In [1]:
```
"""
@Authour:              Adharsh.S
Date created:          20.10.22
language written:      python

"""

#importing the necessary package for the usage of

import matplotlib.pyplot as plt # for ploting the file if we needed for visualization
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
sns.set(color_codes=True) # adds a nice background to the graphs
%matplotlib inline
import pandas as pd #pandas package
import numpy as np #numpy package
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis  # for analysing the LDA
```

In [2]:
```
#header_list = ["baseline value "," accelerations","fetal_health "]
df = pd.read_csv("train.csv")#names=header_list)
df.head() #head of the the data
df.shape # shape of the data
df.info() # info of the data
df.describe() # describtion of the data
```

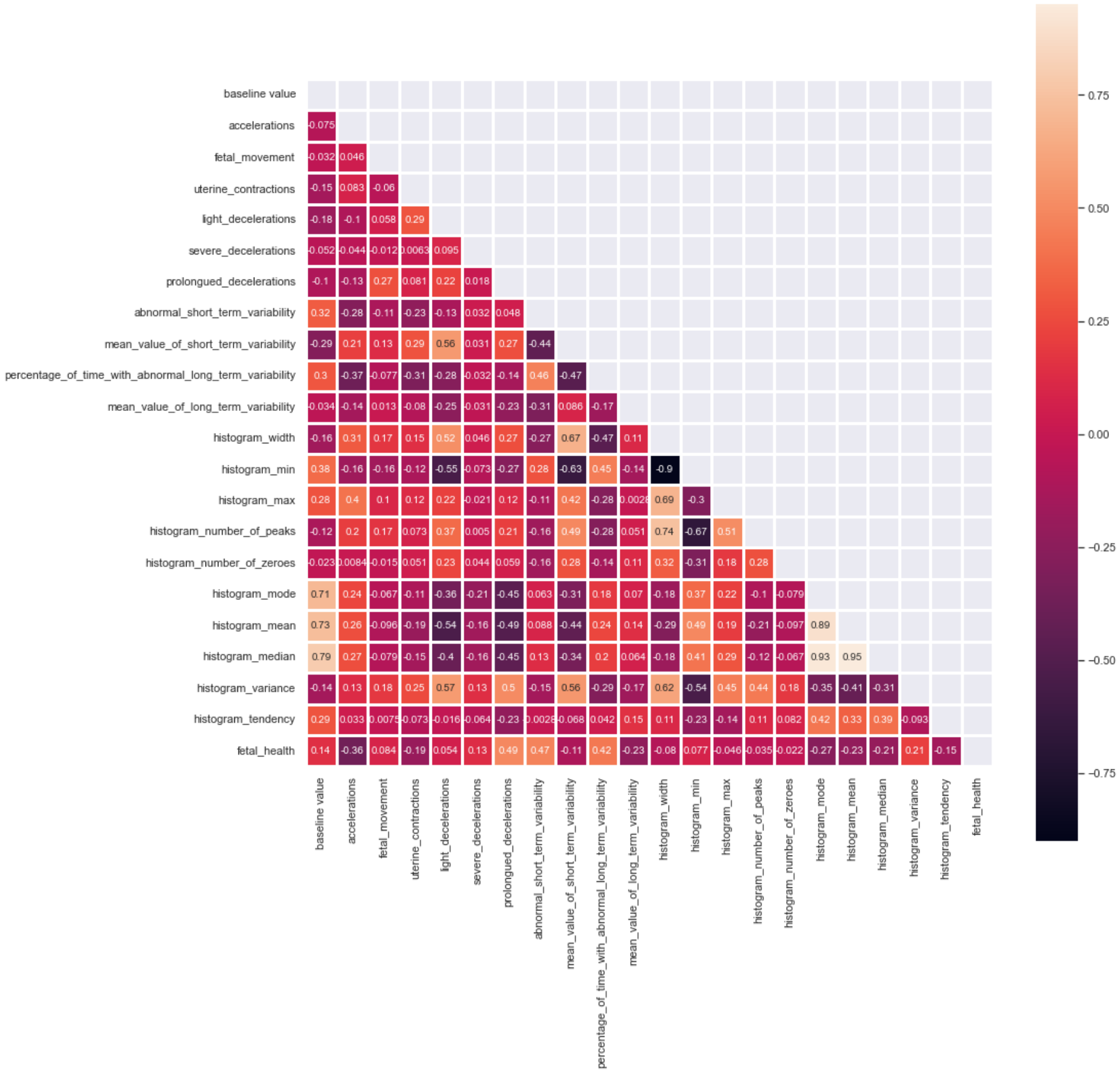
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1700 entries, 0 to 1699
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   baseline value                                          1700 non-null   int64  
 1   accelerations                                           1700 non-null   float64
 2   fetal_movement                                          1700 non-null   float64
 3   uterine_contractions                                    1700 non-null   float64
 4   light_decelerations                                     1700 non-null   float64
 5   severe_decelerations                                    1700 non-null   float64
 6   prolongued_decelerations                                1700 non-null   float64
 7   abnormal_short_term_variability                         1700 non-null   int64  
 8   mean_value_of_short_term_variability                    1700 non-null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  1700 non-null   int64  
 10  mean_value_of_long_term_variability                     1700 non-null   float64
 11  histogram_width                                         1700 non-null   int64  
 12  histogram_min                                           1700 non-null   int64  
 13  histogram_max                                           1700 non-null   int64  
 14  histogram_number_of_peaks                               1700 non-null   int64  
 15  histogram_number_of_zeroes                              1700 non-null   int64  
 16  histogram_mode                                          1700 non-null   int64  
 17  histogram_mean                                          1700 non-null   int64  
 18  histogram_median                                        1700 non-null   int64  
 19  histogram_variance                                      1700 non-null   int64  
 20  histogram_tendency                                      1700 non-null   int64  
 21  fetal_health                                            1700 non-null   int64  
dtypes: float64(8), int64(14)
memory usage: 292.3 KB
```

Out[2]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnorma
count	1700.000000	1700.000000	1700.000000	1700.000000	1700.000000	1700.000000	1700.000000	
mean	133.213529	0.003212	0.010211	0.004356	0.001899	0.000004	0.000158	
std	9.873344	0.003888	0.050124	0.002943	0.002976	0.000059	0.000587	
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000	
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000	
75%	140.000000	0.006000	0.003000	0.006000	0.003000	0.000000	0.000000	
max	159.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000	

8 rows × 22 columns

In [3]:
```
# for the visualization of the data for our understanding
Target = df["fetal_health"]
corr = df.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("dark"):
    f, ax = plt.subplots(figsize=(15, 15))
    ax = sns.heatmap(corr,mask=mask,square=True,linewidths=2.5,cmap="rocket",annot=True)
```



In [4]:
```
#for avoiding the duplicates of the data
df_dup = df.drop_duplicates(subset = None , keep = 'first', inplace = False)
```

In [5]:
```
print("Count of type 1.0 fetal health in the dataset ",len(df.loc[df["fetal_health"]==1.0])) #for counting the data
print("Count of type 2.0 fetal health in the dataset ",len(df.loc[df["fetal_health"]==2.0]))
print("Count of type 3.0 fetal health in the dataset ",len(df.loc[df["fetal_health"]==3.0]))
```

```
Count of type 1.0 fetal health in the dataset  1323
Count of type 2.0 fetal health in the dataset  236
Count of type 3.0 fetal health in the dataset  141
```

In [6]:
```
# for locating the data
X = df_dup.iloc[:,:-1]
y = df_dup.iloc[:,-1]
```

In [7]:
```
#for scaling the data for the usage
scale = StandardScaler()
X = scale.fit_transform(X)
X = pd.DataFrame(X,columns=df_dup.iloc[:,:-1].columns)
```

In [8]:
```
#for resampling the data over a random state
from imblearn.over_sampling import RandomOverSampler
ROS = RandomOverSampler(random_state=42)
X_ros, y_ros = ROS.fit_resample(X,y)
from collections import Counter
print('Resampled dataset shape %s' % Counter(y_ros))
```

```
Resampled dataset shape Counter({1: 1317, 3: 1317, 2: 1317})
```

In [9]:
```
# for allocating the test and train data for the predicting
import statsmodels.api as sm
X = sm.add_constant(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 10, test_size = 0.2)

print('X_train', X_train.shape)
print('y_train', y_train.shape)
print('X_test', X_test.shape)
print('y_test', y_test.shape)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:7: FutureWarning: pandas.Int64Index is depr
ecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,
C:\Users\DELL\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:7: FutureWarning: pandas.Float64Index is de
precated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.
  from pandas import (to_datetime, Int64Index, DatetimeIndex, Period,

X_train (1354, 22)
y_train (1354,)
X_test (339, 22)
y_test (339,)

C:\Users\DELL\anaconda3\lib\site-packages\statsmodels\tsa\tsatools.py:142: FutureWarning: In a future version of pandas
all arguments of concat except for the argument 'objs' will be keyword-only.
  x = pd.concat(x[::order], 1)
```

In [10]:
```
# for the preprocessing method
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [11]:
```
# for discriminant analysis method
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=1)
X_train = lda.fit_transform(X_train, y_train)
X_test = lda.transform(X_test)
```

In [12]:
```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(max_depth=2, random_state=0)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

In [13]:
```
#for the confusion matrix and the accuracy score too
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred)
print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))
```

```
[[243  22   1]
 [ 14  34   0]
 [  1  10  14]]
Accuracy0.8584070796460177
```

In [14]:
```
y_pred
```

Out[14]:
```
array([1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 3, 1, 1, 1,
       1, 3, 2, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 3, 1,
       1, 2, 1, 1, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1,
       3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 3, 1, 3, 2, 1, 1, 2,
       1, 1, 1, 1, 2, 1, 3, 2, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1,
       1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
       1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 2, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1,
       1, 1, 1, 1, 2, 1, 1, 3, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 3, 2, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 3, 3, 1, 2,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 2, 1, 1, 2, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 1, 2, 1, 1,
       1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
       2, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int64)
```

In [26]:
```
import csv
data = [y_pred]
file = open('/output.csv', 'w+', newline ='')
with file:
    write = csv.writer(file)
    write.writerow(data)
```

In [36]:
```
import csv
data = [y_pred]
# open the file in the write mode
with open("/output.csv", 'w', encoding='UTF8') as f:
    # create the csv writer
    writer = csv.writer(f)

    # write a row to the csv file
    writer.writerow(data)
```

In [55]:
```
from pandas import DataFrame
data = [y_pred]
df = DataFrame(data)

export_csv = df.to_csv (r'output.csv', index = None, header=True)# here you have to write path, where result file will be
print (df)
```

```
   0    1    2    3    4    5    6    7    8    9    ...  329  330  331  332  \
0    1    1    1    1    1    1    1    2    1    1  ...    1    2    1    1

   333  334  335  336  337  338
0    1    1    1    1    1    1

[1 rows x 339 columns]
```