Hyper-parameter tuning methods such as grid search and random search are Markdown as they take a lot of time to execute.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.utils import resample
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.formula.api import ols
import statsmodels.api as sm
from scipy.stats import chi2_contingency

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_columns',None)
```

In [2]:
```
data = pd.read_csv('fetal_health.csv')
```

In [3]:
```
data.shape
```

Out[3]:
```
(2126, 22)
```

In [4]:
```
data.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   float64
 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   float64
 8   mean_value_of_short_term_variability                    2126 non-null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  2126 non-null   float64
 10  mean_value_of_long_term_variability                     2126 non-null   float64
 11  histogram_width                                         2126 non-null   float64
 12  histogram_min                                           2126 non-null   float64
 13  histogram_max                                           2126 non-null   float64
 14  histogram_number_of_peaks                               2126 non-null   float64
 15  histogram_number_of_zeroes                              2126 non-null   float64
 16  histogram_mode                                          2126 non-null   float64
 17  histogram_mean                                          2126 non-null   float64
 18  histogram_median                                        2126 non-null   float64
 19  histogram_variance                                      2126 non-null   float64
 20  histogram_tendency                                      2126 non-null   float64
 21  fetal_health                                            2126 non-null   float64
dtypes: float64(22)
memory usage: 365.5 KB
```

In [5]:
```
data.columns
```

Out[5]:
```
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',
       'fetal_health'],
      dtype='object')
```

In [6]:
```
data.rename(columns = {'baseline value':'baseline',
                       'abnormal_short_term_variability':'abnrml_ST_variability',
                       'mean_value_of_short_term_variability':'mean_ST_variability',
                      'percentage_of_time_with_abnormal_long_term_variability':'abnrml_LT_variability',
                       'mean_value_of_long_term_variability':'mean_of_LT_variability',
                       'histogram_width':'hist_width',
                       'histogram_min':'hist_min',
                       'histogram_max':'hist_max',
                       'histogram_number_of_peaks':'hist_no_of_peaks',
                       'histogram_number_of_zeroes':'hist_no_of_zeroes',
                       'histogram_mode':'hist_mode',
                       'histogram_mean':'hist_mean',
                       'histogram_median':'hist_median',
                       'histogram_variance':'hist_var',
                       'histogram_tendency':'hist_tendency'},inplace = True)
```

In [7]:
```
data.fetal_health = data.fetal_health.astype(int)
```

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 22 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   baseline                  2126 non-null   float64
 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   abnrml_ST_variability     2126 non-null   float64
 8   mean_ST_variability       2126 non-null   float64
 9   abnrml_LT_variability     2126 non-null   float64
 10  mean_of_LT_variability    2126 non-null   float64
 11  hist_width                2126 non-null   float64
 12  hist_min                  2126 non-null   float64
 13  hist_max                  2126 non-null   float64
 14  hist_no_of_peaks          2126 non-null   float64
 15  hist_no_of_zeroes         2126 non-null   float64
 16  hist_mode                 2126 non-null   float64
 17  hist_mean                 2126 non-null   float64
 18  hist_median               2126 non-null   float64
 19  hist_var                  2126 non-null   float64
 20  hist_tendency             2126 non-null   float64
 21  fetal_health              2126 non-null   int32
dtypes: float64(21), int32(1)
memory usage: 357.2 KB
```

In [9]:
```
data.describe()
```

Out[9]:

	baseline	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnrml_ST_variat
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.00(
mean	133.303857	0.003178	0.009481	0.004366	0.001889	0.000003	0.000159	46.99(
std	9.840844	0.003866	0.046666	0.002946	0.002960	0.000057	0.000590	17.19:
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	12.00(
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000	32.00(
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000	49.00(
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	0.000000	61.00(
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000	87.00(

In [10]:
```
data.head()
```

Out[10]:

	baseline	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnrml_ST_variability	m
0	120.0	0.000	0.0	0.000	0.000	0.0	0.0	73.0	
1	132.0	0.006	0.0	0.006	0.003	0.0	0.0	17.0	
2	133.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0	
3	134.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0	
4	132.0	0.007	0.0	0.008	0.000	0.0	0.0	16.0	

NULL VALUES

In [11]:
```
data.isnull().sum()
```

Out[11]:
```
baseline                    0
accelerations               0
fetal_movement              0
uterine_contractions        0
light_decelerations         0
severe_decelerations        0
prolongued_decelerations    0
abnrml_ST_variability       0
mean_ST_variability         0
abnrml_LT_variability       0
mean_of_LT_variability      0
hist_width                  0
hist_min                    0
hist_max                    0
hist_no_of_peaks            0
hist_no_of_zeroes           0
hist_mode                   0
hist_mean                   0
hist_median                 0
hist_var                    0
hist_tendency               0
fetal_health                0
dtype: int64
```

In [12]:
```
data.isnull().sum().sum() #no null values
```

Out[12]:
```
0
```

In [13]:
```
data.fetal_health = data.fetal_health.replace({1 : 0, 2 : 1, 3 : 2})
```

In [14]:
```
df = data.copy()
```

EDA

fetal_health - target variable

0 - Normal 1 - Suspect 2 - Pathelogical

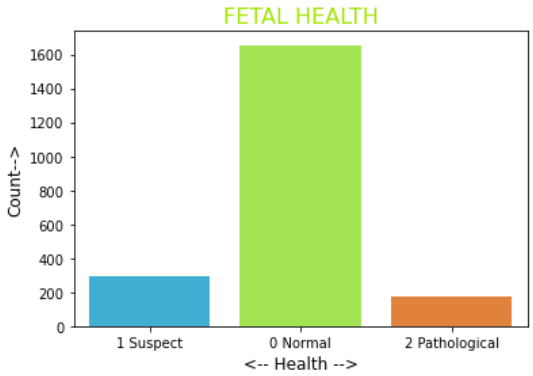
In [15]:
```
df.fetal_health = df.fetal_health.replace({0 : '0 Normal', 1 : '1 Suspect', 2 : '2 Pathological'})
```

In [16]:
```
pd.DataFrame(df.fetal_health.value_counts())
```

Out[16]:

	fetal_health
0 Normal	1655
1 Suspect	295
2 Pathological	176

In [17]:
```
sns.countplot(x = df.fetal_health,palette = 'turbo');
plt.title('FETAL HEALTH',color = '#a2e400',size = 16);
plt.xlabel('<-- Health -->',size = 12);
plt.ylabel('Count-->',size = 12);
```



1) baseline

Baseline Fetal Heart Rate (FHR)

#continuous

In [18]:
```
df.baseline.nunique()
```

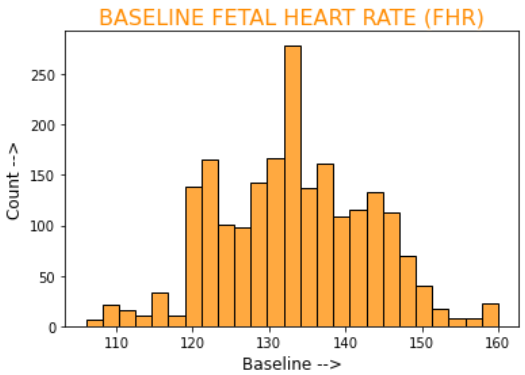
Out[18]: 48

In [19]:
```
pd.DataFrame(df.baseline.describe())
```

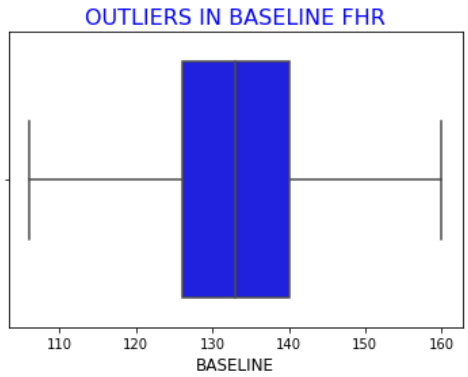
Out[19]:

	baseline
count	2126.000000
mean	133.303857
std	9.840844
min	106.000000
25%	126.000000
50%	133.000000
75%	140.000000
max	160.000000

In [20]:
```
sns.histplot(x = df.baseline,color = 'darkorange');
plt.title('BASELINE FETAL HEART RATE (FHR)',size = 16,color = 'darkorange');
plt.xlabel('Baseline -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [21]:
```
sns.boxplot(x = df.baseline,color = 'blue');
plt.title('OUTLIERS IN BASELINE FHR',color = 'blue',size = 16)
plt.xlabel('BASELINE',size = 12);
```



An abnormal baseline is termed bradycardia when the baseline FHR is less than 110 bpm; it is termed tachycardia when the baseline FHR is greater than 160 bpm."

In [22]:
```
df[df.baseline < 110]
```

Out[22]:

	baseline	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnrml_ST_variability
1659	106.0	0.011	0.0	0.009	0.0	0.0	0.0	62.0
1660	106.0	0.000	0.0	0.010	0.0	0.0	0.0	63.0
1661	106.0	0.000	0.0	0.010	0.0	0.0	0.0	63.0
1662	106.0	0.001	0.0	0.011	0.0	0.0	0.0	63.0
1663	106.0	0.000	0.0	0.010	0.0	0.0	0.0	63.0
1664	106.0	0.000	0.0	0.009	0.0	0.0	0.0	64.0
1665	106.0	0.006	0.0	0.006	0.0	0.0	0.0	64.0

In [23]:
```
df[df.baseline > 160]
```

Out[23]:

	baseline	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnrml_ST_variability	me

2) accelerations

Number of accelerations per second(heart rate)

#continuous

In [24]:
```
df.accelerations.nunique()
```

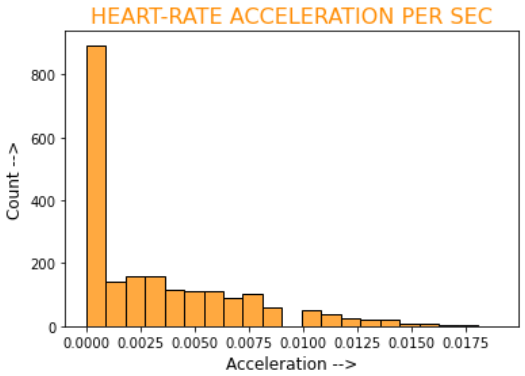
Out[24]: 20

In [25]:
```
pd.DataFrame(df.accelerations.describe())
```

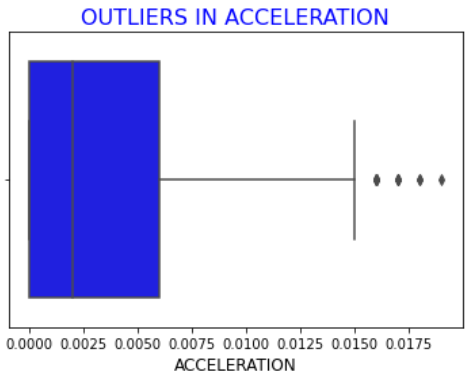
Out[25]:

	accelerations
count	2126.000000
mean	0.003178
std	0.003866
min	0.000000
25%	0.000000
50%	0.002000
75%	0.006000
max	0.019000

In [26]:
```
sns.histplot(x = df.accelerations,color = 'darkorange');
plt.title('HEART-RATE ACCELERATION PER SEC',size = 16,color = 'darkorange');
plt.xlabel('Acceleration -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [27]:
```
sns.boxplot(x = df.accelerations,color = 'blue');
plt.title('OUTLIERS IN ACCELERATION',color = 'blue',size = 16)
plt.xlabel('ACCELERATION',size = 12);
```



3) fetal_movement

Number of fetal movements per second

#continuous

In [28]:
```
df.fetal_movement.nunique()
```

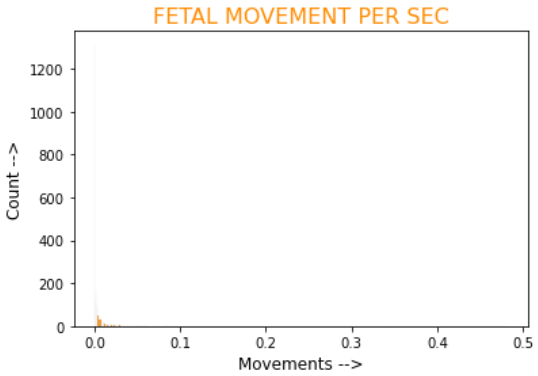
Out[28]: 102

In [29]:
```
pd.DataFrame(df.fetal_movement.describe())
```

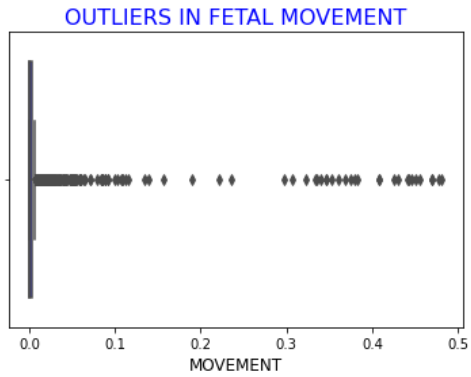
Out[29]:

	fetal_movement
count	2126.000000
mean	0.009481
std	0.046666
min	0.000000
25%	0.000000
50%	0.000000
75%	0.003000
max	0.481000

In [30]:
```
sns.histplot(x = df.fetal_movement,color = 'darkorange');
plt.title('FETAL MOVEMENT PER SEC',size = 16,color = 'darkorange');
plt.xlabel('Movements -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [31]:
```
sns.boxplot(x = df.fetal_movement,color = 'blue');
plt.title('OUTLIERS IN FETAL MOVEMENT',color = 'blue',size = 16)
plt.xlabel('MOVEMENT',size = 12);
```



In [32]:
```
df.drop(['fetal_movement'],axis = 1,inplace =True) #can't use this column
```

4)uterine_contractions

Number of uterine contractions per second

#continuous

In [33]:
```
df.uterine_contractions.nunique()
```

Out[33]: 16

In [34]:
```
pd.DataFrame(df.uterine_contractions.value_counts().sort_index()) #its still continuous
```

Out[34]:

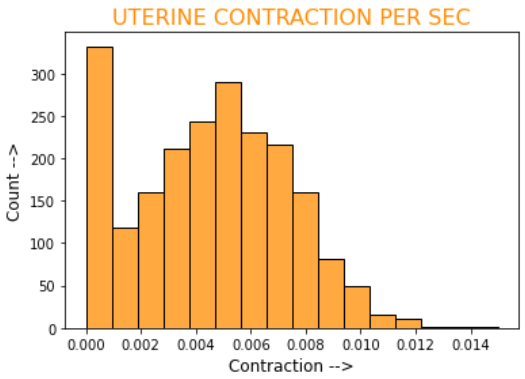
	uterine_contractions
0.000	332
0.001	118
0.002	160
0.003	212
0.004	244
0.005	290
0.006	231
0.007	216
0.008	160
0.009	82
0.010	49
0.011	16
0.012	11
0.013	2
0.014	2
0.015	1

In [35]:
```
pd.DataFrame(df.uterine_contractions.describe())
```

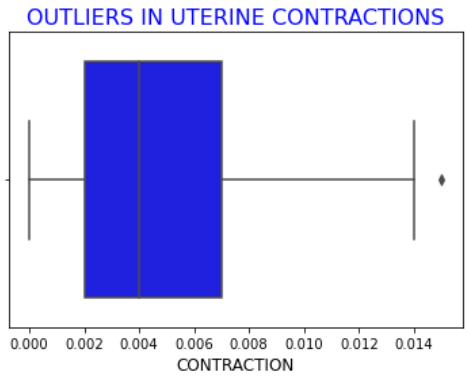
Out[35]:

	uterine_contractions
count	2126.000000
mean	0.004366
std	0.002946
min	0.000000
25%	0.002000
50%	0.004000
75%	0.007000
max	0.015000

In [36]:
```
sns.histplot(x = df.uterine_contractions,bins = 16,color = 'darkorange');
plt.title('UTERINE CONTRACTION PER SEC',size = 16,color = 'darkorange');
plt.xlabel('Contraction -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [37]:
```
sns.boxplot(x = df.uterine_contractions,color = 'blue');
plt.title('OUTLIERS IN UTERINE CONTRACTIONS',color = 'blue',size = 16)
plt.xlabel('CONTRACTION',size = 12);
```



5) light_decelerations

Number of Light decelerations per second

#continuous

In [38]:
```
df.light_decelerations.nunique()
```

Out[38]: 16

In [39]:
```
pd.DataFrame(df.light_decelerations.value_counts().sort_index())
```

Out[39]:

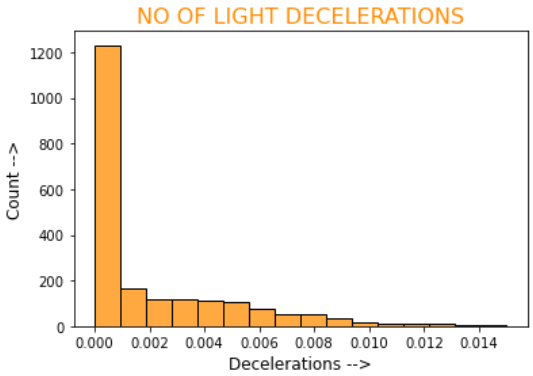
	light_decelerations
0.000	1231
0.001	163
0.002	115
0.003	118
0.004	114
0.005	107
0.006	74
0.007	54
0.008	55
0.009	37
0.010	15
0.011	13
0.012	12
0.013	8
0.014	7
0.015	3

In [40]:
```
pd.DataFrame(df.light_decelerations.describe())
```

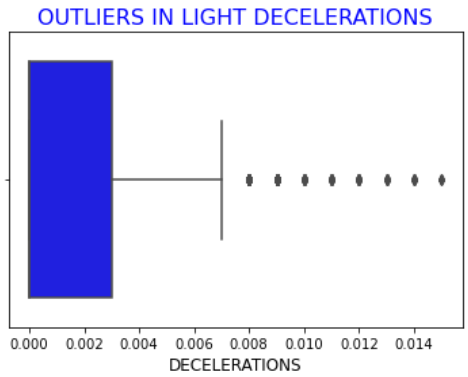
Out[40]:

	light_decelerations
count	2126.000000
mean	0.001889
std	0.002960
min	0.000000
25%	0.000000
50%	0.000000
75%	0.003000
max	0.015000

In [41]:
```
sns.histplot(x = df.light_decelerations,bins = 16,color = 'darkorange');
plt.title('NO OF LIGHT DECELERATIONS',size = 16,color = 'darkorange');
plt.xlabel('Decelerations -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [42]:
```
sns.boxplot(x = df.light_decelerations,color = 'blue');
plt.title('OUTLIERS IN LIGHT DECELERATIONS',color = 'blue',size = 16)
plt.xlabel('DECELERATIONS',size = 12);
```



6) severe_decelerations

Number of Severe Decelerations per second

In [43]:
```
df.severe_decelerations.nunique()
```

Out[43]: 2

In [44]:
```
pd.DataFrame(df.severe_decelerations.value_counts())
```

Out[44]:

	severe_decelerations
0.000	2119
0.001	7

In [45]:
```
df.drop(['severe_decelerations'],axis = 1,inplace = True)
```

7) prolongued_decelerations

Number of Prolongued Decelerations per second

#continuous

In [46]:
```
df.prolongued_decelerations.nunique()
```

Out[46]: 6

In [47]:
```
pd.DataFrame(df.prolongued_decelerations.value_counts().sort_index())
```

Out[47]:

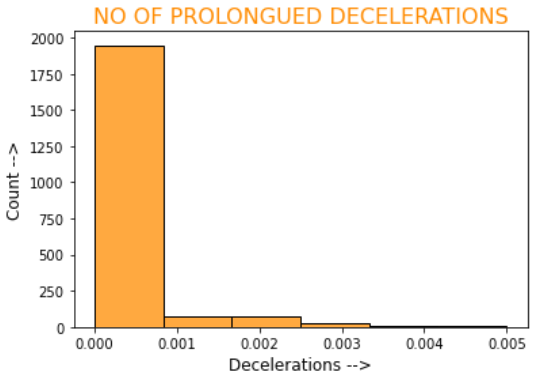
	prolongued_decelerations
0.000	1948
0.001	70
0.002	72
0.003	24
0.004	9
0.005	3

In [48]:
```
pd.DataFrame(df.prolongued_decelerations.describe())
```

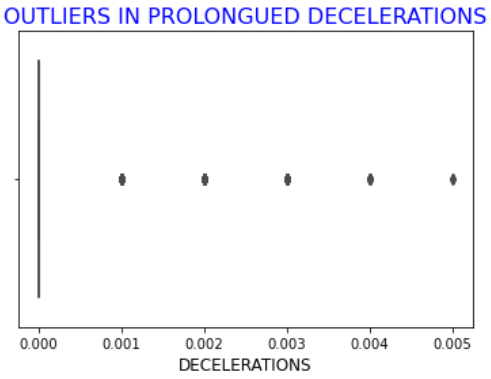
Out[48]:

	prolongued_decelerations
count	2126.000000
mean	0.000159
std	0.000590
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	0.005000

In [49]:
```
sns.histplot(x = df.prolongued_decelerations,bins = 6,color = 'darkorange');
plt.title('NO OF PROLONGUED DECELERATIONS',size = 16,color = 'darkorange');
plt.xlabel('Decelerations -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [50]:
```
sns.boxplot(x = df.prolongued_decelerations,color = 'blue');
plt.title('OUTLIERS IN PROLONGUED DECELERATIONS',color = 'blue',size = 16)
plt.xlabel('DECELERATIONS',size = 12);
```



In [51]:
```
df.drop(['prolongued_decelerations'],axis = 1,inplace = True)
```

8) abnrml_ST_variability

Percentage of time with abnormal short term variability

#cont

In [52]:
```
df.abnrml_ST_variability.nunique()
```

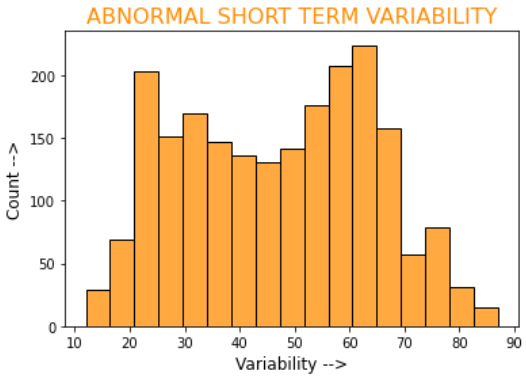
Out[52]: 75

In [53]:
```
pd.DataFrame(df.abnrml_ST_variability.describe())
```

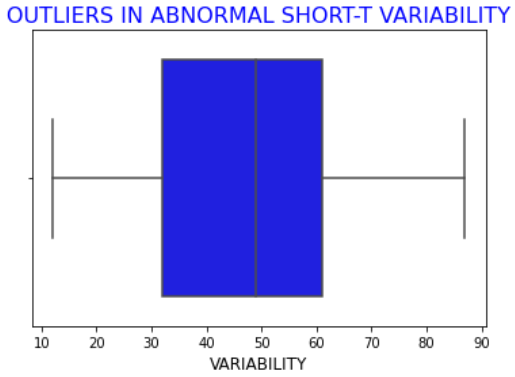
Out[53]:

	abnrml_ST_variability
count	2126.000000
mean	46.990122
std	17.192814
min	12.000000
25%	32.000000
50%	49.000000
75%	61.000000
max	87.000000

In [54]:
```
sns.histplot(x = df.abnrml_ST_variability,color = 'darkorange');
plt.title('ABNORMAL SHORT TERM VARIABILITY',size = 16,color = 'darkorange');
plt.xlabel('Variability -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [55]:
```
sns.boxplot(x = df.abnrml_ST_variability,color = 'blue');
plt.title('OUTLIERS IN ABNORMAL SHORT-T VARIABILITY',color = 'blue',size = 16)
plt.xlabel('VARIABILITY',size = 12);
```



9) mean_ST_variability

Mean value of short term variability

#continuous

In [56]:
```
df.mean_ST_variability.nunique()
```

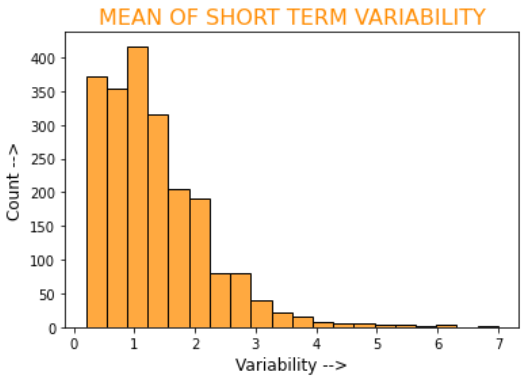
Out[56]: 57

In [57]:
```
pd.DataFrame(df.mean_ST_variability.describe())
```

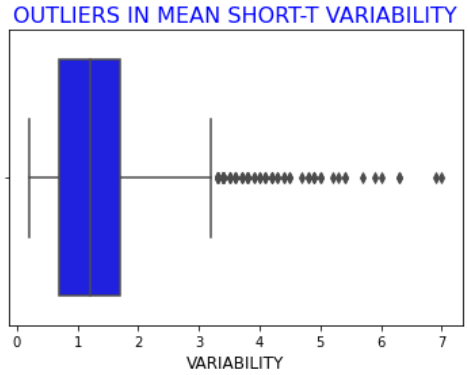
Out[57]:

	mean_ST_variability
count	2126.000000
mean	1.332785
std	0.883241
min	0.200000
25%	0.700000
50%	1.200000
75%	1.700000
max	7.000000

In [58]:
```
sns.histplot(x = df.mean_ST_variability,bins = 20,color = 'darkorange');
plt.title('MEAN OF SHORT TERM VARIABILITY',size = 16,color = 'darkorange');
plt.xlabel('Variability -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [59]:
```
sns.boxplot(x = df.mean_ST_variability,color = 'blue');
plt.title('OUTLIERS IN MEAN SHORT-T VARIABILITY',color = 'blue',size = 16)
plt.xlabel('VARIABILITY',size = 12);
```



10) abnrml_LT_variability

Percentage of time with abnormal long term variability #cont

In [60]:
```
df.abnrml_LT_variability.nunique()
```

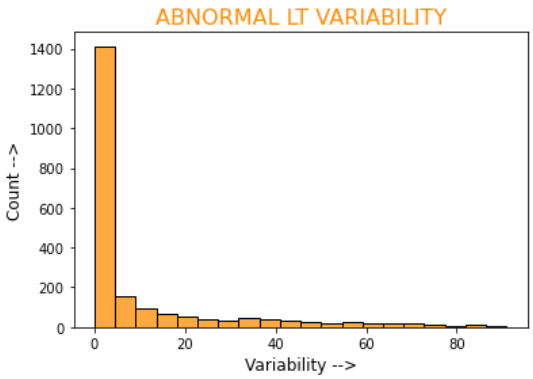
Out[60]: 87

In [61]:
```
pd.DataFrame(df.abnrml_LT_variability.describe())
```

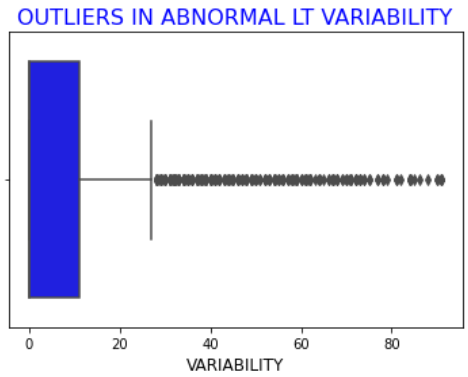
Out[61]:

	abnrml_LT_variability
count	2126.00000
mean	9.84666
std	18.39688
min	0.00000
25%	0.00000
50%	0.00000
75%	11.00000
max	91.00000

In [62]:
```
sns.histplot(x = df.abnrml_LT_variability,bins = 20,color = 'darkorange');
plt.title('ABNORMAL LT VARIABILITY',size = 16,color = 'darkorange');
plt.xlabel('Variability -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [63]:
```
sns.boxplot(x = df.abnrml_LT_variability,color = 'blue');
plt.title('OUTLIERS IN ABNORMAL LT VARIABILITY',color = 'blue',size = 16)
plt.xlabel('VARIABILITY',size = 12);
```



11) mean_of_LT_variability

Mean value of long term variability

In [64]:
```
df.mean_of_LT_variability.nunique()
```

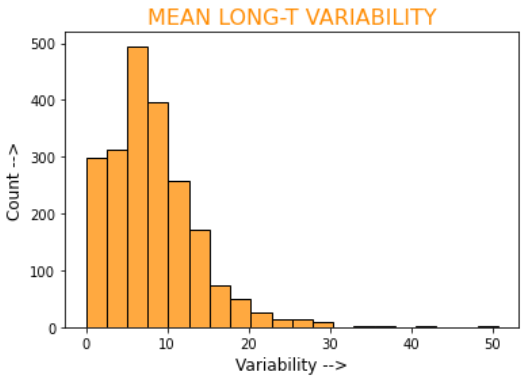
Out[64]: 249

In [65]:
```
pd.DataFrame(df.mean_of_LT_variability.describe())
```

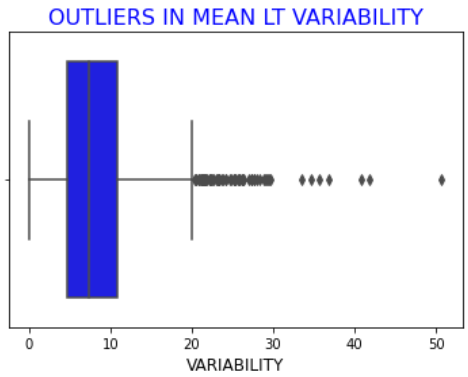
Out[65]:

	mean_of_LT_variability
count	2126.000000
mean	8.187629
std	5.628247
min	0.000000
25%	4.600000
50%	7.400000
75%	10.800000
max	50.700000

In [66]:
```
sns.histplot(x = df.mean_of_LT_variability,bins = 20,color = 'darkorange');
plt.title('MEAN LONG-T VARIABILITY',size = 16,color = 'darkorange');
plt.xlabel('Variability -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [67]:
```
sns.boxplot(x = df.mean_of_LT_variability,color = 'blue');
plt.title('OUTLIERS IN MEAN LT VARIABILITY',color = 'blue',size = 16)
plt.xlabel('VARIABILITY',size = 12);
```



12) hist_width

In [68]:
```
df.hist_width.nunique()
```

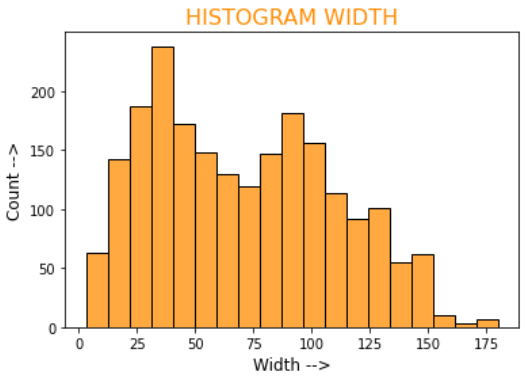
Out[68]: 154

In [69]:
```
pd.DataFrame(df.hist_width.describe())
```

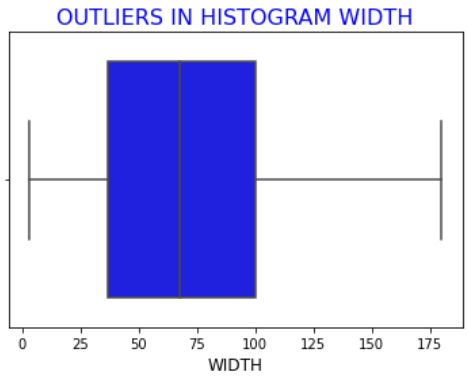
Out[69]:

	hist_width
count	2126.000000
mean	70.445908
std	38.955693
min	3.000000
25%	37.000000
50%	67.500000
75%	100.000000
max	180.000000

In [70]:
```
sns.histplot(x = df.hist_width,color = 'darkorange');
plt.title('HISTOGRAM WIDTH',size = 16,color = 'darkorange');
plt.xlabel('Width -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [71]:
```
sns.boxplot(x = df.hist_width,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM WIDTH',color = 'blue',size = 16)
plt.xlabel('WIDTH',size = 12);
```



13) hist_min

In [72]:
```
df.hist_min.nunique()
```

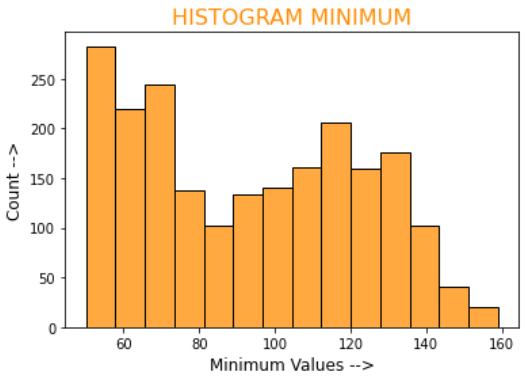
Out[72]: 109

In [73]:
```
pd.DataFrame(df.hist_min.describe())
```

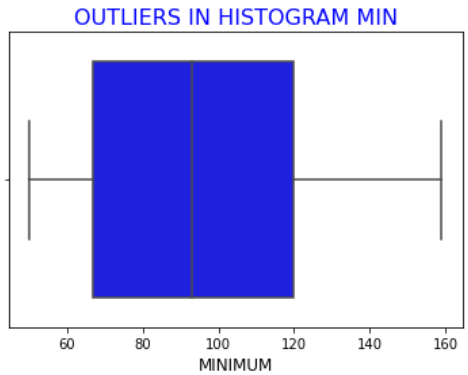
Out[73]:

	hist_min
count	2126.000000
mean	93.579492
std	29.560212
min	50.000000
25%	67.000000
50%	93.000000
75%	120.000000
max	159.000000

In [74]:
```
sns.histplot(x = df.hist_min,color = 'darkorange');
plt.title('HISTOGRAM MINIMUM',size = 16,color = 'darkorange');
plt.xlabel('Minimum Values -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [75]:
```
sns.boxplot(x = df.hist_min,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM MIN',color = 'blue',size = 16)
plt.xlabel('MINIMUM',size = 12);
```



14) hist_max

Histogram maximum value

In [76]:
```
df.hist_max.nunique()
```

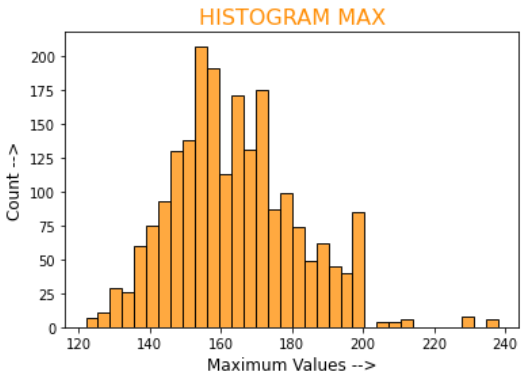
Out[76]: 86

In [77]:
```
pd.DataFrame(df.hist_max.describe())
```

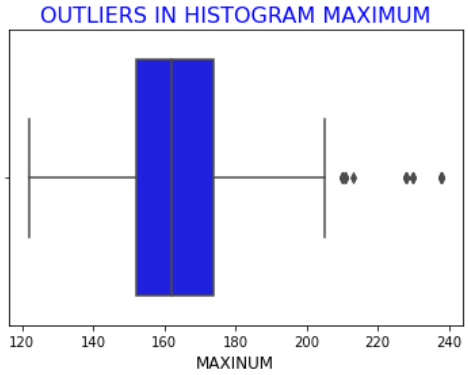
Out[77]:

	hist_max
count	2126.000000
mean	164.025400
std	17.944183
min	122.000000
25%	152.000000
50%	162.000000
75%	174.000000
max	238.000000

In [78]:
```
sns.histplot(x = df.hist_max,color = 'darkorange');
plt.title('HISTOGRAM MAX',size = 16,color = 'darkorange');
plt.xlabel('Maximum Values -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [79]:
```
sns.boxplot(x = df.hist_max,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM MAXIMUM',color = 'blue',size = 16)
plt.xlabel('MAXINUM',size = 12);
```



15) hist_no_of_peaks

Number of peaks in the exam histogram

In [80]:
```
df.hist_no_of_peaks.nunique()
```

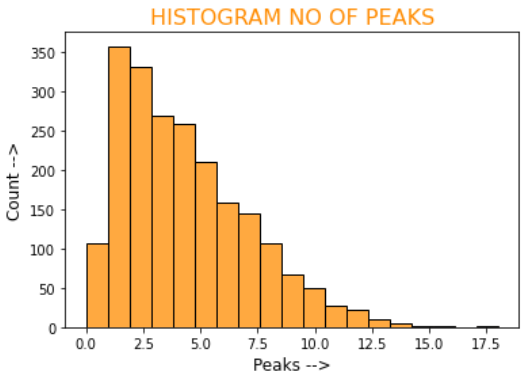
Out[80]: 18

In [81]:
```
pd.DataFrame(df.hist_no_of_peaks.describe())
```

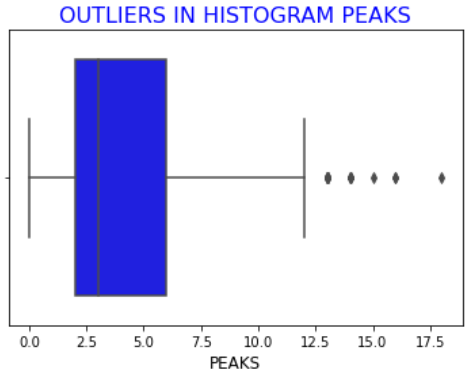
Out[81]:

	hist_no_of_peaks
count	2126.000000
mean	4.068203
std	2.949386
min	0.000000
25%	2.000000
50%	3.000000
75%	6.000000
max	18.000000

In [82]:
```
sns.histplot(x = df.hist_no_of_peaks,bins = 19,color = 'darkorange');
plt.title('HISTOGRAM NO OF PEAKS',size = 16,color = 'darkorange');
plt.xlabel('Peaks -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [83]:
```
sns.boxplot(x = df.hist_no_of_peaks,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM PEAKS',color = 'blue',size = 16)
plt.xlabel('PEAKS',size = 12);
```



16) hist_no_of_zeroes

Number of zeroes in the exam histogram #cat

In [84]:
```
df.hist_no_of_zeroes.nunique()
```

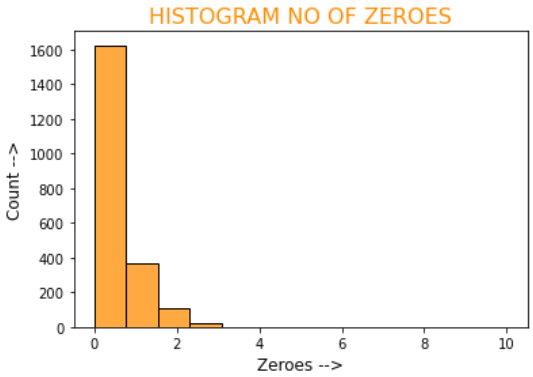
Out[84]: 9

In [85]:
```
pd.DataFrame(df.hist_no_of_zeroes.value_counts().sort_index())
```

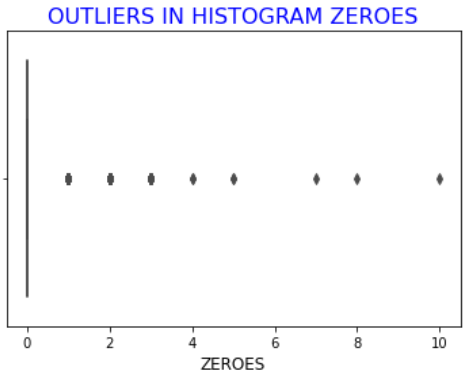
Out[85]:

	hist_no_of_zeroes
0.0	1624
1.0	366
2.0	108
3.0	21
4.0	2
5.0	2
7.0	1
8.0	1
10.0	1

In [86]:
```
sns.histplot(x = df.hist_no_of_zeroes,color = 'darkorange');
plt.title('HISTOGRAM NO OF ZEROES',size = 16,color = 'darkorange');
plt.xlabel('Zeroes -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [87]:
```
sns.boxplot(x = df.hist_no_of_zeroes,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM ZEROES',color = 'blue',size = 16)
plt.xlabel('ZEROES',size = 12);
```



In [88]:
```
df.drop(['hist_no_of_zeroes'],axis = 1,inplace = True)
```

17) hist_mode

#CONT

In [89]:
```
df.hist_mode.nunique()
```

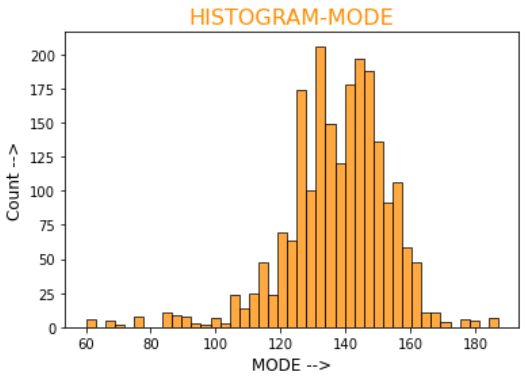
Out[89]: 88

In [90]:
```
pd.DataFrame(df.hist_mode.describe())
```

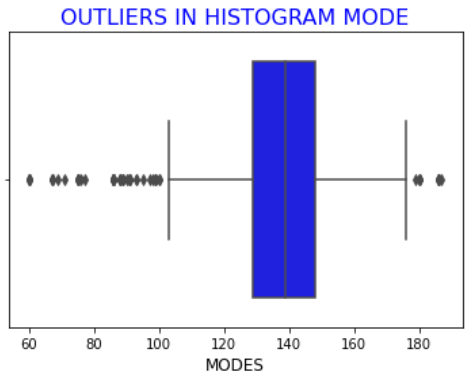
Out[90]:

	hist_mode
count	2126.000000
mean	137.452023
std	16.381289
min	60.000000
25%	129.000000
50%	139.000000
75%	148.000000
max	187.000000

In [91]:
```
sns.histplot(x = df.hist_mode,color = 'darkorange');
plt.title('HISTOGRAM-MODE',size = 16,color = 'darkorange');
plt.xlabel('MODE -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [92]:
```
sns.boxplot(x = df.hist_mode,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM MODE',color = 'blue',size = 16)
plt.xlabel('MODES',size = 12);
```



18) hist_mean

Hist mean

In [93]:
```
df.hist_mean.nunique()
```

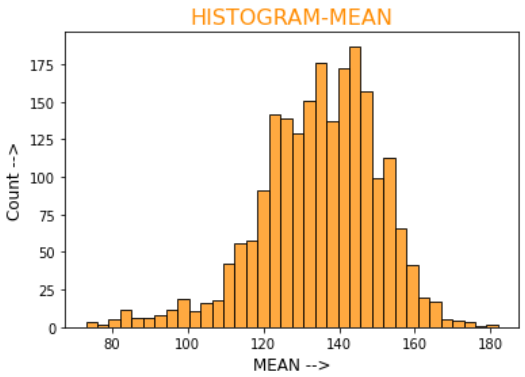
Out[93]: 103

In [94]:
```
pd.DataFrame(df.hist_mean.describe())
```

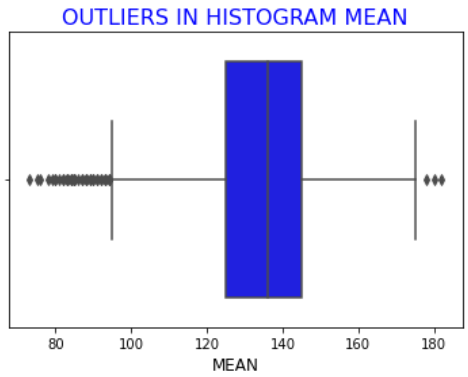
Out[94]:

	hist_mean
count	2126.000000
mean	134.610536
std	15.593596
min	73.000000
25%	125.000000
50%	136.000000
75%	145.000000
max	182.000000

In [95]:
```
sns.histplot(x = df.hist_mean,color = 'darkorange');
plt.title('HISTOGRAM-MEAN',size = 16,color = 'darkorange');
plt.xlabel('MEAN -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [96]:
```
sns.boxplot(x = df.hist_mean,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM MEAN',color = 'blue',size = 16)
plt.xlabel('MEAN',size = 12);
```



19) hist_median

Hist Median

In [97]:
```
df.hist_median.nunique()
```

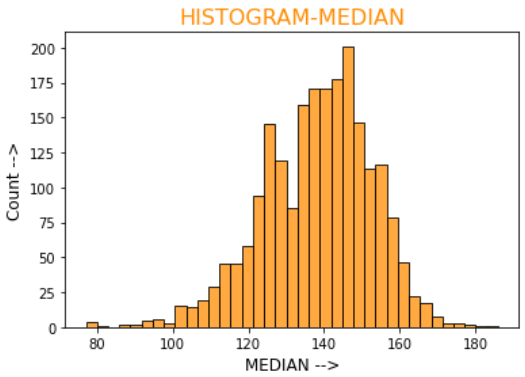
Out[97]: 95

In [98]:
```
pd.DataFrame(df.hist_median.describe())
```

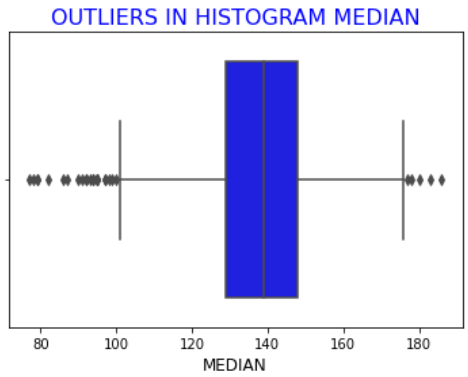
Out[98]:

	hist_median
count	2126.000000
mean	138.090310
std	14.466589
min	77.000000
25%	129.000000
50%	139.000000
75%	148.000000
max	186.000000

In [99]:
```
sns.histplot(x = df.hist_median,color = 'darkorange');
plt.title('HISTOGRAM-MEDIAN',size = 16,color = 'darkorange');
plt.xlabel('MEDIAN -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [100]:
```
sns.boxplot(x = df.hist_median,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM MEDIAN',color = 'blue',size = 16)
plt.xlabel('MEDIAN',size = 12);
```



20) hist_var

Hist variance

In [101]:
```
df.hist_var.nunique()
```

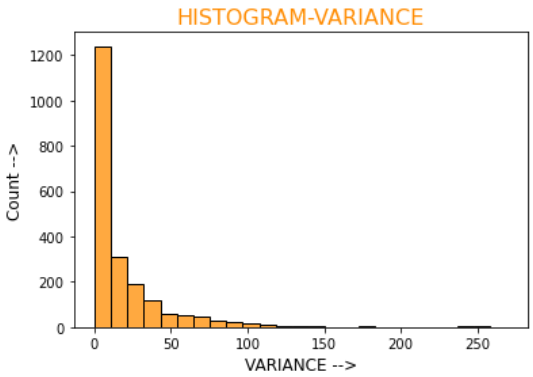
Out[101]: 133

In [102]:
```
pd.DataFrame(df.hist_var.describe())
```

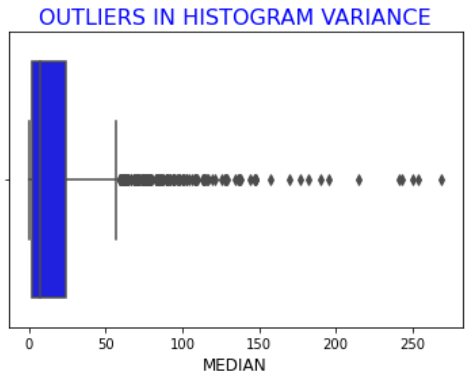
Out[102]:

	hist_var
count	2126.000000
mean	18.808090
std	28.977636
min	0.000000
25%	2.000000
50%	7.000000
75%	24.000000
max	269.000000

In [103]:
```
sns.histplot(x = df.hist_var,bins = 25,color = 'darkorange');
plt.title('HISTOGRAM-VARIANCE',size = 16,color = 'darkorange');
plt.xlabel('VARIANCE -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [104]:
```
sns.boxplot(x = df.hist_var,color = 'blue');
plt.title('OUTLIERS IN HISTOGRAM VARIANCE',color = 'blue',size = 16)
plt.xlabel('MEDIAN',size = 12);
```



21) hist_tendency

Histogram trend:

-1 : Decelerating , 0 : Stable , 1 : Accelerating

In [105]:
```
df.hist_tendency.nunique()
```

Out[105]: 3

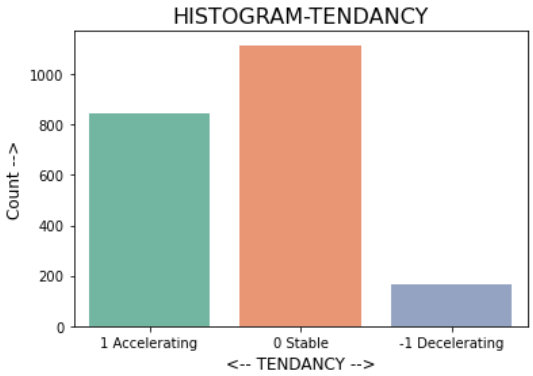
In [106]:
```
df.hist_tendency = df.hist_tendency.replace({-1.0:'-1 Decelerating', 0.0:'0 Stable', 1.0:'1 Accelerating'})
```

In [107]:
```
pd.DataFrame(df.hist_tendency.value_counts().sort_index())
```

Out[107]:

	hist_tendency
-1 Decelerating	165
0 Stable	1115
1 Accelerating	846

In [108]:
```
sns.countplot(x = df.hist_tendency,palette = 'Set2');
plt.title('HISTOGRAM-TENDANCY',size = 16);
plt.xlabel('<-- TENDANCY -->',size = 12);
plt.ylabel('Count -->',size = 12);
```

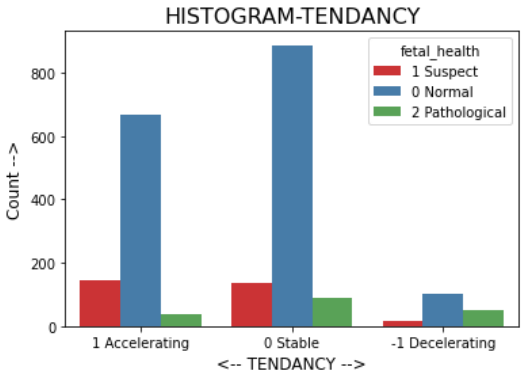


In [109]:
```
pd.DataFrame(df.hist_tendency.groupby(df['fetal_health']).value_counts())
```

Out[109]:

		hist_tendency
fetal_health	**hist_tendency**	
0 Normal	**0 Stable**	887
	1 Accelerating	667
	-1 Decelerating	101
1 Suspect	**1 Accelerating**	143
	0 Stable	137
	-1 Decelerating	15
2 Pathological	**0 Stable**	91
	-1 Decelerating	49
	1 Accelerating	36

In [110]:
```
sns.countplot(x = df.hist_tendency,hue = df.fetal_health,palette = 'Set1');
plt.title('HISTOGRAM-TENDANCY',size = 16);
plt.xlabel('<-- TENDANCY -->',size = 12);
plt.ylabel('Count -->',size = 12);
```



In [111]:
```
df.hist_tendency = df.hist_tendency.replace({'-1 Decelerating':-1.0, '0 Stable':0.0, '1 Accelerating':1.0})
```

In [112]:
```
df.fetal_health = df.fetal_health.replace({'0 Normal' : 0, '1 Suspect':1, '2 Pathological':2})
```

In [113]:
```
df.columns
```

Out[113]:
```
Index(['baseline', 'accelerations', 'uterine_contractions',
       'light_decelerations', 'abnrml_ST_variability', 'mean_ST_variability',
       'abnrml_LT_variability', 'mean_of_LT_variability', 'hist_width',
       'hist_min', 'hist_max', 'hist_no_of_peaks', 'hist_mode', 'hist_mean',
       'hist_median', 'hist_var', 'hist_tendency', 'fetal_health'],
      dtype='object')
```

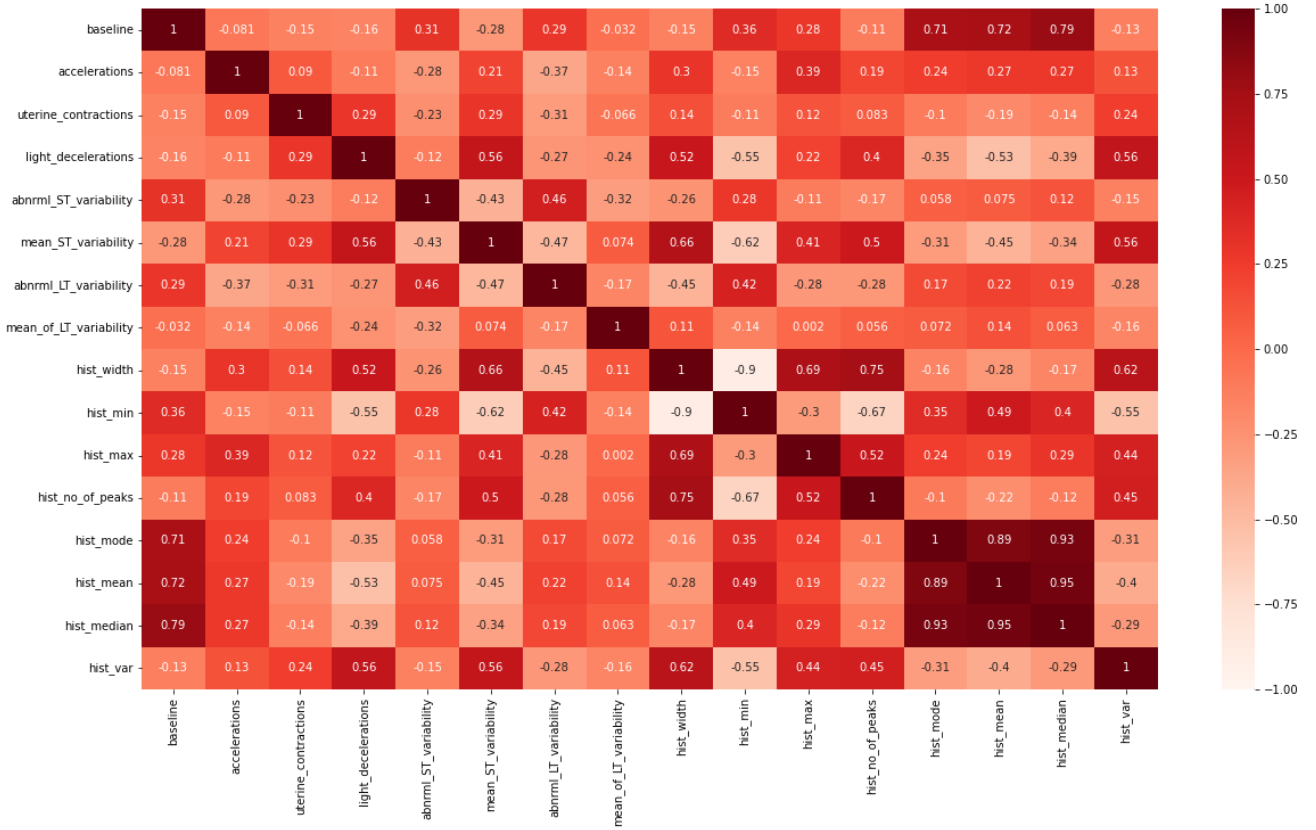
In [114]:
```
cont_features = df.iloc[:,[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]] #continuous variables
cont_features
```

Out[114]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability
0	120.0	0.000	0.000	0.000	73.0	0.5	43.0	2.
1	132.0	0.006	0.006	0.003	17.0	2.1	0.0	10.
2	133.0	0.003	0.008	0.003	16.0	2.1	0.0	13.
3	134.0	0.003	0.008	0.003	16.0	2.4	0.0	23.
4	132.0	0.007	0.008	0.000	16.0	2.4	0.0	19.
...	...	...	...	...	...	...	...	..
2121	140.0	0.000	0.007	0.000	79.0	0.2	25.0	7.
2122	140.0	0.001	0.007	0.000	78.0	0.4	22.0	7.
2123	140.0	0.001	0.007	0.000	79.0	0.4	20.0	6.
2124	140.0	0.001	0.006	0.000	78.0	0.4	27.0	7.
2125	142.0	0.002	0.008	0.000	74.0	0.4	36.0	5.

2126 rows × 16 columns

In [115]:
```
plt.figure(figsize=(20,11))
sns.heatmap(cont_features.corr(),vmin = -1,vmax = 1,annot=True,cmap = 'Reds');
```



Standardize

In [116]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [117]:
```
cont_features_sc = pd.DataFrame(sc.fit_transform(cont_features),columns = cont_features.columns)
cont_features_sc.head()
```

Out[117]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability
0	-1.352220	-0.822388	-1.482465	-0.638438	1.513190	-0.943095	1.802542	-1.028560
1	-0.132526	0.730133	0.554627	0.375243	-1.744751	0.868841	-0.535361	0.393176
2	-0.030884	-0.046128	1.233657	0.375243	-1.802928	0.868841	-0.535361	0.926327
3	0.070757	-0.046128	1.233657	0.375243	-1.802928	1.208579	-0.535361	2.632411
4	-0.132526	0.988886	1.233657	-0.638438	-1.802928	1.208579	-0.535361	2.081488

In [118]:
```
non_cont = [i for i in df.columns if i not in cont_features.columns]
df[non_cont].head()
```

Out[118]:

	hist_tendency	fetal_health
0	1.0	1
1	0.0	0
2	0.0	0
3	1.0	0
4	1.0	0

In [119]:
```
df = pd.concat([cont_features,df[non_cont]],axis = 1)
```

In [120]:
```
df.head()
```

Out[120]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability	h
0	120.0	0.000	0.000	0.000	73.0	0.5	43.0	2.4	
1	132.0	0.006	0.006	0.003	17.0	2.1	0.0	10.4	
2	133.0	0.003	0.008	0.003	16.0	2.1	0.0	13.4	
3	134.0	0.003	0.008	0.003	16.0	2.4	0.0	23.0	
4	132.0	0.007	0.008	0.000	16.0	2.4	0.0	19.9	

In [121]:
```
df.shape
```

Out[121]: (2126, 18)

X and y

In [122]:
```
X = df.drop(['fetal_health'],axis = 1)
y = df['fetal_health']
```

In [123]:
```
print(X.shape)
print(y.shape)
```

```
(2126, 17)
(2126,)
```

train and test

In [124]:
```
from sklearn.model_selection import train_test_split
```

In [125]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 808) #split for non-smote data
```

In [126]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1700, 17)
(426, 17)
(1700,)
(426,)
```

SMOTE

In [127]:
```
df.fetal_health.value_counts()
```

Out[127]:
```
0    1655
1     295
2     176
Name: fetal_health, dtype: int64
```

In [128]:
```
norm = df[df.fetal_health == 0]
susp = df[df.fetal_health == 1]
ptlg = df[df.fetal_health == 2]
```

In [129]:
```
norm.shape
```

Out[129]:
```
(1655, 18)
```

In [130]:
```
susp_oversample = resample(susp,replace = True,n_samples = len(norm),random_state = 8)
```

In [131]:
```
susp_oversample.shape
```

Out[131]:
```
(1655, 18)
```

In [132]:
```
ptlg_oversample = resample(ptlg,replace = True,n_samples = len(norm),random_state = 8)
```

In [133]:
```
ptlg_oversample.shape
```

Out[133]:
```
(1655, 18)
```

In [134]:
```
df_smote = pd.concat([norm,susp_oversample,ptlg_oversample],axis = 0)
```

In [135]:
```
df_smote = df_smote.sample(frac = 1,random_state = 66).reset_index(drop = True) #shuffle
```

In [136]:
```
df_smote.fetal_health.value_counts()
```

Out[136]:
```
1    1655
0    1655
2    1655
Name: fetal_health, dtype: int64
```

In [137]:
```
df_smote.head()
```

Out[137]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability	l
0	137.0	0.000	0.001	0.000	77.0	0.2	40.0	6.6	
1	138.0	0.004	0.004	0.006	50.0	1.9	0.0	5.6	
2	152.0	0.000	0.003	0.000	61.0	0.4	71.0	5.8	
3	145.0	0.000	0.001	0.000	50.0	0.7	17.0	7.3	
4	135.0	0.000	0.002	0.000	65.0	0.4	68.0	5.0	

X and y

In [138]:
```
Xs = df_smote.drop(['fetal_health'],axis = 1)
ys = df_smote['fetal_health']
```

In [139]:
```
print(Xs.shape)
print(ys.shape)
```

```
(4965, 17)
(4965,)
```

train and test

In [140]:
```
from sklearn.model_selection import train_test_split
```

In [141]:
```
Xtrain,Xtest,ytrain,ytest = train_test_split(Xs,ys,test_size = 0.2,random_state = 808) #split for smote data
```

In [142]:
```
print(Xtrain.shape)
print(Xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

```
(3972, 17)
(993, 17)
(3972,)
(993,)
```

feature imp

In [143]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
```

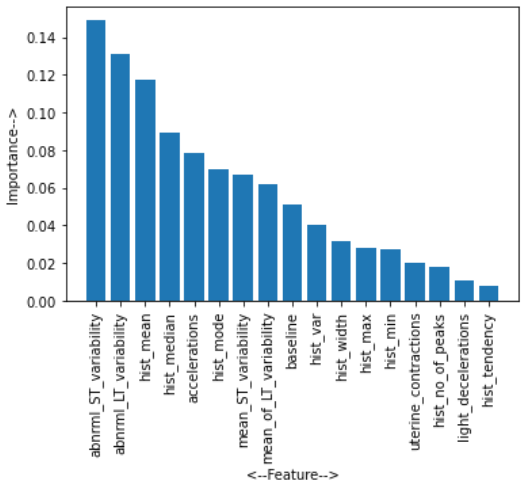
In [144]:
```
rf.fit(Xtrain,ytrain)
```

Out[144]:
```
RandomForestClassifier()
```

In [145]:
```
importances = rf.feature_importances_
```

In [146]:
```
sorted_idx = importances.argsort()[::-1] #sort in descending order
```

In [147]:
```
plt.bar(range(X_train.shape[1]), importances[sorted_idx])
plt.xticks(range(X_train.shape[1]), X.columns[sorted_idx], rotation=90)
plt.xlabel("<--Feature-->");
plt.ylabel("Importance-->");
```



In [148]:
```
pd.DataFrame({'FEATURES':X.columns[sorted_idx],'IMPORTANCE':importances[sorted_idx]})
```

Out[148]:

	FEATURES	IMPORTANCE
0	abnrml_ST_variability	0.149106
1	abnrml_LT_variability	0.131413
2	hist_mean	0.117340
3	hist_median	0.089196
4	accelerations	0.078844
5	hist_mode	0.070197
6	mean_ST_variability	0.067164
7	mean_of_LT_variability	0.061702
8	baseline	0.050820
9	hist_var	0.040229
10	hist_width	0.031394
11	hist_max	0.028283
12	hist_min	0.027586
13	uterine_contractions	0.019916
14	hist_no_of_peaks	0.018115
15	light_decelerations	0.010715
16	hist_tendency	0.007982

MODELS

1) DECISION TREE

In [149]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [150]:
```
dt = DecisionTreeClassifier(random_state = 86)
```

In [151]:
```
model_params = {'criterion' : ['gini','entropy'],
                'max_depth': [2,3,4,5,6,7,8,9,None],
               'min_samples_split' : [i for i in range (1,11)]}
```

search = GridSearchCV(dt,model_params,cv = 10) search.fit(X_train,y_train) search.best_params_

In [152]:
```
dt = DecisionTreeClassifier(criterion = 'gini',max_depth = 7 ,min_samples_split = 9,random_state = 86)
```

In [153]:
```
model = dt.fit(X_train,y_train)
```

In [154]:
```
accuracy_score(model.predict(X_train),y_train)
```

Out[154]:
```
0.9576470588235294
```

In [155]:
```
y_pred = model.predict(X_test)
```

evaluation

In [156]:
```
acc = accuracy_score(y_pred,y_test)
print('Accuracy:',acc)
```

```
Accuracy: 0.9389671361502347
```

In [157]:
```
cm = pd.DataFrame(confusion_matrix(y_test,y_pred,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

In [158]:
```
cr = classification_report(y_test,y_pred)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.95      0.98      0.96       349
           1       0.84      0.76      0.80        49
           2       0.92      0.79      0.85        28

    accuracy                           0.94       426
   macro avg       0.90      0.84      0.87       426
weighted avg       0.94      0.94      0.94       426
```

2) DECISION TREE - smote

In [159]:
```
dt = DecisionTreeClassifier(random_state = 86)
```

In [160]:
```
model_params = {'criterion' : ['gini','entropy'],
                'max_depth': [2,3,4,5,6,7,8,9,None],
               'min_samples_split' : [i for i in range (1,11)]}
```

search = GridSearchCV(dt,model_params,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [161]:
```
dt = DecisionTreeClassifier(criterion = 'entropy',max_depth = None,min_samples_split = 3,random_state = 86)
```

In [162]:
```
model2 = dt.fit(Xtrain,ytrain)
```

In [163]:
```
accuracy_score(model2.predict(Xtrain),ytrain)
```

Out[163]: 0.999496475327291

In [164]:
```
y_pred2 = model2.predict(Xtest)
```

evaluation

In [165]:
```
acc2 = accuracy_score(y_pred2,ytest)
print('Accuracy:',acc2)
```

```
Accuracy: 0.9758308157099698
```

In [166]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred2,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[166]:

	0	1	2
0	319	22	2
1	0	313	0
2	0	0	337

In [167]:
```
print(classification_report(ytest,y_pred2))
```

```
              precision    recall  f1-score   support

           0       1.00      0.93      0.96       343
           1       0.93      1.00      0.97       313
           2       0.99      1.00      1.00       337

    accuracy                           0.98       993
   macro avg       0.98      0.98      0.98       993
weighted avg       0.98      0.98      0.98       993
```

3) RANDOM FOREST

In [168]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [169]:
```
rf = RandomForestClassifier(random_state = 86)
```

In [170]:
```
params = {'criterion' : ['gini','entropy'],
          'n_estimators':[20,40,60,80,100,120],
          'max_features':['sqrt',0.25,0.50,0.75,1],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split':[i for i in range(1,11,1)]}
```

search = GridSearchCV(rf,params,n_jobs = -1,cv = 10) search.fit(X_train,y_train) search.best_params_

In [171]:
```
rf = RandomForestClassifier(criterion = 'entropy',n_estimators = 100 ,max_features = 0.5 ,max_depth = None,min_samples_split =
```

In [172]:
```
model3 = rf.fit(X_train,y_train)
```

In [173]:
```
accuracy_score(model3.predict(X_train),y_train)
```

Out[173]:
```
0.9994117647058823
```

In [174]:
```
y_pred3 = model3.predict(X_test)
```

evaluation

In [175]:
```
acc3 = accuracy_score(y_pred3,y_test)
print('Accuracy:',acc3)
```

```
Accuracy: 0.9624413145539906
```

In [176]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred3,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[176]:

	0	1	2
0	347	2	0
1	7	41	1
2	5	1	22

In [177]:
```
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       349
           1       0.93      0.84      0.88        49
           2       0.96      0.79      0.86        28

    accuracy                           0.96       426
   macro avg       0.95      0.87      0.91       426
weighted avg       0.96      0.96      0.96       426
```

4) RANDOM FOREST - smote

In [178]:
```
rf = RandomForestClassifier(random_state = 86)
```

In [179]:
```
params = {'criterion' : ['gini','entropy'],
          'n_estimators':[20,40,60,80,100,120],
          'max_features':['sqrt',0.25,0.50,0.75,1],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split':[i for i in range(1,11,1)]}
```

search = GridSearchCV(rf,params,n_jobs = -1,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [180]:
```
rf = RandomForestClassifier(criterion = 'gini',n_estimators = 60,max_features = 1,max_depth = None,min_samples_split = 2,rando
```

In [181]:
```
model4 = rf.fit(Xtrain,ytrain)
```

In [182]:
```
accuracy_score(model4.predict(Xtrain),ytrain)
```

Out[182]:
```
0.999496475327291
```

In [183]:
```
y_pred4 = model4.predict(Xtest)
```

evaluation

In [184]:
```
acc4 = accuracy_score(y_pred4,ytest)
print('Accuracy :',acc4)
```

```
Accuracy : 0.9929506545820745
```

In [185]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred4,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[185]:

	0	1	2
0	336	7	0
1	0	313	0
2	0	0	337

In [186]:
```
print(classification_report(ytest,y_pred4))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       343
           1       0.98      1.00      0.99       313
           2       1.00      1.00      1.00       337

    accuracy                           0.99       993
   macro avg       0.99      0.99      0.99       993
weighted avg       0.99      0.99      0.99       993
```

HANDLING OUTLIERS

In [187]:
```
df = data.copy()
```

In [188]:
```
df.drop(['fetal_movement','severe_decelerations','prolongued_decelerations','hist_no_of_zeroes'],axis = 1,inplace = True) #dro
```

In [189]:
```
df.head()
```

Out[189]:

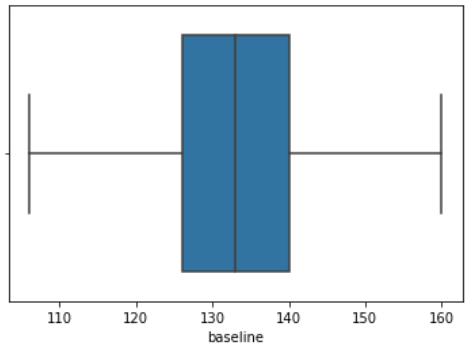
	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability	h
0	120.0	0.000	0.000	0.000	73.0	0.5	43.0	2.4	
1	132.0	0.006	0.006	0.003	17.0	2.1	0.0	10.4	
2	133.0	0.003	0.008	0.003	16.0	2.1	0.0	13.4	
3	134.0	0.003	0.008	0.003	16.0	2.4	0.0	23.0	
4	132.0	0.007	0.008	0.000	16.0	2.4	0.0	19.9	

In [190]:
```
df.columns
```

Out[190]:
```
Index(['baseline', 'accelerations', 'uterine_contractions',
       'light_decelerations', 'abnrml_ST_variability', 'mean_ST_variability',
       'abnrml_LT_variability', 'mean_of_LT_variability', 'hist_width',
       'hist_min', 'hist_max', 'hist_no_of_peaks', 'hist_mode', 'hist_mean',
       'hist_median', 'hist_var', 'hist_tendency', 'fetal_health'],
      dtype='object')
```

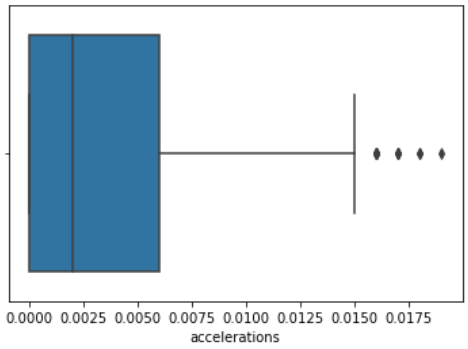
1) baseline

In [191]:
```
sns.boxplot(x = df.baseline);
```



2) accelerations

In [192]:
```
sns.boxplot(x= df.accelerations);
```



In [193]:
```
q1 = df.accelerations.quantile(0.25)
q3 = df.accelerations.quantile(0.75)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1  - 1.5 * IQR
```

In [194]:
```
len(df[df.accelerations > ub]) # total outliers values
```

Out[194]: 14

In [195]:
```
df[df.accelerations > ub]
```

Out[195]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variabilit
181	138.0	0.017	0.005	0.000	35.0	5.3	0.0	4.5
497	130.0	0.016	0.002	0.000	34.0	2.1	0.0	3.(
529	142.0	0.019	0.000	0.000	32.0	2.3	0.0	0.(
530	142.0	0.016	0.000	0.002	32.0	3.1	0.0	1.(
531	142.0	0.016	0.004	0.000	38.0	1.3	0.0	0.(
552	136.0	0.016	0.004	0.000	35.0	4.9	0.0	5.2
630	134.0	0.017	0.004	0.000	48.0	2.2	0.0	0.(
1093	122.0	0.016	0.001	0.000	22.0	2.2	0.0	1.5
1094	122.0	0.018	0.002	0.000	22.0	2.5	0.0	2.2
1096	123.0	0.017	0.002	0.000	24.0	2.2	0.0	1.1
1248	112.0	0.018	0.007	0.000	25.0	1.4	0.0	0.(
1858	138.0	0.016	0.005	0.000	51.0	0.9	0.0	0.
1859	138.0	0.017	0.004	0.000	51.0	0.9	0.0	0.2
1862	138.0	0.016	0.000	0.000	51.0	1.0	0.0	2.4

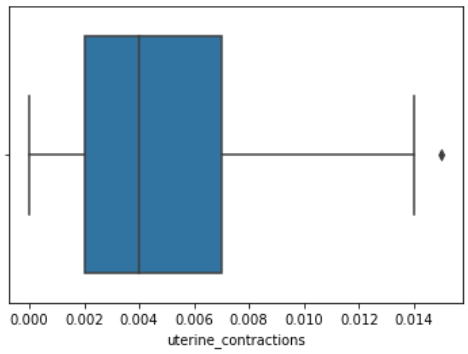
In [196]:
```
df.accelerations = np.where(df.accelerations > ub,ub,df.accelerations)
```

In [197]:
```
len(df[df.accelerations > ub])
```

Out[197]: 0

3) uterine_contractions

In [198]:
```
sns.boxplot(x = df.uterine_contractions);
```



In [199]:
```
q1 = df.uterine_contractions.quantile(0.25)
q3 = df.uterine_contractions.quantile(0.75)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [200]:
```
df[df.uterine_contractions > ub]
```

Out[200]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability
1164	131.0	0.011	0.015	0.0	26.0	1.5	0.0	3.(

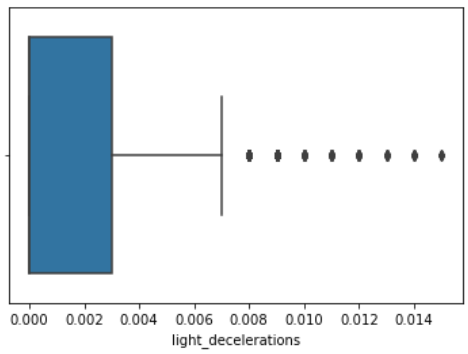
In [201]:
```
df.drop(df[df.uterine_contractions > ub].index,inplace = True)
```

In [202]:
```
len(df[df.uterine_contractions > ub])
```

Out[202]: 0

4) light_decelerations

In [203]:
```
sns.boxplot(x = df.light_decelerations);
```



In [204]:
```
q1 = df.light_decelerations.quantile(0.25)
q3 = df.light_decelerations.quantile(0.75)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
```

In [205]:
```
len(df[df.light_decelerations > ub])
```

Out[205]: 150

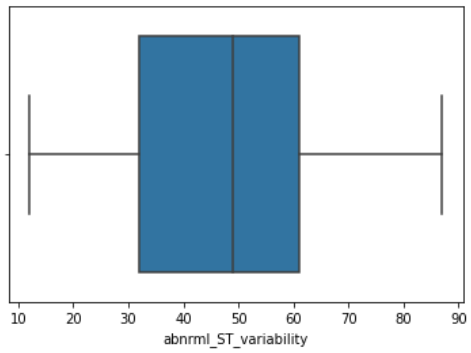
In [206]:
```
df.light_decelerations = np.where(df.light_decelerations > ub,ub,df.light_decelerations)
```

In [207]:
```
len(df[df.light_decelerations > ub])
```

Out[207]: 0

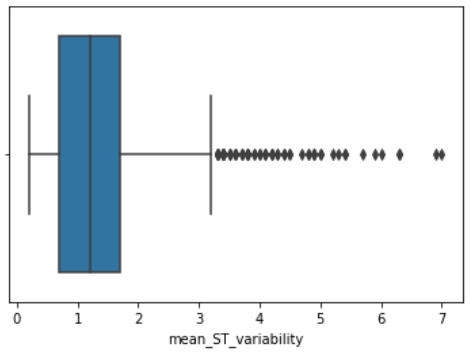
5) abnrml_ST_variability

In [208]:
```
sns.boxplot(x = df.abnrml_ST_variability);
```



6) mean_ST_variability

In [209]:
```
sns.boxplot(x = df.mean_ST_variability);
```



In [210]:
```
q1 = df.mean_ST_variability.quantile(0.25)
q3 = df.mean_ST_variability.quantile(0.75)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [211]:
```
len(df[df.mean_ST_variability > ub]) # no of ouliers
```

Out[211]: 70

In [212]:
```
df[df.mean_ST_variability > ub] #outliers
```

Out[212]:

	baseline	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variab
5	134.0	0.001	0.010	0.0075	26.0	5.9	0.0	
6	134.0	0.001	0.013	0.0075	29.0	6.3	0.0	
28	132.0	0.000	0.001	0.0075	29.0	4.4	0.0	
29	132.0	0.000	0.000	0.0075	26.0	6.0	0.0	
30	132.0	0.000	0.002	0.0075	26.0	4.5	0.0	
...	...	...	...	...	...	...	...	...
2024	129.0	0.000	0.005	0.0075	58.0	3.9	0.0	
2034	129.0	0.000	0.006	0.0050	67.0	3.3	0.0	
2048	128.0	0.000	0.008	0.0075	63.0	4.2	0.0	
2049	125.0	0.005	0.007	0.0000	66.0	4.1	5.0	
2051	127.0	0.003	0.005	0.0000	66.0	4.2	5.0	

In [213]:
```
df[df.mean_ST_variability > ub].fetal_health.value_counts().sort_index()
```

Out[213]:
```
0    60
1     3
2     7
Name: fetal_health, dtype: int64
```

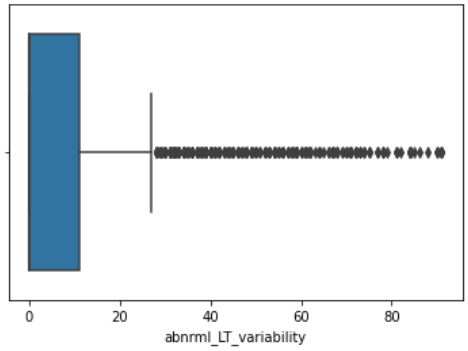
In [214]:
```
df.mean_ST_variability = np.where(df.mean_ST_variability > ub,ub,df.mean_ST_variability)
```

In [215]:
```
len(df[df.mean_ST_variability > ub])
```

Out[215]:
```
0
```

7) abnrml_LT_variability

In [216]:
```
sns.boxplot(x = df.abnrml_LT_variability);
```



In [217]:
```
q3 = df.abnrml_LT_variability.quantile(0.75)
q1 = df.abnrml_LT_variability.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
```

In [218]:
```
len(df[df.abnrml_LT_variability > ub])# so many outliers;lets drop some and adjust rest
```

Out[218]:
```
309
```

In [219]:
```
ub
```

Out[219]:
```
27.5
```

In [220]:
```
df.drop(df[df.abnrml_LT_variability > 2*ub].index,inplace = True)
```

In [221]:
```
len(df[df.abnrml_LT_variability > ub])
```

Out[221]:
```
200
```

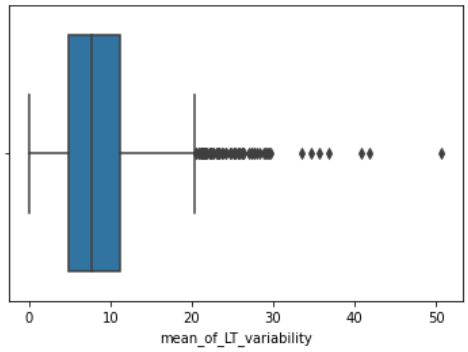
In [222]:
```
df.abnrml_LT_variability = np.where(df.abnrml_LT_variability > ub,ub,df.abnrml_LT_variability)
```

In [223]:
```
len(df[df.abnrml_LT_variability > ub])
```

Out[223]:
```
0
```

8) mean_of_LT_variability

In [224]:
```
sns.boxplot(x = df.mean_of_LT_variability);
```



In [225]:
```
q3 = df.mean_of_LT_variability.quantile(0.75)
q1 = df.mean_of_LT_variability.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
```

In [226]:
```
len(df[df.mean_of_LT_variability > ub])
```

Out[226]: 70

In [227]:
```
ub
```

Out[227]: 20.4

In [228]:
```
df.drop(df[df.mean_of_LT_variability > 1.5 * ub].index,inplace = True)
```

In [229]:
```
len(df[df.mean_of_LT_variability > ub])
```

Out[229]: 63

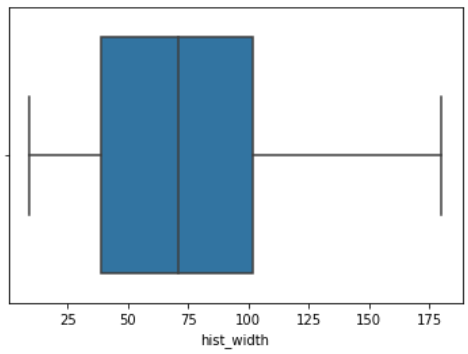
In [230]:
```
df.mean_of_LT_variability = np.where(df.mean_of_LT_variability > ub,ub,df.mean_of_LT_variability)
```

In [231]:
```
len(df[df.mean_of_LT_variability > ub])
```

Out[231]: 0

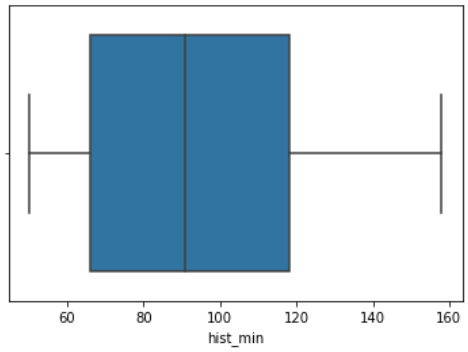
9) hist_width

In [232]:
```
sns.boxplot(x = df.hist_width);
```



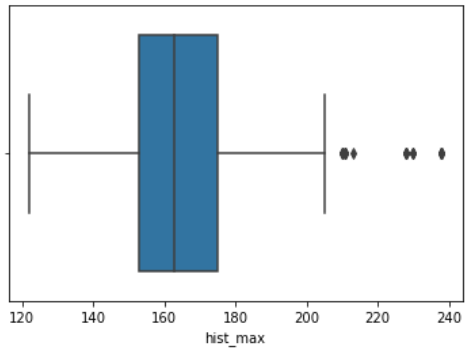
10) hist_min

In [233]:
```
sns.boxplot(x = df.hist_min);
```



11) hist_max

In [234]:
```
sns.boxplot(x = df.hist_max);
```



In [235]:
```
q3 = df.hist_max.quantile(0.75)
q1 = df.hist_max.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
```

In [236]:
```
ub
```

Out[236]:
```
208.0
```

In [237]:
```
len(df[df.hist_max > ub])
```

Out[237]:
```
24
```

In [238]:
```
df[df.hist_max > ub].fetal_health.value_counts()
```

Out[238]:
```
0    18
2     6
Name: fetal_health, dtype: int64
```

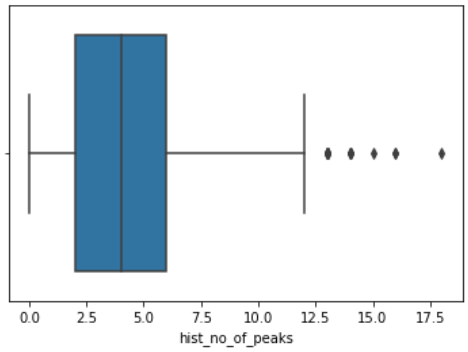
In [239]:
```
df.hist_max = np.where(df.hist_max > ub,ub,df.hist_max)
```

In [240]:
```
len(df[df.hist_max > ub])
```

Out[240]:
```
0
```

12) hist_no_of_peaks

In [241]:
```
sns.boxplot(x = df.hist_no_of_peaks);
```



In [242]:
```
q3 = df.hist_no_of_peaks.quantile(0.75)
q1 = df.hist_no_of_peaks.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
```

In [243]:
```
len(df[df.hist_no_of_peaks > ub])
```

Out[243]: 19

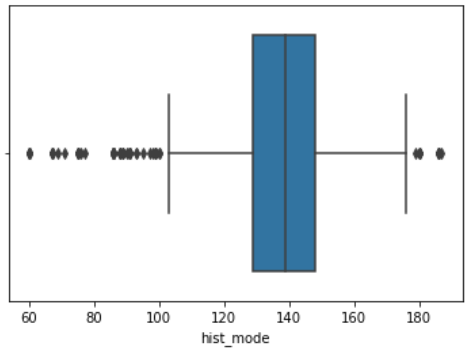
In [244]:
```
df.hist_no_of_peaks = np.where(df.hist_no_of_peaks > ub,ub,df.hist_no_of_peaks)
```

In [245]:
```
len(df[df.hist_no_of_peaks > ub])
```

Out[245]: 0

13) hist_mode

In [246]:
```
sns.boxplot(x = df.hist_mode);
```



In [247]:
```
q3 = df.hist_mode.quantile(0.75)
q1 = df.hist_mode.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [248]:
```
len(df[df.hist_mode < lb]) + len(df[df.hist_mode > ub])
```

Out[248]: 73

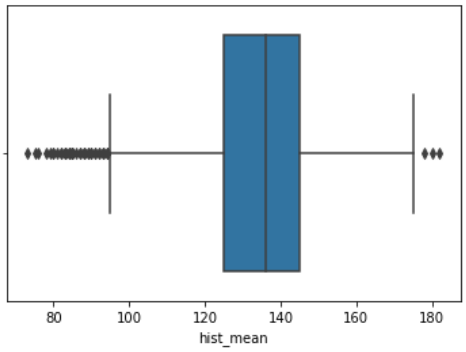
In [249]:
```
df.hist_mode = np.where(df.hist_mode > ub,ub,np.where(df.hist_mode < lb,lb,df.hist_mode))
```

In [250]:
```
len(df[df.hist_mode < lb]) + len(df[df.hist_mode > ub])
```

Out[250]: 0

14) hist_mean

In [251]:
```
sns.boxplot(x = df.hist_mean);
```



In [252]:
```
q3 = df.hist_mean.quantile(0.75)
q1 = df.hist_mean.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [253]:
```
len(df[df.hist_mean < lb]) + len(df[df.hist_mean > ub])
```

Out[253]: 45

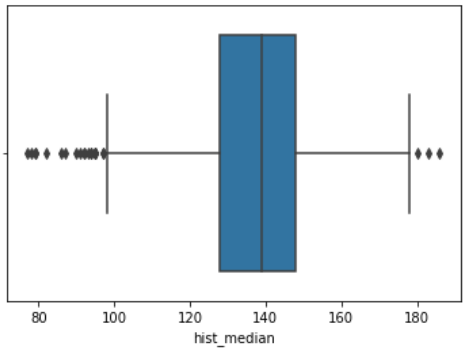
In [254]:
```
df.hist_mean = np.where(df.hist_mean > ub,ub,np.where(df.hist_mean < lb,lb,df.hist_mean))
```

In [255]:
```
len(df[df.hist_mean < lb]) + len(df[df.hist_mean > ub])
```

Out[255]: 0

15) hist_median

In [256]:
```
sns.boxplot(x = df.hist_median);
```



In [257]:
```
q3 = df.hist_median.quantile(0.75)
q1 = df.hist_median.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [258]:
```
len(df[df.hist_median < lb]) + len(df[df.hist_median > ub])
```

Out[258]: 23

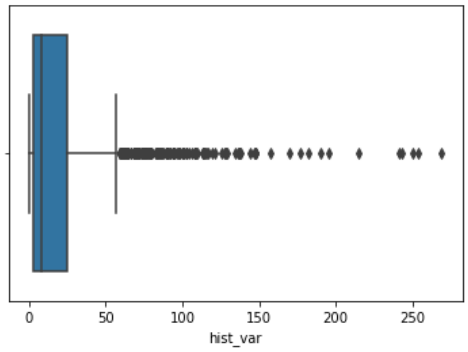
In [259]:
```
df.hist_median = np.where(df.hist_median > ub,ub,np.where(df.hist_median < lb,lb,df.hist_median))
```

In [260]:
```
len(df[df.hist_median < lb]) + len(df[df.hist_median > ub])
```

Out[260]: 0

16) hist_var

In [261]:
```
sns.boxplot(x = df.hist_var);
```



In [262]:
```
df.columns
```

Out[262]:
```
Index(['baseline', 'accelerations', 'uterine_contractions',
       'light_decelerations', 'abnrml_ST_variability', 'mean_ST_variability',
       'abnrml_LT_variability', 'mean_of_LT_variability', 'hist_width',
       'hist_min', 'hist_max', 'hist_no_of_peaks', 'hist_mode', 'hist_mean',
       'hist_median', 'hist_var', 'hist_tendency', 'fetal_health'],
      dtype='object')
```

In [263]:
```
q3 = df.hist_var.quantile(0.75)
q1 = df.hist_var.quantile(0.25)
IQR = q3 - q1
ub = q3 + 1.5 * IQR
lb = q1 - 1.5 * IQR
```

In [264]:
```
len(df[df.hist_var > ub])
```

Out[264]:
```
184
```

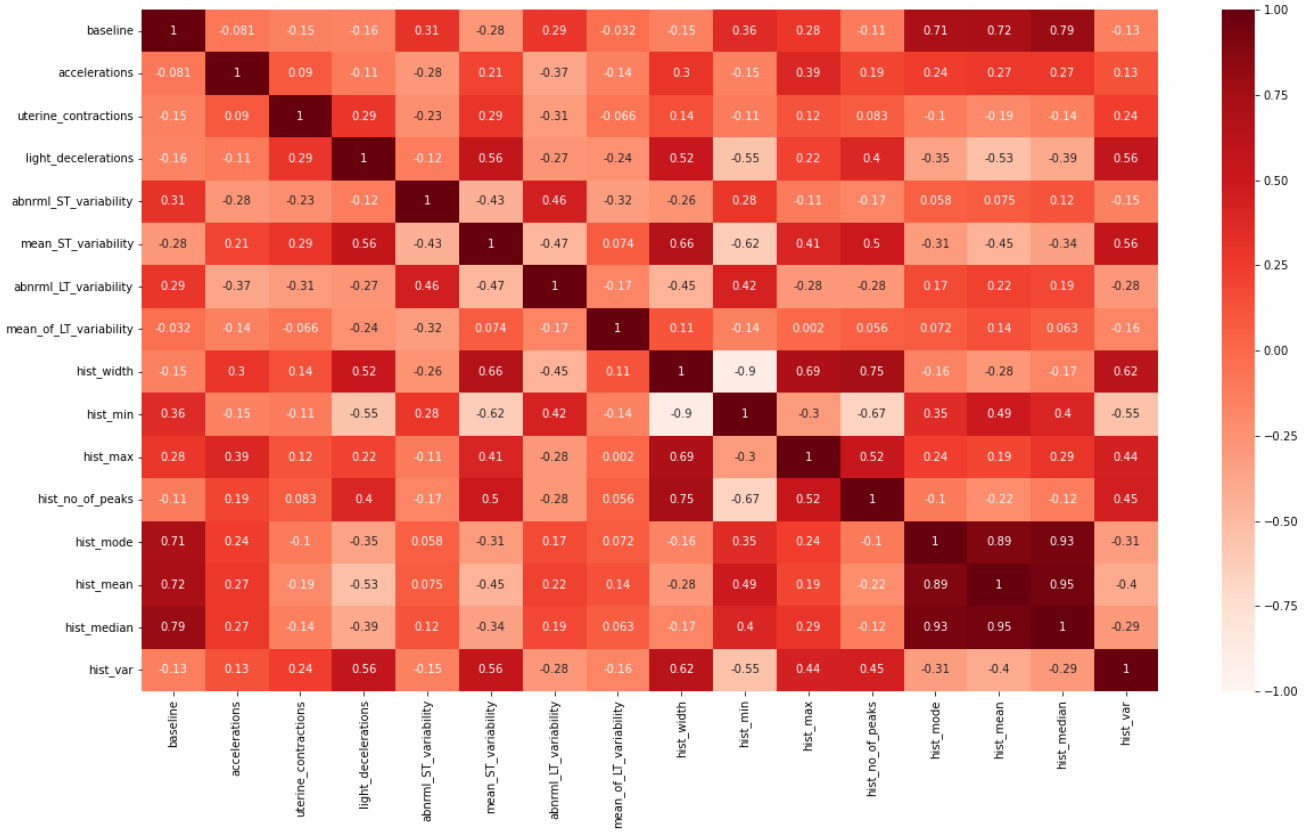
In [265]:
```
df.hist_var = np.where(df.hist_var > ub,ub,df.hist_var)
```

In [266]:
```
len(df[df.hist_var > ub])
```

Out[266]:
```
0
```

STATISTICAL TESTS

In [267]:
```
plt.figure(figsize=(20,11));
sns.heatmap(cont_features.corr(),vmin = -1,vmax = 1,annot=True,cmap = 'Reds');
```



In [268]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [269]:
```
vif= cont_features
vif=pd.DataFrame()
vif['FEATURE']=cont_features.columns

vif['VIF']=[variance_inflation_factor(cont_features.values,i)for i in range(len(cont_features.columns))]
print(vif)
```

```
                FEATURE          VIF
0              baseline   697.340142
1         accelerations     4.717564
2  uterine_contractions     3.962631
3   light_decelerations     3.833837
4  abnrml_ST_variability    15.741282
5    mean_ST_variability     9.382637
6  abnrml_LT_variability     2.358030
7  mean_of_LT_variability     5.980798
8           hist_width          inf
9            hist_min          inf
10           hist_max          inf
11     hist_no_of_peaks     6.734765
12          hist_mode   619.777903
13          hist_mean  1439.921280
14        hist_median  2083.003702
15           hist_var     3.087393
```

In [270]:
```
df.drop(['baseline','hist_width','hist_min','hist_max','hist_mode','hist_mean','hist_median'],axis = 1,inplace = True)
```

In [271]:
```
df.columns
```

Out[271]:
```
Index(['accelerations', 'uterine_contractions', 'light_decelerations',
       'abnrml_ST_variability', 'mean_ST_variability', 'abnrml_LT_variability',
       'mean_of_LT_variability', 'hist_no_of_peaks', 'hist_var',
       'hist_tendency', 'fetal_health'],
      dtype='object')
```

In [272]:
```
cont_features = df.iloc[:,[0,1,2,3,4,5,6,7,8]] #continuous variables
cont_features
```

Out[272]:

	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability	hist_no
0	0.000	0.000	0.000	73.0	0.5	27.5	2.4	
1	0.006	0.006	0.003	17.0	2.1	0.0	10.4	
2	0.003	0.008	0.003	16.0	2.1	0.0	13.4	
3	0.003	0.008	0.003	16.0	2.4	0.0	20.4	
4	0.007	0.008	0.000	16.0	2.4	0.0	19.9	
...	...	...	...	...	...	...	...	...
2121	0.000	0.007	0.000	79.0	0.2	25.0	7.2	
2122	0.001	0.007	0.000	78.0	0.4	22.0	7.1	
2123	0.001	0.007	0.000	79.0	0.4	20.0	6.1	
2124	0.001	0.006	0.000	78.0	0.4	27.0	7.0	
2125	0.002	0.008	0.000	74.0	0.4	27.5	5.0	

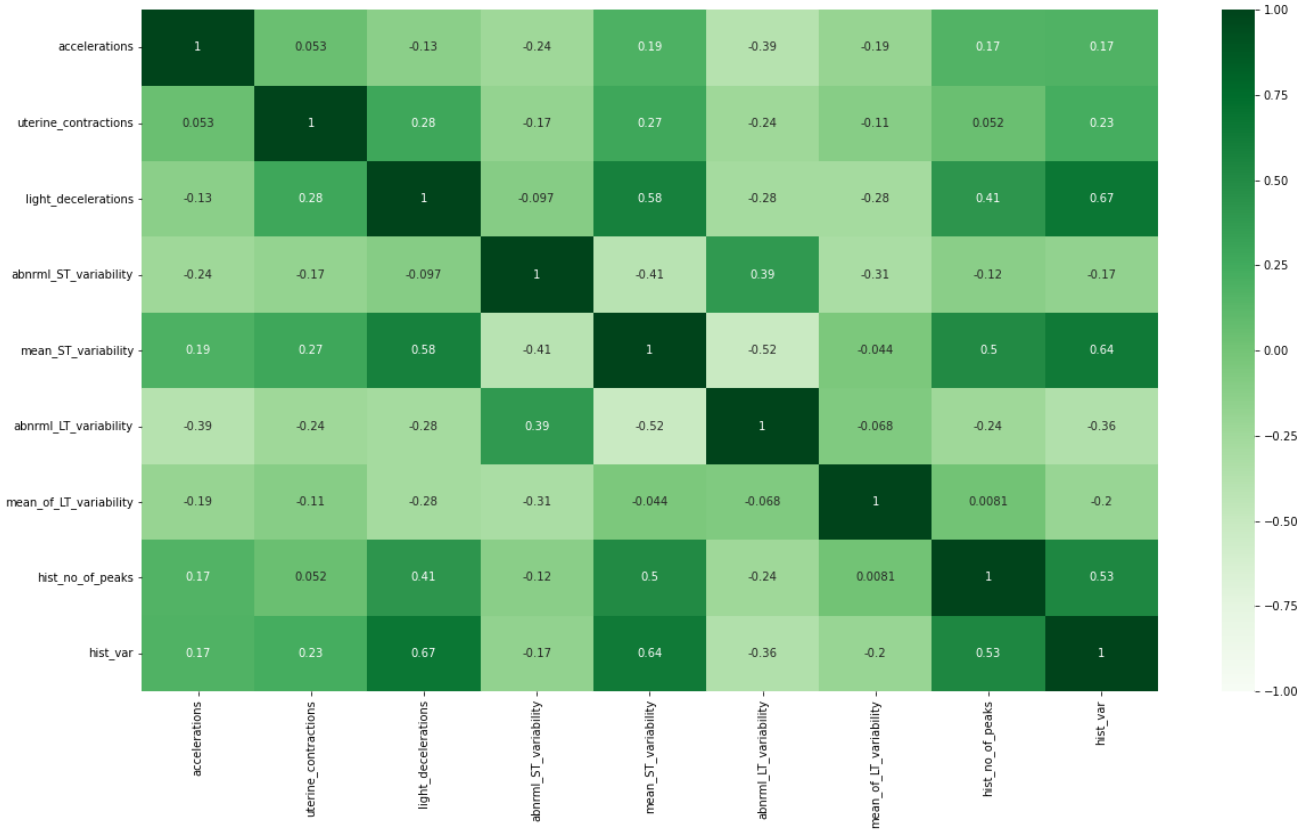
2009 rows × 9 columns

In [273]:
```
vif= cont_features
vif=pd.DataFrame()
vif['FEATURE']=cont_features.columns

vif['VIF']=[variance_inflation_factor(cont_features.values,i)for i in range(len(cont_features.columns))]
print(vif)
```

```
                 FEATURE       VIF
0          accelerations  2.376174
1    uterine_contractions  3.556454
2     light_decelerations  4.001274
3   abnrml_ST_variability  5.258422
4     mean_ST_variability  8.711974
5   abnrml_LT_variability  2.034155
6  mean_of_LT_variability  3.368643
7       hist_no_of_peaks  4.958460
8              hist_var  4.404451
```

In [274]:
```
plt.figure(figsize=(20,11));
sns.heatmap(cont_features.corr(),vmin = -1,vmax = 1,annot=True,cmap = 'Greens');
```



In [275]:
```
df.head()
```

Out[275]:

	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	mean_ST_variability	abnrml_LT_variability	mean_of_LT_variability	hist_no_of_
0	0.000	0.000	0.000	73.0	0.5	27.5	2.4	
1	0.006	0.006	0.003	17.0	2.1	0.0	10.4	
2	0.003	0.008	0.003	16.0	2.1	0.0	13.4	
3	0.003	0.008	0.003	16.0	2.4	0.0	20.4	
4	0.007	0.008	0.000	16.0	2.4	0.0	19.9	

In [276]:
```
df.columns
```

Out[276]:
```
Index(['accelerations', 'uterine_contractions', 'light_decelerations',
       'abnrml_ST_variability', 'mean_ST_variability', 'abnrml_LT_variability',
       'mean_of_LT_variability', 'hist_no_of_peaks', 'hist_var',
       'hist_tendency', 'fetal_health'],
      dtype='object')
```

ANOVA

In [277]:
```
from statsmodels.formula.api import ols
import statsmodels.api as sm
```

In [278]:
```
anv_mdl = ols('accelerations ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[278]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	0.003459	0.003459	262.946592	1.152514e-55
Residual	2007.0	0.026405	0.000013	NaN	NaN

In [279]:
```
anv_mdl = ols('uterine_contractions ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[279]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	0.000238	0.000238	28.717412	9.336984e-08
Residual	2007.0	0.016666	0.000008	NaN	NaN

In [280]:
```
anv_mdl = ols('light_decelerations ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[280]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	0.000126	0.000126	19.424908	0.000011
Residual	2007.0	0.013061	0.000007	NaN	NaN

In [281]:
```
anv_mdl = ols('abnrml_ST_variability ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[281]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	97503.296074	97503.296074	420.666533	5.003892e-85
Residual	2007.0	465188.219108	231.782870	NaN	NaN

In [282]:
```
anv_mdl = ols('mean_ST_variability ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl #drop
```

Out[282]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	0.231143	0.231143	0.404815	0.524686
Residual	2007.0	1145.964955	0.570984	NaN	NaN

In [283]:
```
anv_mdl = ols('abnrml_LT_variability ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[283]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	9879.512232	9879.512232	120.279658	3.200431e-27
Residual	2007.0	164850.660989	82.137848	NaN	NaN

In [284]:
```
anv_mdl = ols('mean_of_LT_variability ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[284]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	2152.636134	2152.636134	87.90774	1.785206e-20
Residual	2007.0	49146.306155	24.487447	NaN	NaN

In [285]:
```
anv_mdl = ols('hist_no_of_peaks ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[285]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	36.424821	36.424821	4.341472	0.037321
Residual	2007.0	16838.672242	8.389971	NaN	NaN

In [286]:
```
anv_mdl = ols('hist_var ~ fetal_health',data = df).fit()
anv_tbl = sm.stats.anova_lm(anv_mdl)
anv_tbl
```

Out[286]:

	df	sum_sq	mean_sq	F	PR(>F)
fetal_health	1.0	22946.369076	22946.369076	69.057898	1.737977e-16
Residual	2007.0	666880.456211	332.277258	NaN	NaN

CHI 2

In [287]:
```
from scipy.stats import chi2_contingency
```

In [288]:
```
chi2_contingency(pd.crosstab(df.hist_tendency,df.fetal_health))[1]
```

Out[288]: 1.462731719810631e-38

In [289]:
```
df.drop(['mean_ST_variability'],axis =1 ,inplace = True)
```

In [290]:
```
df.columns
```

Out[290]:
```
Index(['accelerations', 'uterine_contractions', 'light_decelerations',
       'abnrml_ST_variability', 'abnrml_LT_variability',
       'mean_of_LT_variability', 'hist_no_of_peaks', 'hist_var',
       'hist_tendency', 'fetal_health'],
      dtype='object')
```

In [291]:
```
df.head()
```

Out[291]:

	accelerations	uterine_contractions	light_decelerations	abnrml_ST_variability	abnrml_LT_variability	mean_of_LT_variability	hist_no_of_peaks	hist_var	hist
0	0.000	0.000	0.000	73.0	27.5	2.4	2.0	58.0	
1	0.006	0.006	0.003	17.0	0.0	10.4	6.0	12.0	
2	0.003	0.008	0.003	16.0	0.0	13.4	5.0	13.0	
3	0.003	0.008	0.003	16.0	0.0	20.4	11.0	13.0	
4	0.007	0.008	0.000	16.0	0.0	19.9	9.0	11.0	

In [292]:
```
df.shape
```

Out[292]: (2009, 10)

MODELS

X and y

In [293]:
```
X = df.drop(['fetal_health'],axis = 1)
y = df['fetal_health']
```

In [294]:
```
print(X.shape)
print(y.shape)
```

```
(2009, 9)
(2009,)
```

Standardize

In [295]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [296]:
```
X = sc.fit_transform(X)
```

train and test

In [297]:
```
from sklearn.model_selection import train_test_split
```

In [298]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 808) #split for non-smote data
```

In [299]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1607, 9)
(402, 9)
(1607,)
(402,)
```

SMOTE

In [300]:
```
df.fetal_health.value_counts()
```

Out[300]:
```
0    1627
1     251
2     131
Name: fetal_health, dtype: int64
```

In [301]:
```
norm = df[df.fetal_health == 0]
susp = df[df.fetal_health == 1]
ptlg = df[df.fetal_health == 2]
```

In [302]:
```
norm.shape
```

Out[302]:
```
(1627, 10)
```

In [303]:
```
susp_oversample = resample(susp,replace = True,n_samples = len(norm),random_state = 8)
```

In [304]:
```
susp_oversample.shape
```

Out[304]:
```
(1627, 10)
```

In [305]:
```
ptlg_oversample = resample(ptlg,replace = True,n_samples = len(norm),random_state = 8)
```

In [306]:
```
ptlg_oversample.shape
```

Out[306]:
```
(1627, 10)
```

In [307]:
```
df_smote = pd.concat([norm,susp_oversample,ptlg_oversample],axis = 0)
```

In [308]:
```
df_smote = df_smote.sample(frac = 1,random_state = 66).reset_index(drop = True) #shuffle
```

In [309]:
```
df_smote.fetal_health.value_counts()
```

Out[309]:
```
2    1627
0    1627
1    1627
Name: fetal_health, dtype: int64
```

X and y

In [310]:
```
Xs = df_smote.drop(['fetal_health'],axis = 1)
ys = df_smote['fetal_health']
```

In [311]:
```
print(Xs.shape)
print(ys.shape)
```

```
(4881, 9)
(4881,)
```

Standardize

In [312]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

In [313]:
```
Xs = sc.fit_transform(Xs)
```

train and test

In [314]:
```
from sklearn.model_selection import train_test_split
```

In [315]:
```
Xtrain,Xtest,ytrain,ytest = train_test_split(Xs,ys,test_size = 0.2,random_state = 808) #split for smote data
```

In [316]:
```
print(Xtrain.shape)
print(Xtest.shape)
print(ytrain.shape)
print(ytest.shape)
```

```
(3904, 9)
(977, 9)
(3904,)
(977,)
```

5) LOGISTIC REGRESSION

In [317]:
```
from sklearn.linear_model import LogisticRegression
```

STATISTICAL METHODS

one vs rest

In [318]:
```
ovr = LogisticRegression(multi_class = 'ovr',random_state = 86)
```

In [319]:
```
model5 = ovr.fit(X_train,y_train)
```

In [320]:
```
accuracy_score(model5.predict(X_train),y_train)
```

Out[320]:
```
0.8904791537025514
```

In [321]:
```
y_pred5 = model5.predict(X_test)
```

evaluation

In [322]:
```
acc5 = accuracy_score(y_pred5,y_test)
print('Accuracy:',acc5)
```

```
Accuracy: 0.8582089552238806
```

In [323]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred5,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[323]:

	0	1	2
0	304	9	7
1	30	29	1
2	7	3	12

In [324]:
```
print(classification_report(y_test,y_pred5))
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       320
           1       0.71      0.48      0.57        60
           2       0.60      0.55      0.57        22

    accuracy                           0.86       402
   macro avg       0.73      0.66      0.69       402
weighted avg       0.85      0.86      0.85       402
```

one vs rest - smote

In [325]:
```
ovr = LogisticRegression(multi_class = 'ovr',random_state = 86)
```

In [326]:
```
model6 = ovr.fit(Xtrain,ytrain)
```

In [327]:
```
accuracy_score(model6.predict(Xtrain),ytrain)
```

Out[327]:
```
0.8242827868852459
```

In [328]:
```
y_pred6 = model6.predict(Xtest)
```

evaluation

In [329]:
```
acc6 = accuracy_score(y_pred6,ytest)
print('Accuracy:',acc6)
```

```
Accuracy: 0.8331627430910952
```

In [330]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred6,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[330]:

	0	1	2
0	263	28	30
1	33	253	36
2	2	34	298

In [331]:
```
print(classification_report(ytest,y_pred6))
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       321
           1       0.80      0.79      0.79       322
           2       0.82      0.89      0.85       334

    accuracy                           0.83       977
   macro avg       0.83      0.83      0.83       977
weighted avg       0.83      0.83      0.83       977
```

mutinomial

In [332]:
```
multinomial = LogisticRegression(multi_class = 'multinomial',random_state = 86)
```

In [333]:
```
model7 = multinomial.fit(X_train,y_train)
```

In [334]:
```
accuracy_score(model7.predict(X_train),y_train)
```

Out[334]:
```
0.8892345986309894
```

In [335]:
```
y_pred7 = model7.predict(X_test)
```

evaluation

In [336]:
```
acc7 = accuracy_score(y_pred7,y_test)
print('Accuracy:',acc7)
```

```
Accuracy: 0.8631840796019901
```

In [337]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred7,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[337]:

	0	1	2
0	304	10	6
1	30	29	1
2	5	3	14

In [338]:
```
print(classification_report(y_test,y_pred7))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92       320
           1       0.69      0.48      0.57        60
           2       0.67      0.64      0.65        22

    accuracy                           0.86       402
   macro avg       0.75      0.69      0.71       402
weighted avg       0.85      0.86      0.85       402
```

mutinomial - smote

In [339]:
```
multinomial = LogisticRegression(multi_class = 'multinomial',random_state = 86)
```

In [340]:
```
model8 = multinomial.fit(Xtrain,ytrain)
```

In [341]:
```
accuracy_score(model8.predict(Xtrain),ytrain)
```

Out[341]: 0.8271004098360656

In [342]:
```
y_pred8 = model8.predict(Xtest)
```

evaluation

In [343]:
```
acc8 = accuracy_score(y_pred8,ytest)
print('Accuracy:',acc8)
```

```
Accuracy: 0.8280450358239508
```

In [344]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred8,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[344]:

	0	1	2
0	260	38	23
1	28	261	33
2	7	39	288

In [345]:
```
print(classification_report(ytest,y_pred8))
```

```
              precision    recall  f1-score   support

           0       0.88      0.81      0.84       321
           1       0.77      0.81      0.79       322
           2       0.84      0.86      0.85       334

    accuracy                           0.83       977
   macro avg       0.83      0.83      0.83       977
weighted avg       0.83      0.83      0.83       977
```

MACHINE LEARNING MODELS

liblinear

In [346]:
```
lib = LogisticRegression(solver='liblinear',random_state = 86)
```

In [347]:
```
model9 = lib.fit(X_train,y_train)
```

In [348]:
```
accuracy_score(model9.predict(X_train),y_train) # accuracy of train data
```

Out[348]:
```
0.8911014312383323
```

In [349]:
```
y_pred9 = model9.predict(X_test)
```

evaluation

In [350]:
```
acc9 = accuracy_score(y_pred9,y_test)
print('Accuracy:',acc9)
```

```
Accuracy: 0.8606965174129353
```

In [351]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred9,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[351]:

	0	1	2
0	304	9	7
1	30	29	1
2	6	3	13

In [352]:
```
print(classification_report(y_test,y_pred9))
```

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       320
           1       0.71      0.48      0.57        60
           2       0.62      0.59      0.60        22

    accuracy                           0.86       402
   macro avg       0.74      0.67      0.70       402
weighted avg       0.85      0.86      0.85       402
```

liblinear - smote

In [353]:
```
lib = LogisticRegression(solver='liblinear',random_state = 86)
```

In [354]:
```
model10 = lib.fit(Xtrain,ytrain)
```

In [355]:
```
accuracy_score(model10.predict(Xtrain),ytrain)
```

Out[355]:
```
0.8242827868852459
```

In [356]:
```
y_pred10 = model10.predict(Xtest)
```

evaluation

In [357]:
```
acc10 = accuracy_score(y_pred10,ytest)
print('Accuracy:',acc10)
```

```
Accuracy: 0.8331627430910952
```

In [358]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred10,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[358]:

	0	1	2
0	263	28	30
1	33	253	36
2	2	34	298

In [359]:
```
print(classification_report(ytest,y_pred10))
```

```
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       321
           1       0.80      0.79      0.79       322
           2       0.82      0.89      0.85       334

    accuracy                           0.83       977
   macro avg       0.83      0.83      0.83       977
weighted avg       0.83      0.83      0.83       977
```

lbfgs

In [360]:
```
lbfgs = LogisticRegression(solver='lbfgs',random_state = 86)
```

In [361]:
```
model11 = lbfgs.fit(X_train,y_train)
```

In [362]:
```
accuracy_score(model11.predict(X_train),y_train)
```

Out[362]: 0.8892345986309894

In [363]:
```
y_pred11 = model11.predict(X_test)
```

evaluation

In [364]:
```
acc11 =accuracy_score(y_pred11,y_test)
print('Accuracy:',acc11)
```

```
Accuracy: 0.8631840796019901
```

In [365]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred11,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[365]:

	0	1	2
0	304	10	6
1	30	29	1
2	5	3	14

In [366]:
```
print(classification_report(y_test,y_pred11))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92       320
           1       0.69      0.48      0.57        60
           2       0.67      0.64      0.65        22

    accuracy                           0.86       402
   macro avg       0.75      0.69      0.71       402
weighted avg       0.85      0.86      0.85       402
```

lbfgs - smote

In [367]:
```
lbfgs = LogisticRegression(solver='lbfgs',random_state = 86)
```

In [368]:
```
model12 = lbfgs.fit(Xtrain,ytrain)
```

In [369]:
```
accuracy_score(model12.predict(Xtrain),ytrain)
```

Out[369]: 0.8271004098360656

In [370]:
```
y_pred12 = model12.predict(Xtest)
```

evaluation

In [371]:
```
acc12 = accuracy_score(y_pred12,ytest)
print('Accuracy:',acc12)
```

```
Accuracy: 0.8280450358239508
```

In [372]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred12,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[372]:

	0	1	2
0	260	38	23
1	28	261	33
2	7	39	288

In [373]:
```
print(classification_report(ytest,y_pred12))
```

```
              precision    recall  f1-score   support

           0       0.88      0.81      0.84       321
           1       0.77      0.81      0.79       322
           2       0.84      0.86      0.85       334

    accuracy                           0.83       977
   macro avg       0.83      0.83      0.83       977
weighted avg       0.83      0.83      0.83       977
```

6) DECISION TREE

In [374]:
```
from sklearn.tree import DecisionTreeClassifier #decision tree after all statistical test
```

In [375]:
```
dt = DecisionTreeClassifier(random_state = 86)
```

In [376]:
```
params = {'criterion' : ['gini','entropy'],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split' : [i for i in range (1,11)]}
```

search = GridSearchCV(dt,params,cv = 10) search.fit(X_train,y_train) search.best_params_

In [377]:
```
dt = DecisionTreeClassifier(criterion = 'entropy',max_depth = 6,min_samples_split = 8,random_state = 86)
```

In [378]:
```
model13 = dt.fit(X_train,y_train)
```

In [379]:
```
accuracy_score(model13.predict(X_train),y_train)
```

Out[379]: 0.9471064094586186

In [380]:
```
y_pred13 = model13.predict(X_test)
```

evaluation

In [381]:
```
acc13= accuracy_score(y_pred13,y_test)
print('Accuracy:',acc13)
```

```
Accuracy: 0.9203980099502488
```

In [382]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred13,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[382]:

	0	1	2
0	313	2	5
1	22	38	0
2	2	1	19

In [383]:
```
print(classification_report(y_test,y_pred13))
```

```
              precision    recall  f1-score   support

           0       0.93      0.98      0.95       320
           1       0.93      0.63      0.75        60
           2       0.79      0.86      0.83        22

    accuracy                           0.92       402
   macro avg       0.88      0.83      0.84       402
weighted avg       0.92      0.92      0.92       402
```

7) DECISION TREE - smote

In [384]:
```
dt = DecisionTreeClassifier(random_state = 86)
```

In [385]:
```
params = {'criterion' : ['gini','entropy'],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split' : [i for i in range (1,11)]}
```

search = GridSearchCV(dt,params,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [386]:
```
dt = DecisionTreeClassifier(criterion = 'entropy',max_depth = None ,min_samples_split = 4,random_state = 83)
```

In [387]:
```
model14 = dt.fit(Xtrain,ytrain)
```

In [388]:
```
accuracy_score(model14.predict(Xtrain),ytrain)
```

Out[388]: 0.9969262295081968

In [389]:
```
y_pred14 = model14.predict(Xtest)
```

evaluation

In [390]:
```
acc14 = accuracy_score(y_pred14,ytest)
print('Accuracy:',acc14)
```

```
Accuracy: 0.9785056294779939
```

In [391]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred14,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[391]:

	0	1	2
0	301	15	5
1	0	321	1
2	0	0	334

In [392]:
```
print(classification_report(ytest,y_pred14))
```

```
              precision    recall  f1-score   support

           0       1.00      0.94      0.97       321
           1       0.96      1.00      0.98       322
           2       0.98      1.00      0.99       334

    accuracy                           0.98       977
   macro avg       0.98      0.98      0.98       977
weighted avg       0.98      0.98      0.98       977
```

8) RANDOM FOREST

In [393]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [394]:
```
rf = RandomForestClassifier(random_state = 86)
```

In [395]:
```
params = {'criterion' : ['gini','entropy'],
          'n_estimators':[20,40,60,80,100,120],
          'max_features':['sqrt',0.25,0.50,0.75,1],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split':[i for i in range(1,11,1)]}
```

search = GridSearchCV(rf,params,n_jobs = -1,cv = 10) search.fit(X_train,y_train) search.best_params_

In [396]:
```
rf = RandomForestClassifier(criterion = 'entropy',max_features = 0.75,max_depth = None,min_samples_split = 2,n_estimators = 40
```

In [397]:
```
model15 = rf.fit(X_train,y_train)
```

In [398]:
```
accuracy_score(model15.predict(X_train),y_train)
```

Out[398]:
```
0.9981331673926571
```

In [399]:
```
y_pred15 = model15.predict(X_test)
```

evaluation

In [400]:
```
acc15 = accuracy_score(y_test,y_pred15)
print('Accuracy:',acc15)
```

```
Accuracy: 0.9402985074626866
```

In [401]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred15,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[401]:

	0	1	2
0	317	3	0
1	18	42	0
2	2	1	19

In [402]:
```
print(classification_report(y_test,y_pred15))
```

```
              precision    recall  f1-score   support

           0       0.94      0.99      0.96       320
           1       0.91      0.70      0.79        60
           2       1.00      0.86      0.93        22

    accuracy                           0.94       402
   macro avg       0.95      0.85      0.89       402
weighted avg       0.94      0.94      0.94       402
```

9) RANDOM FOREST -smote

In [403]:
```
rf = RandomForestClassifier(random_state = 86)
```

In [404]:
```
params = {'criterion' : ['gini','entropy'],
          'n_estimators':[20,40,60,80,100,120],
          'max_features':['sqrt',0.25,0.50,0.75,1],
          'max_depth': [2,3,4,5,6,7,8,9,None],
          'min_samples_split':[i for i in range(1,11,1)]}
```

search = GridSearchCV(rf,params,n_jobs = -1,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [405]:
```
rf = RandomForestClassifier(criterion = 'entropy',max_depth = None,max_features = 'sqrt',min_samples_split = 3,n_estimators = 
```

In [406]:
```
model16 = rf.fit(Xtrain,ytrain)
```

In [407]:
```
accuracy_score(model16.predict(Xtrain),ytrain)
```

Out[407]:
```
0.9971823770491803
```

In [408]:
```
y_pred16 = model16.predict(Xtest)
```

evaluation

In [409]:
```
acc16 = accuracy_score(ytest,y_pred16)
print('Accuracy:',acc16)
```

```
Accuracy: 0.9866939611054247
```

In [410]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred16,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[410]:

	0	1	2
0	309	11	1
1	0	321	1
2	0	0	334

In [411]:
```
print(classification_report(ytest,y_pred16))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       321
           1       0.97      1.00      0.98       322
           2       0.99      1.00      1.00       334

    accuracy                           0.99       977
   macro avg       0.99      0.99      0.99       977
weighted avg       0.99      0.99      0.99       977
```

10) ADABOOST

In [412]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [413]:
```
ada = AdaBoostClassifier(random_state = 86)
```

In [414]:
```
params = {'n_estimators' : [i for i in range(0,500,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001]}
```

search = GridSearchCV(ada,params,n_jobs = -1,cv = 10) search.fit(X_train,y_train) search.best_params_

In [415]:
```
ada = AdaBoostClassifier(n_estimators = 100,learning_rate = 0.2,random_state = 83)
```

In [416]:
```
model17 = ada.fit(X_train,y_train)
```

In [417]:
```
accuracy_score(model17.predict(X_train),y_train)
```

Out[417]:
```
0.9029247044181705
```

In [418]:
```
y_pred17 = model17.predict(X_test)
```

evaluation

In [419]:
```
acc17 = accuracy_score(y_test,y_pred17)
print('Accuracy:',acc17)
```

```
Accuracy: 0.900497512437811
```

In [420]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred17,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[420]:

	0	1	2
0	308	11	1
1	22	37	1
2	4	1	17

In [421]:
```
print(classification_report(y_test,y_pred17))
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       320
           1       0.76      0.62      0.68        60
           2       0.89      0.77      0.83        22

    accuracy                           0.90       402
   macro avg       0.86      0.78      0.82       402
weighted avg       0.90      0.90      0.90       402
```

10) ADABOOST - smote

In [422]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [423]:
```
ada = AdaBoostClassifier(random_state = 86)
```

In [424]:
```
params = {'n_estimators' : [i for i in range(0,500,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001]}
```

search = GridSearchCV(ada,params,n_jobs = -1,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [425]:
```
ada = AdaBoostClassifier(n_estimators = 350,learning_rate = 0.2,random_state = 83)
```

In [426]:
```
model18 = ada.fit(Xtrain,ytrain)
```

In [427]:
```
accuracy_score(model18.predict(Xtrain),ytrain)
```

Out[427]: 0.8652663934426229

In [428]:
```
y_pred18 = model18.predict(Xtest)
```

evaluation

In [429]:
```
acc18 = accuracy_score(ytest,y_pred18)
print('Accuracy:',acc18)
```

```
Accuracy: 0.8894575230296827
```

In [430]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred18,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[430]:

	0	1	2
0	260	54	7
1	28	283	11
2	0	8	326

In [431]:
```
print(classification_report(ytest,y_pred18))
```

```
              precision    recall  f1-score   support

           0       0.90      0.81      0.85       321
           1       0.82      0.88      0.85       322
           2       0.95      0.98      0.96       334

    accuracy                           0.89       977
   macro avg       0.89      0.89      0.89       977
weighted avg       0.89      0.89      0.89       977
```

11) KNN

In [432]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [433]:
```
knn = KNeighborsClassifier()
```

In [434]:
```
param = {'algorithm' : ['ball_tree','kd_tree','brute'],
         'n_neighbors' :list(np.arange(1,90,2)),
         'weights' :['uniform','distance'],
         'metric': ['euclidean','manhanttan','minkowki']}
```

search = GridSearchCV(knn,param,cv = 10) search.fit(X_train,y_train) search.best_params_

In [435]:
```
knn = KNeighborsClassifier(algorithm = 'ball_tree',metric = 'euclidean',n_neighbors = 3,weights = 'distance')
```

In [436]:
```
model19 = knn.fit(X_train,y_train)
```

In [437]:
```
accuracy_score(model19.predict(X_train),y_train)
```

Out[437]:
```
0.998755444928438
```

In [438]:
```
y_pred19 = model19.predict(X_test)
```

evaluation

In [439]:
```
acc19 = accuracy_score(y_test,y_pred19)
print('Accuracy:',acc19)
```

```
Accuracy: 0.9029850746268657
```

In [440]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred19,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[440]:

	0	1	2
0	307	11	2
1	20	38	2
2	3	1	18

In [441]:
```
print(classification_report(y_test,y_pred19))
```

```
              precision    recall  f1-score   support

           0       0.93      0.96      0.94       320
           1       0.76      0.63      0.69        60
           2       0.82      0.82      0.82        22

    accuracy                           0.90       402
   macro avg       0.84      0.80      0.82       402
weighted avg       0.90      0.90      0.90       402
```

12) KNN - smote

In [442]:
```
knn = KNeighborsClassifier()
```

In [443]:
```
param = {'algorithm' : ['ball_tree','kd_tree','brute'],
         'n_neighbors' :list(np.arange(1,90,2)),
         'weights' :['uniform','distance'],
         'metric': ['euclidean','manhanttan','minkowki']}
```

search = GridSearchCV(knn,param,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [444]:
```
knn = KNeighborsClassifier(algorithm = 'brute',metric = 'euclidean',n_neighbors = 1,weights = 'uniform')
```

In [445]:
```
model20 = knn.fit(Xtrain,ytrain)
```

In [446]:
```
accuracy_score(model20.predict(Xtrain),ytrain)
```

Out[446]: 0.9969262295081968

In [447]:
```
y_pred20 = model20.predict(Xtest)
```

evaluation

In [448]:
```
acc20 = accuracy_score(ytest,y_pred20)
print('Accuracy:',acc20)
```

```
Accuracy: 0.9856704196519959
```

In [449]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred20,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[449]:

	0	1	2
0	307	13	1
1	0	322	0
2	0	0	334

In [450]:
```
print(classification_report(ytest,y_pred20))
```

```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98       321
           1       0.96      1.00      0.98       322
           2       1.00      1.00      1.00       334

    accuracy                           0.99       977
   macro avg       0.99      0.99      0.99       977
weighted avg       0.99      0.99      0.99       977
```

13) SVM

rbf

In [451]:
```
from sklearn.svm import SVC
svc=SVC(kernel='rbf',random_state = 86)
```

In [452]:
```
model21 = svc.fit(X_train,y_train)
```

In [453]:
```
accuracy_score(model21.predict(X_train),y_train)
```

Out[453]: 0.9153702551337897

In [454]:
```
y_pred21 = model21.predict(X_test)
```

evaluation

In [455]:
```
acc21 = accuracy_score(y_pred21,y_test)
print('Accuracy:',acc21)
```

```
Accuracy: 0.8855721393034826
```

In [456]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred21,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[456]:

	0	1	2
0	310	6	4
1	31	29	0
2	3	2	17

In [457]:
```
print(classification_report(y_test,y_pred21))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       320
           1       0.78      0.48      0.60        60
           2       0.81      0.77      0.79        22

    accuracy                           0.89       402
   macro avg       0.83      0.74      0.77       402
weighted avg       0.88      0.89      0.88       402
```

rbf - smote

In [458]:
```
from sklearn.svm import SVC
svc=SVC(kernel='rbf',random_state = 86)
```

In [459]:
```
model22 = svc.fit(Xtrain,ytrain)
```

In [460]:
```
accuracy_score(model22.predict(Xtrain),ytrain)
```

Out[460]: 0.9182889344262295

In [461]:
```
y_pred22 = model22.predict(Xtest)
```

evaluation

In [462]:
```
acc22 = accuracy_score(y_pred22,ytest)
print('Accuracy:',acc22)
```

```
Accuracy: 0.9263050153531218
```

In [463]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred22,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[463]:

	0	1	2
0	286	30	5
1	18	294	10
2	3	6	325

In [464]:
```
print(classification_report(ytest,y_pred22))
```

```
              precision    recall  f1-score   support

           0       0.93      0.89      0.91       321
           1       0.89      0.91      0.90       322
           2       0.96      0.97      0.96       334

    accuracy                           0.93       977
   macro avg       0.93      0.93      0.93       977
weighted avg       0.93      0.93      0.93       977
```

linear

In [465]:
```
from sklearn.svm import SVC
svc=SVC(kernel='linear',random_state = 86)
```

In [466]:
```
model23 = svc.fit(X_train,y_train)
```

In [467]:
```
accuracy_score(model23.predict(X_train),y_train)
```

Out[467]: 0.8929682638456752

In [468]:
```
y_pred23 = model23.predict(X_test)
```

evaluation

In [469]:
```
acc23 = accuracy_score(y_pred23,y_test)
print('Accuracy:',acc23)
```

```
Accuracy: 0.8656716417910447
```

In [470]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred23,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[470]:

	0	1	2
0	304	10	6
1	29	30	1
2	5	3	14

In [471]:
```
print(classification_report(y_test,y_pred23))
```

```
              precision    recall  f1-score   support

           0       0.90      0.95      0.92       320
           1       0.70      0.50      0.58        60
           2       0.67      0.64      0.65        22

    accuracy                           0.87       402
   macro avg       0.75      0.70      0.72       402
weighted avg       0.86      0.87      0.86       402
```

linear - smote

In [472]:
```
from sklearn.svm import SVC
svc=SVC(kernel='linear',random_state = 86)
```

In [473]:
```
model24 = svc.fit(Xtrain,ytrain)
```

In [474]:
```
accuracy_score(model24.predict(Xtrain),ytrain)
```

Out[474]:
```
0.828125
```

In [475]:
```
y_pred24 = model24.predict(Xtest)
```

evaluation

In [476]:
```
acc24 = accuracy_score(y_pred24,ytest)
print('Accuracy:',acc24)
```

```
Accuracy: 0.8423746161719551
```

In [477]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred24,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[477]:

	0	1	2
0	260	34	27
1	26	265	31
2	0	36	298

In [478]:
```
print(classification_report(ytest,y_pred24))
```

```
              precision    recall  f1-score   support

           0       0.91      0.81      0.86       321
           1       0.79      0.82      0.81       322
           2       0.84      0.89      0.86       334

    accuracy                           0.84       977
   macro avg       0.85      0.84      0.84       977
weighted avg       0.85      0.84      0.84       977
```

poly

In [479]:
```
from sklearn.svm import SVC
svc=SVC(kernel='poly',degree = 3,random_state = 86)
```

In [480]:
```
model25 = svc.fit(X_train,y_train)
```

In [481]:
```
accuracy_score(model25.predict(X_train),y_train)
```

Out[481]: 0.9222153080273802

In [482]:
```
y_pred25 = model25.predict(X_test)
```

evaluation

In [483]:
```
acc25 = accuracy_score(y_pred25,y_test)
print('Accuracy:',acc25)
```

```
Accuracy: 0.8805970149253731
```

In [484]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred25,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[484]:

	0	1	2
0	310	6	4
1	33	26	1
2	2	2	18

In [485]:
```
print(classification_report(y_test,y_pred25))
```

```
              precision    recall  f1-score   support

           0       0.90      0.97      0.93       320
           1       0.76      0.43      0.55        60
           2       0.78      0.82      0.80        22

    accuracy                           0.88       402
   macro avg       0.82      0.74      0.76       402
weighted avg       0.87      0.88      0.87       402
```

poly - smote

In [486]:
```
from sklearn.svm import SVC
svc=SVC(kernel='poly',degree = 3,random_state = 86)
```

In [487]:
```
model26 = svc.fit(Xtrain,ytrain)
```

In [488]:
```
accuracy_score(model26.predict(Xtrain),ytrain)
```

Out[488]: 0.9111168032786885

In [489]:
```
y_pred26 = model26.predict(Xtest)
```

evaluation

In [490]:
```
acc26 = accuracy_score(y_pred26,ytest)
print('Accuracy:',acc26)
```

```
Accuracy: 0.9211873080859775
```

In [491]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred26,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[491]:

	0	1	2
0	280	36	5
1	18	298	6
2	3	9	322

In [492]:
```
print(classification_report(ytest,y_pred26))
```

```
              precision    recall  f1-score   support

           0       0.93      0.87      0.90       321
           1       0.87      0.93      0.90       322
           2       0.97      0.96      0.97       334

    accuracy                           0.92       977
   macro avg       0.92      0.92      0.92       977
weighted avg       0.92      0.92      0.92       977
```

14) NAIVE BAYES

GaussianNB

In [493]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

In [494]:
```
model27 = gnb.fit(X_train,y_train)
```

In [495]:
```
accuracy_score(model27.predict(X_train),y_train)
```

Out[495]: 0.8158058494088364

In [496]:
```
y_pred27 = model27.predict(X_test)
```

evaluation

In [497]:
```
acc27 = accuracy_score(y_test,y_pred27)
print('Accuracy:',acc27)
```

```
Accuracy: 0.8233830845771144
```

In [498]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred27,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[498]:

	0	1	2
0	267	35	18
1	10	47	3
2	1	4	17

In [499]:
```
print(classification_report(y_test,y_pred27))
```

```
              precision    recall  f1-score   support

           0       0.96      0.83      0.89       320
           1       0.55      0.78      0.64        60
           2       0.45      0.77      0.57        22

    accuracy                           0.82       402
   macro avg       0.65      0.80      0.70       402
weighted avg       0.87      0.82      0.84       402
```

GaussianNB - smote

In [500]:
```
gnb = GaussianNB()
model28 = gnb.fit(Xtrain,ytrain)
```

In [501]:
```
accuracy_score(model28.predict(Xtrain),ytrain)
```

Out[501]: 0.7871413934426229

In [502]:
```
y_pred28 = model28.predict(Xtest)
```

evaluation

In [503]:
```
acc28 = accuracy_score(y_pred28,ytest)
print('Accuracy:',acc28)
```

```
Accuracy: 0.8055271238485159
```

In [504]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred28,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[504]:

	0	1	2
0	238	39	44
1	19	283	20
2	13	55	266

In [505]:
```
print(classification_report(ytest,y_pred28))
```

```
              precision    recall  f1-score   support

           0       0.88      0.74      0.81       321
           1       0.75      0.88      0.81       322
           2       0.81      0.80      0.80       334

    accuracy                           0.81       977
   macro avg       0.81      0.81      0.81       977
weighted avg       0.81      0.81      0.81       977
```

BernoulliNB

In [506]:
```
from sklearn.naive_bayes import BernoulliNB
```

In [507]:
```
bnb = BernoulliNB()
```

In [508]:
```
model29 = bnb.fit(X_train,y_train)
```

In [509]:
```
accuracy_score(model29.predict(X_train),y_train)
```

Out[509]:
```
0.8189172370877411
```

In [510]:
```
y_pred29 = model29.predict(X_test)
```

evaluation

In [511]:
```
acc29 = accuracy_score(y_test,y_pred29)
print('Accuracy:',acc29)
```

```
Accuracy: 0.8333333333333334
```

In [512]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred29,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[512]:

	0	1	2
0	284	25	11
1	20	40	0
2	8	3	11

In [513]:
```
print(classification_report(y_test,y_pred29))
```

```
              precision    recall  f1-score   support

           0       0.91      0.89      0.90       320
           1       0.59      0.67      0.62        60
           2       0.50      0.50      0.50        22

    accuracy                           0.83       402
   macro avg       0.67      0.68      0.67       402
weighted avg       0.84      0.83      0.84       402
```

BernoulliNB - smote

In [514]:
```
bnb = BernoulliNB()
```

In [515]:
```
model30 = bnb.fit(Xtrain,ytrain)
```

In [516]:
```
y_pred30 = model30.predict(Xtest)
```

evaluation

In [517]:
```
acc30 = accuracy_score(ytest,y_pred30)
print('Accuracy:',acc30)
```

```
Accuracy: 0.7604912998976459
```

In [518]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred30,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[518]:

	0	1	2
0	225	28	68
1	42	261	19
2	16	61	257

In [519]:
```
print(classification_report(ytest,y_pred30))
```

```
              precision    recall  f1-score   support

           0       0.80      0.70      0.75       321
           1       0.75      0.81      0.78       322
           2       0.75      0.77      0.76       334

    accuracy                           0.76       977
   macro avg       0.76      0.76      0.76       977
weighted avg       0.76      0.76      0.76       977
```

15) GRADBOOST

In [520]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [521]:
```
gbc = GradientBoostingClassifier(random_state = 83)
```

In [522]:
```
params = {'n_estimators' : [i for i in range(0,400,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001],
           'max_depth':[3,4,5,6,7,8,9,None],
           'max_features':['sqrt',0.25,0.50,0.75,1]}
```

search = GridSearchCV(gbc,params,n_jobs = -1,cv = 10) search.fit(X_train,y_train) search.best_params_

In [523]:
```
gbc = GradientBoostingClassifier(n_estimators = 100, learning_rate = 0.05,max_depth = 5, max_features = 'sqrt',random_state = 
```

In [524]:
```
model31 = gbc.fit(X_train,y_train)
```

In [525]:
```
y_pred31 = model31.predict(X_test)
```

evaluation

In [526]:
```
acc31 = accuracy_score(y_test,y_pred31)
print('Accuracy:',acc31)
```

```
Accuracy: 0.9328358208955224
```

In [527]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred31,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[527]:

	0	1	2
0	313	7	0
1	17	43	0
2	2	1	19

In [528]:
```
print(classification_report(y_test,y_pred31))
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       320
           1       0.84      0.72      0.77        60
           2       1.00      0.86      0.93        22

    accuracy                           0.93       402
   macro avg       0.93      0.85      0.89       402
weighted avg       0.93      0.93      0.93       402
```

16) GRADBOOST - smote

In [529]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [530]:
```
gbc = GradientBoostingClassifier(random_state = 86)
```

In [531]:
```
params = {'n_estimators' : [i for i in range(0,400,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001],
           'max_depth':[3,4,5,6,7,8,9,None],
           'max_features':['sqrt',0.25,0.50,0.75,1]}
```

search = GridSearchCV(gbc,params,n_jobs = -1,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [532]:
```
gbc = GradientBoostingClassifier(n_estimators = 200,learning_rate = 0.05,max_depth = None,max_features = 0.25,random_state = 8
```

In [533]:
```
model32 = gbc.fit(Xtrain,ytrain)
```

In [534]:
```
y_pred32 = model32.predict(Xtest)
```

evaluation

In [535]:
```
acc32 = accuracy_score(ytest,y_pred32)
print('Accuracy:',acc32)
```

```
Accuracy: 0.9938587512794268
```

In [536]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred32,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[536]:

	0	1	2
0	316	4	1
1	0	321	1
2	0	0	334

In [537]:
```
print(classification_report(ytest,y_pred32))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       321
           1       0.99      1.00      0.99       322
           2       0.99      1.00      1.00       334

    accuracy                           0.99       977
   macro avg       0.99      0.99      0.99       977
weighted avg       0.99      0.99      0.99       977
```

17) XG BOOST

In [538]:
```
from xgboost import XGBClassifier
```

In [539]:
```
xgb = XGBClassifier(random_state = 83)
```

In [540]:
```
params = {'n_estimators' : [i for i in range(0,500,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001],
           'max_depth':[1,2,3,4,5,6,7,8,9,None],
           'min_child_weight':[1,3,5,7],
           'gamma':[0,0.1,0.2,0.3]}
```

search = GridSearchCV(xgb,params,n_jobs = -1,cv = 10) search.fit(X_train,y_train) search.best_params_

In [541]:
```
xgb = XGBClassifier(learning_rate = 0.2,n_estimators = 200, gamma = 0.1,min_child_weight = 1,max_depth = 2,random_state = 86)
```

In [542]:
```
model33 = xgb.fit(X_train,y_train)
```

In [543]:
```
y_pred33 = model33.predict(X_test)
```

evaluation

In [544]:
```
acc33 = accuracy_score(y_test,y_pred33)
print('Accuracy:',acc33)
```

```
Accuracy: 0.9303482587064676
```

In [545]:
```
pd.DataFrame(confusion_matrix(y_test,y_pred33,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[545]:

	0	1	2
0	314	6	0
1	19	41	0
2	2	1	19

In [546]:
```
print(classification_report(y_test,y_pred33))
```

```
              precision    recall  f1-score   support

           0       0.94      0.98      0.96       320
           1       0.85      0.68      0.76        60
           2       1.00      0.86      0.93        22

    accuracy                           0.93       402
   macro avg       0.93      0.84      0.88       402
weighted avg       0.93      0.93      0.93       402
```

18) XG BOOST - smote

In [547]:
```
from xgboost import XGBClassifier
```

In [548]:
```
xgb = XGBClassifier(random_state = 83)
```

In [549]:
```
params = {'n_estimators' : [i for i in range(0,500,50)],
          'learning_rate':[1.0,0.2,0.1,0.01,0.05,0.001],
           'max_depth':[1,2,3,4,5,6,7,8,9,None],
           'min_child_weight':[1,3,5,7],
           'gamma':[0,0.1,0.2,0.3]}
```

search = GridSearchCV(xgb,params,n_jobs = -1,cv = 10) search.fit(Xtrain,ytrain) search.best_params_

In [550]:
```
xgb = XGBClassifier(learning_rate = 1.0,n_estimators = 350, gamma = 0,min_child_weight = 1,max_depth = 7,random_state = 86)
```

In [551]:
```
model34 = xgb.fit(Xtrain,ytrain)
```

In [552]:
```
y_pred34 = model34.predict(Xtest)
```

evaluation

In [553]:
```
acc34 = accuracy_score(ytest,y_pred34)
print('Accuracy:',acc34)
```

```
Accuracy: 0.9907881269191402
```

In [554]:
```
pd.DataFrame(confusion_matrix(ytest,y_pred34,labels = [0,1,2]),index = [0,1,2],columns = [0,1,2])
```

Out[554]:

	0	1	2
0	313	7	1
1	0	321	1
2	0	0	334

In [555]:
```
print(classification_report(ytest,y_pred34))
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       321
           1       0.98      1.00      0.99       322
           2       0.99      1.00      1.00       334

    accuracy                           0.99       977
   macro avg       0.99      0.99      0.99       977
weighted avg       0.99      0.99      0.99       977
```

BEST MODELS

In [556]:
```
MODELS = {'Models':['DT(before analysis)','RF(before analysis)','DT','RF','Gradient Boost','XG Boost'],
          'Accuracy':[acc2,acc4,acc14,acc16,acc32,acc34]}                                    #all these models are on smote da
```

In [557]:
```
MODELS = pd.DataFrame(MODELS)
```

In [558]:
```
BEST_MODELS = MODELS.sort_values('Accuracy',ascending = False).reset_index(drop = True)
BEST_MODELS
```

Out[558]:

	Models	Accuracy
0	Gradient Boost	0.993859
1	RF(before analysis)	0.992951
2	XG Boost	0.990788
3	RF	0.986694
4	DT	0.978506
5	DT(before analysis)	0.975831