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

In [2]:
```python
df=pd.read_csv("fetal_health.csv")
```

In [3]:
```python
df.head()
```

Out[3]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_variability	mean_value_of_short_term_variability	percentage_of_time_with_a
0	120.0	0.000	0.0	0.000	0.000	0.0	0.0	73.0	0.5	
1	132.0	0.006	0.0	0.006	0.003	0.0	0.0	17.0	2.1	
2	133.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0	2.1	
3	134.0	0.003	0.0	0.008	0.003	0.0	0.0	16.0	2.4	
4	132.0	0.007	0.0	0.008	0.000	0.0	0.0	16.0	2.4	

5 rows × 22 columns

In [4]:
```python
df.isnull().sum()
```

Out[4]:
```
baseline value                                            0
accelerations                                             0
fetal_movement                                            0
uterine_contractions                                      0
light_decelerations                                       0
severe_decelerations                                      0
prolongued_decelerations                                  0
abnormal_short_term_variability                           0
mean_value_of_short_term_variability                      0
percentage_of_time_with_abnormal_long_term_variability    0
mean_value_of_long_term_variability                       0
histogram_width                                           0
histogram_min                                             0
histogram_max                                             0
histogram_number_of_peaks                                 0
histogram_number_of_zeroes                                0
histogram_mode                                            0
histogram_mean                                            0
histogram_median                                          0
histogram_variance                                        0
histogram_tendency                                        0
fetal_health                                              0
dtype: int64
```

In [5]:
```python
df.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 [6]:
```python
df.describe()
```

Out[6]:

	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_tim
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	
mean	133.303857	0.003178	0.009481	0.004366	0.001889	0.000003	0.000159	46.990122	1.332785	
std	9.840844	0.003866	0.046666	0.002946	0.002960	0.000057	0.000590	17.192814	0.883241	
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	12.000000	0.200000	
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000	32.000000	0.700000	
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000	49.000000	1.200000	
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	0.000000	61.000000	1.700000	
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000	87.000000	7.000000	

8 rows × 22 columns

In [7]:
```python
X=df.loc[ : , df.columns != 'fetal_health']
y=df["fetal_health"]
```

In [8]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.20, random_state=42)
```

In [9]:
```python
X_train.head()
```

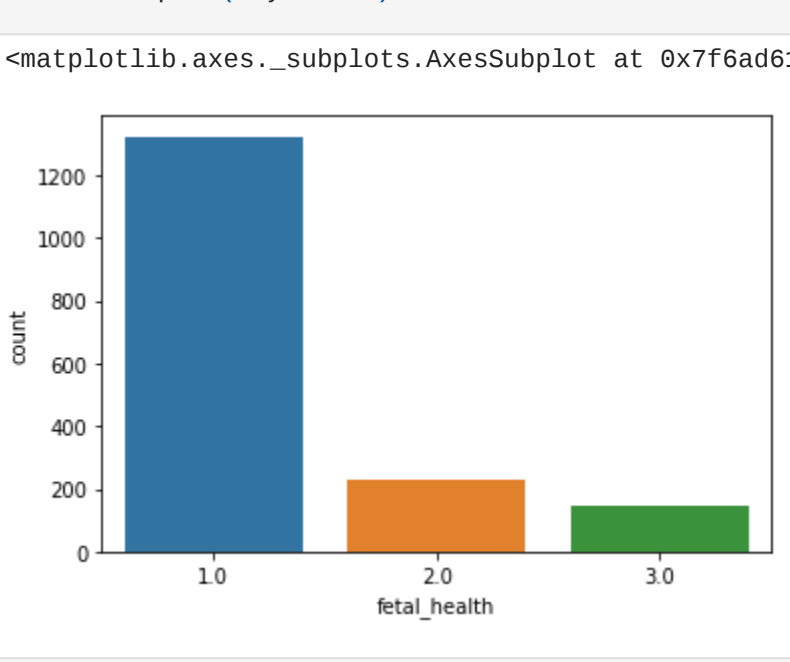
Out[9]:

	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_wi
1233	125.0	0.000	0.0	0.008	0.000	0.0	0.0	32.0	1.1	
480	140.0	0.000	0.0	0.001	0.000	0.0	0.0	60.0	0.8	
1111	122.0	0.000	0.0	0.000	0.001	0.0	0.0	20.0	1.8	
1303	137.0	0.005	0.0	0.005	0.002	0.0	0.0	36.0	0.9	
861	142.0	0.003	0.0	0.004	0.000	0.0	0.0	46.0	0.7	

5 rows × 21 columns

In [10]:
```python
sns.countplot(x=y_train)
```

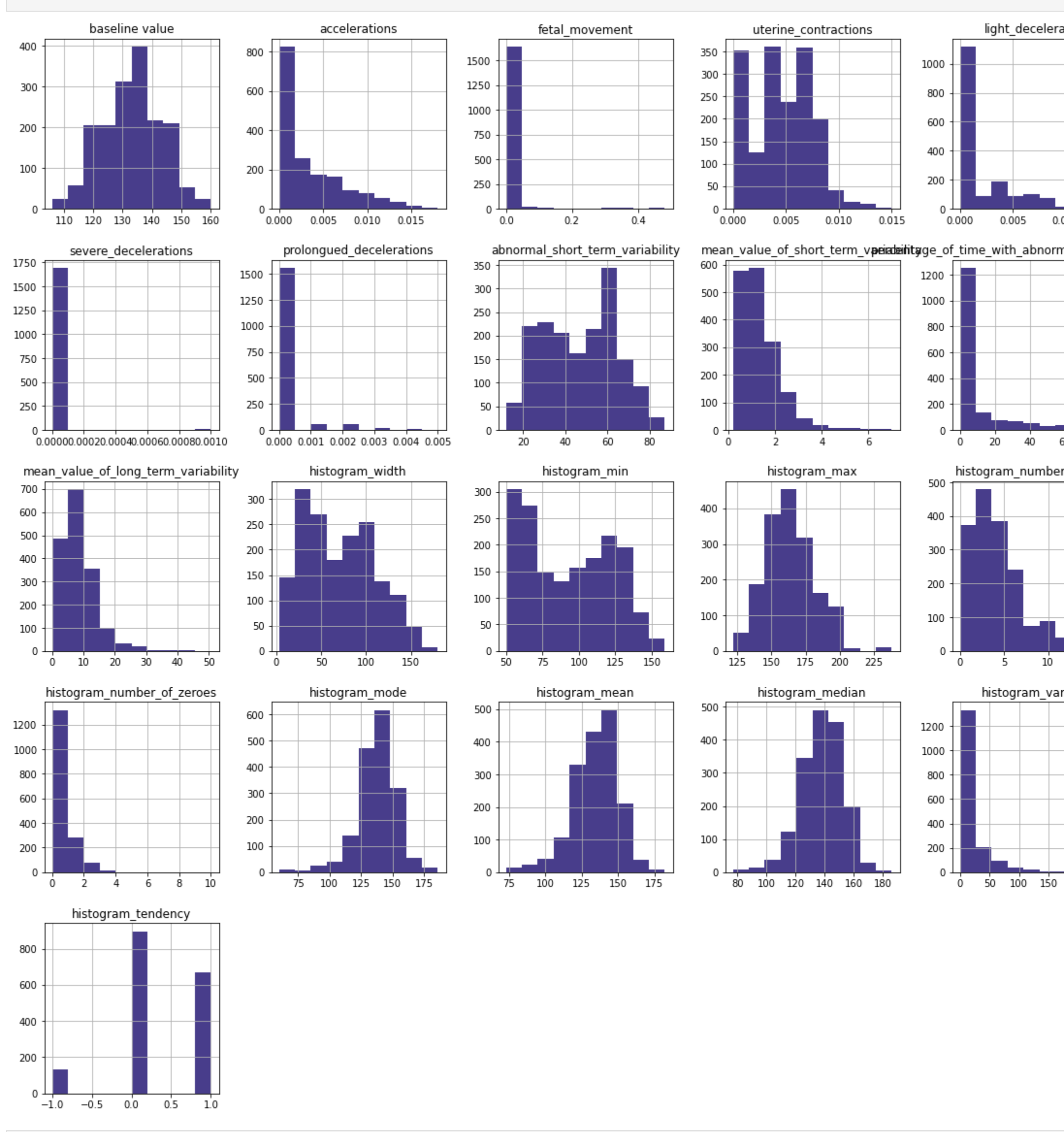
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6ad6191b50>



In [11]:
```python
#it is a unbalanced data set
```

EDA(Exploratory data analysis)

In [12]:
```python
# Evaluating distributions of the features
hist_plot = X_train.hist(figsize = (20,20), color = "#483D8B")
```



In [13]:
```python
X_train.drop("severe_decelerations", axis=1, inplace=True)
```

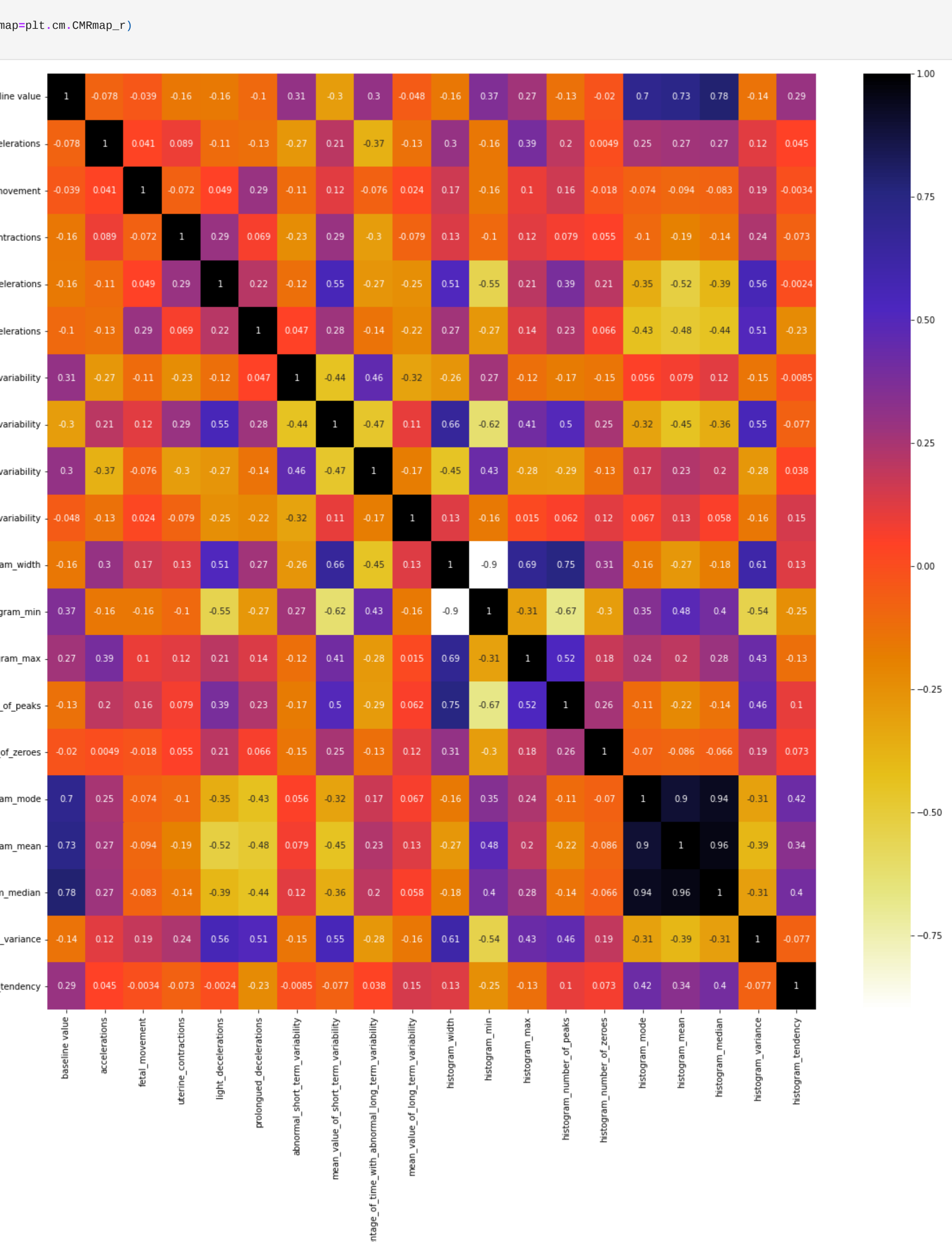
In [14]:
```python
X_train.head()
```

Out[14]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	prolongued_decelerations	abnormal_short_term_variability	mean_value_of_short_term_variability	percentage_of_time_with_abnormal_long_term
1233	125.0	0.000	0.0	0.008	0.000	0.0	32.0	1.1	
480	140.0	0.000	0.0	0.001	0.000	0.0	60.0	0.8	
1111	122.0	0.000	0.0	0.000	0.001	0.0	20.0	1.8	
1303	137.0	0.005	0.0	0.005	0.002	0.0	36.0	0.9	
861	142.0	0.003	0.0	0.004	0.000	0.0	46.0	0.7	

In [15]:
```python
## feature selection
```

In [16]:
```python
#Using Pearson Correlation
plt.figure(figsize=(20,20))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```



In [17]:
```python
X_train.shape
```

Out[17]: (1700, 20)

In [18]:
```python
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

In [19]:
```python
corr_features = correlation(df, 0.9)
len(set(corr_features))
```

Out[19]: 1

In [20]:
```python
corr_features
```

Out[20]: {'histogram_median'}

In [21]:
```python
X_train.drop(corr_features,axis=1,inplace=True)
```

In [22]:
```python
X_train.head()
```

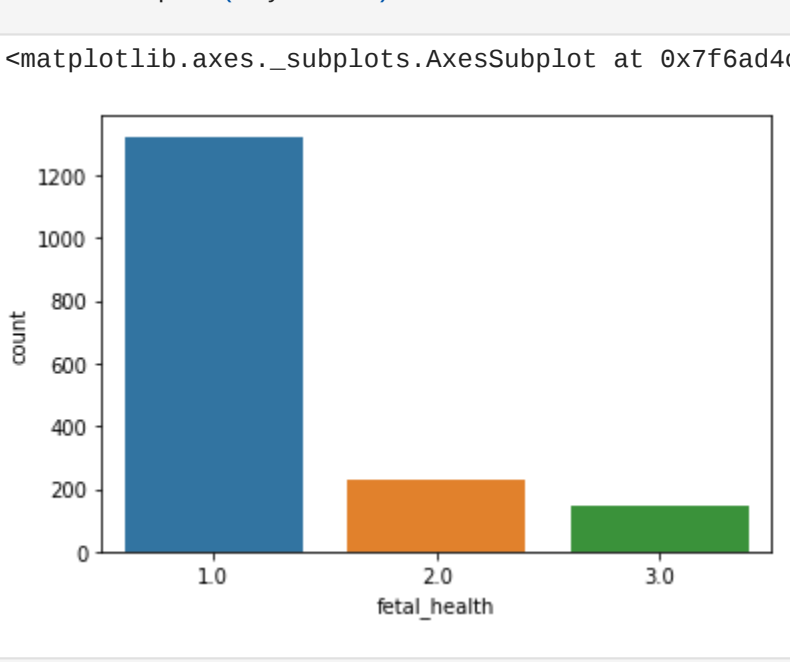
Out[22]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	prolongued_decelerations	abnormal_short_term_variability	mean_value_of_short_term_variability	percentage_of_time_with_abnormal_long_term
1233	125.0	0.000	0.0	0.008	0.000	0.0	32.0	1.1	
480	140.0	0.000	0.0	0.001	0.000	0.0	60.0	0.8	
1111	122.0	0.000	0.0	0.000	0.001	0.0	20.0	1.8	
1303	137.0	0.005	0.0	0.005	0.002	0.0	36.0	0.9	
861	142.0	0.003	0.0	0.004	0.000	0.0	46.0	0.7	

In [23]:
```python
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

In [24]:
```python
sns.countplot(x=y_train)
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6ad4c38f50>



In [25]:
```python
## RandomOverSampler to handle imbalanced data

from imblearn.over_sampling import RandomOverSampler
```

In [26]:
```python
os =  RandomOverSampler(sampling_strategy='minority')
```

In [27]:
```python
X_sam1, y_sam1 = os.fit_resample(X_train, y_train)
```

In [28]:
```python
from collections import Counter
print('Original dataset shape {}'.format(Counter(y_train)))
print('Resampled dataset shape {}'.format(Counter(y_sam1)))
```

```
Original dataset shape Counter({1.0: 1322, 2.0: 231, 3.0: 147})
Resampled dataset shape Counter({1.0: 1322, 3.0: 1322, 2.0: 231})
```

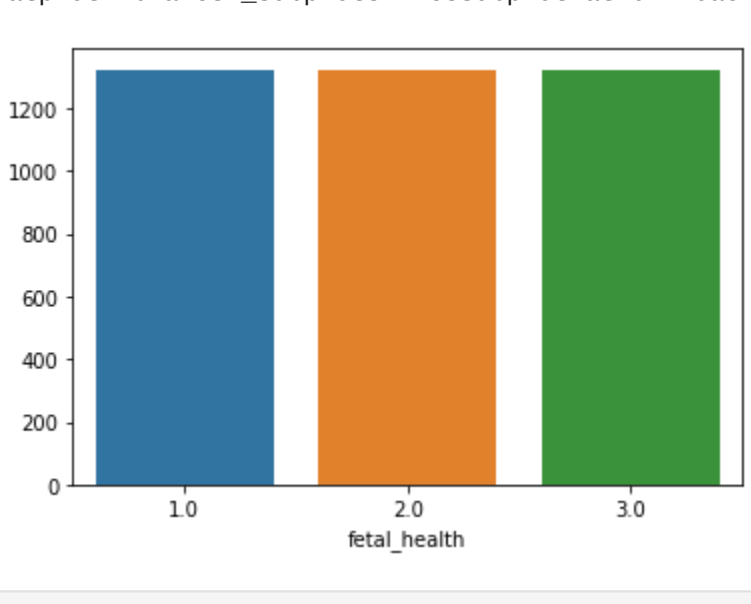
In [29]:
```python
X_training, y_training = os.fit_resample(X_sam1, y_sam1)
```

In [30]:
```python
from collections import Counter
print('Original dataset shape {}'.format(Counter(y_train)))
print('Resampled dataset shape {}'.format(Counter(y_training)))
```

```
Original dataset shape Counter({1.0: 1322, 2.0: 231, 3.0: 147})
Resampled dataset shape Counter({1.0: 1322, 2.0: 1322, 3.0: 1322})
```

In [31]:
```python
sns.countplot(x =y_training)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6acfaddf90>



In [32]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import xgboost
from sklearn.svm import SVC
```

In [33]:
```python
pipeline_lr=Pipeline([('scalar1',StandardScaler()),
                     ('lr_classifier',LogisticRegression())])
```

In [34]:
```python
pipeline_dt=Pipeline([('scalar2',StandardScaler()),
                     ('dt_classifier',DecisionTreeClassifier())])
```

In [35]:
```python
pipeline_randomforest=Pipeline([('scalar3',StandardScaler()),

                     ('rf_classifier',RandomForestClassifier())])
```

In [36]:
```python
pipeline_xg=Pipeline([('scalar4',StandardScaler()),

                     ('xg_classifier',xgboost.XGBClassifier()
)])
```

In [37]:
```python
pipeline_svc=Pipeline([('scalar5',StandardScaler()),

                     ('svc_classifier',SVC())])
```

In [38]:
```python
pipelines = [pipeline_lr, pipeline_dt, pipeline_randomforest,pipeline_xg,pipeline_svc]
```

In [39]:
```python
best_accuracy=0.0
best_classifier=0
best_pipeline=""
```

In [40]:
```python
# Dictionary of pipelines and classifier types for ease of reference
pipe_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'RandomForest',3:'XG boost',4:'Svc'}

# Fit the pipelines
for pipe in pipelines:
        pipe.fit(X_training, y_training)
```

In [41]:
```python
X_test.drop(["histogram_median","severe_decelerations"],axis=1,inplace=True)
```

In [41]:

In [42]:
```python
for i,model in enumerate(pipelines):
    print("{} Test Accuracy: {}".format(pipe_dict[i],model.score(X_test,y_test)))
```

```
Logistic Regression Test Accuracy: 0.8286384976525821
Decision Tree Test Accuracy: 0.92018779342723
RandomForest Test Accuracy: 0.9460093896713615
XG boost Test Accuracy: 0.92018779342723
Svc Test Accuracy: 0.8755868544600939
```

In [43]:
```python
for i,model in enumerate(pipelines):
    if model.score(X_test,y_test)>best_accuracy:
        best_accuracy=model.score(X_test,y_test)
        best_pipeline=model
        best_classifier=i
print('Classifier with best accuracy:{}'.format(pipe_dict[best_classifier]))
```

```
Classifier with best accuracy:RandomForest
```

In [44]:
```python
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

In [45]:
```python
for i,model in enumerate(pipelines):
    y_pred=model.predict(X_test)
    print("{}:Accuracy Score ".format(pipe_dict[i]))
    print("{}".format(accuracy_score(y_test,y_pred)))
    print("{}:Classification report: ".format(pipe_dict[i]))
    print("{}".format(classification_report(y_test,y_pred)))
    print("{}:Confusion matrix:".format(pipe_dict[i]))
    print("{}".format(confusion_matrix(y_test,y_pred)))
```

```
Logistic Regression:Accuracy Score
0.8286384976525821
Logistic Regression:Classification report:
              precision    recall  f1-score   support

         1.0       0.98      0.85      0.91       333
         2.0       0.52      0.75      0.61        64
         3.0       0.50      0.79      0.61        29

    accuracy                           0.83       426
   macro avg       0.67      0.80      0.71       426
weighted avg       0.88      0.83      0.84       426

Logistic Regression:Confusion matrix:
[[282  40  11]
 [  4  48  12]
 [  1   5  23]]
Decision Tree:Accuracy Score
0.92018779342723
Decision Tree:Classification report:
              precision    recall  f1-score   support

         1.0       0.95      0.95      0.95       333
         2.0       0.74      0.77      0.75        64
         3.0       0.93      0.97      0.95        29

    accuracy                           0.92       426
   macro avg       0.88      0.89      0.88       426
weighted avg       0.92      0.92      0.92       426

Decision Tree:Confusion matrix:
[[315  17   1]
 [ 14  49   1]
 [  1   0  28]]
RandomForest:Accuracy Score
0.9460093896713615
RandomForest:Classification report:
              precision    recall  f1-score   support

         1.0       0.97      0.96      0.97       333
         2.0       0.83      0.84      0.84        64
         3.0       0.90      0.97      0.93        29

    accuracy                           0.95       426
   macro avg       0.90      0.92      0.91       426
weighted avg       0.95      0.95      0.95       426

RandomForest:Confusion matrix:
[[321  11   1]
 [  8  54   2]
 [  1   0  28]]
XG boost:Accuracy Score
0.92018779342723
XG boost:Classification report:
              precision    recall  f1-score   support

         1.0       0.98      0.92      0.95       333
         2.0       0.72      0.91      0.80        64
         3.0       0.83      1.00      0.91        29

    accuracy                           0.92       426
   macro avg       0.84      0.94      0.88       426
weighted avg       0.93      0.92      0.92       426

XG boost:Confusion matrix:
[[305  23   5]
 [  5  58   1]
 [  0   0  29]]
Svc:Accuracy Score
0.8755868544600939
Svc:Classification report:
              precision    recall  f1-score   support

         1.0       0.99      0.88      0.93       333
         2.0       0.57      0.84      0.68        64
         3.0       0.75      0.93      0.83        29

    accuracy                           0.88       426
   macro avg       0.77      0.88      0.81       426
weighted avg       0.91      0.88      0.89       426

Svc:Confusion matrix:
[[292  39   2]
 [  3  54   7]
 [  1   1  27]]
```

In [48]:
```python
from sklearn.model_selection import GridSearchCV
```

In [49]:
```python
bestpipe = Pipeline([('scalar6',StandardScaler()),

                     ('classifier',RandomForestClassifier())])

grid_param = {
                 "classifier__n_estimators": [10, 100,200,300, 1000],
                 "classifier__max_depth":[5,8,15,25,30,None],
                 "classifier__min_samples_leaf":[1,2,5,10,15,100],
                 "classifier__max_leaf_nodes": [2, 5,10]}
# create a gridsearch of the pipeline, the fit the best model
gridsearch = GridSearchCV(bestpipe, grid_param, cv=5, verbose=0,n_jobs=-1) # Fit grid search
best_model = gridsearch.fit(X_train,y_train)
```

In [52]:
```python
best_model.best_params_
```

Out[52]:
```
{'classifier__max_depth': 30,
 'classifier__max_leaf_nodes': 10,
 'classifier__min_samples_leaf': 2,
 'classifier__n_estimators': 200}
```

In [54]:
```python
pipefinal = Pipeline([('scalar7',StandardScaler()),

                     ('classifier',RandomForestClassifier(max_depth= 30,max_leaf_nodes= 10,min_samples_leaf= 2,n_estimators= 200))
                     ])
```

In [55]:
```python
pipefinal.fit(X_training,y_training)
```

Out[55]:
```
Pipeline(steps=[('scalar7', StandardScaler()),
                ('classifier',
                 RandomForestClassifier(max_depth=30, max_leaf_nodes=10,
                                        min_samples_leaf=2,
                                        n_estimators=200))])
```

In [56]:
```python
y_predd=pipefinal.predict(X_test)
```

In [57]:
```python
    print("Accuracy Score ")
    print("{}".format(accuracy_score(y_test,y_predd)))
    print(":Classification report: ")
    print("{}".format(classification_report(y_test,y_predd)))
    print("Confusion matrix:")
    print("{}".format(confusion_matrix(y_test,y_predd)))
```

```
Accuracy Score
0.8755868544600939
:Classification report:
              precision    recall  f1-score   support

         1.0       0.98      0.87      0.92       333
         2.0       0.62      0.88      0.72        64
         3.0       0.68      0.93      0.78        29

    accuracy                           0.88       426
   macro avg       0.76      0.89      0.81       426
weighted avg       0.91      0.88      0.88       426

Confusion matrix:
[[290  33  10]
 [  5  56   3]
 [  0   2  27]]
```

In [ ]: