0.987919	0.987919	0.987919	0.987919	0.987919	0.987919	0.999538	90.731243	2.177317

Conclusion

- In this project i have applied Classification algorithms to find out best model to prdict output.
- Among all the models i have selected SVM on the basis of Accuracy score and Cross Validation score.

In [125]:
```
joblib.dump(model3, "regressor.pkl")
```

Out[125]:
```
['regressor.pkl']
```

In [126]:
```
check = fetal.copy()
```

In [127]:
```
check
```

Out[127]:

	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
0	120	0.000	0.000	0.000	0.000	0.0	0.0	73	0.5	
1	132	0.006	0.000	0.006	0.003	0.0	0.0	17	2.1	
2	133	0.003	0.000	0.008	0.003	0.0	0.0	16	2.1	
3	134	0.003	0.000	0.008	0.003	0.0	0.0	16	2.4	
4	132	0.007	0.000	0.008	0.000	0.0	0.0	16	2.4	
...	...	...	...	...	...	...	...	...	...	
2121	140	0.000	0.000	0.007	0.000	0.0	0.0	79	0.2	
2122	140	0.001	0.000	0.007	0.000	0.0	0.0	78	0.4	
2123	140	0.001	0.000	0.007	0.000	0.0	0.0	79	0.4	
2124	140	0.001	0.000	0.006	0.000	0.0	0.0	78	0.4	
2125	142	0.002	0.002	0.008	0.000	0.0	0.0	74	0.4	

2126 rows × 22 columns

In [128]:
```
check.drop("fetal_health",axis=1,inplace=True)
```

In [129]:
```
check
```

Out[129]:

	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
0	120	0.000	0.000	0.000	0.000	0.0	0.0	73	0.5	
1	132	0.006	0.000	0.006	0.003	0.0	0.0	17	2.1	
2	133	0.003	0.000	0.008	0.003	0.0	0.0	16	2.1	
3	134	0.003	0.000	0.008	0.003	0.0	0.0	16	2.4	
4	132	0.007	0.000	0.008	0.000	0.0	0.0	16	2.4	
...	...	...	...	...	...	...	...	...	...	
2121	140	0.000	0.000	0.007	0.000	0.0	0.0	79	0.2	
2122	140	0.001	0.000	0.007	0.000	0.0	0.0	78	0.4	
2123	140	0.001	0.000	0.007	0.000	0.0	0.0	79	0.4	
2124	140	0.001	0.000	0.006	0.000	0.0	0.0	78	0.4	
2125	142	0.002	0.002	0.008	0.000	0.0	0.0	74	0.4	

2126 rows × 21 columns

In [130]:
```
final = check.iloc[:1,:]
```

In [131]:
```
final
```

Out[131]:

	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_ab
0	120	0.0	0.0	0.0	0.0	0.0	0.0	73	0.5	

1 rows × 21 columns

In [132]:
```
final.columns
```

Out[132]:
```
Index(['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'],
      dtype='object')
```

In [133]:
```
final_model = joblib.load("regressor.pkl")
final_model
```

Out[133]:
```
▾                 SVC
SVC(gamma='auto', probability=True)
```

In [134]:
```
final_model.predict(final)
```

Out[134]:
```
array([2], dtype=int64)
```

In [ ]: