In [45]:

```
import numpy as np
import pandas as pd
df=pd.read_csv("/content/fetal_health.csv")
df
```

Out[45]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerat
0	120.0	0.000	0.000	0.000	0.000	
1	132.0	0.006	0.000	0.006	0.003	
2	133.0	0.003	0.000	0.008	0.003	
3	134.0	0.003	0.000	0.008	0.003	
4	132.0	0.007	0.000	0.008	0.000	
...	...	...	...	...	...	
2121	140.0	0.000	0.000	0.007	0.000	
2122	140.0	0.001	0.000	0.007	0.000	
2123	140.0	0.001	0.000	0.007	0.000	
2124	140.0	0.001	0.000	0.006	0.000	
2125	142.0	0.002	0.002	0.008	0.000	

2126 rows × 22 columns

In [46]:

```
df.shape
```

Out[46]:

```
(2126, 22)
```

In [47]:

```
df.size
```

Out[47]:

```
46772
```

In [48]:

```
df.head()
```

Out[48]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_deceleration
0	120.0	0.000	0.0	0.000	0.000	0.
1	132.0	0.006	0.0	0.006	0.003	0.
2	133.0	0.003	0.0	0.008	0.003	0.
3	134.0	0.003	0.0	0.008	0.003	0.
4	132.0	0.007	0.0	0.008	0.000	0.

5 rows × 22 columns

In [49]:

```
df.tail()
```

Out[49]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerat
2121	140.0	0.000	0.000	0.007	0.0	
2122	140.0	0.001	0.000	0.007	0.0	
2123	140.0	0.001	0.000	0.007	0.0	
2124	140.0	0.001	0.000	0.006	0.0	
2125	142.0	0.002	0.002	0.008	0.0	

5 rows × 22 columns

In [50]:

```
df.columns
```

Out[50]:

```
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 [51]:

```
df.dtypes
```

Out[51]:

```
baseline value                                            float64
accelerations                                             float64
fetal_movement                                            float64
uterine_contractions                                      float64
light_decelerations                                       float64
severe_decelerations                                      float64
prolongued_decelerations                                  float64
abnormal_short_term_variability                           float64
mean_value_of_short_term_variability                      float64
percentage_of_time_with_abnormal_long_term_variability    float64
mean_value_of_long_term_variability                       float64
histogram_width                                           float64
histogram_min                                             float64
histogram_max                                             float64
histogram_number_of_peaks                                 float64
histogram_number_of_zeroes                                float64
histogram_mode                                            float64
histogram_mean                                            float64
histogram_median                                          float64
histogram_variance                                        float64
histogram_tendency                                        float64
fetal_health                                              float64
dtype: object
```

In [52]:

```
df.isna().sum()
```

Out[52]:

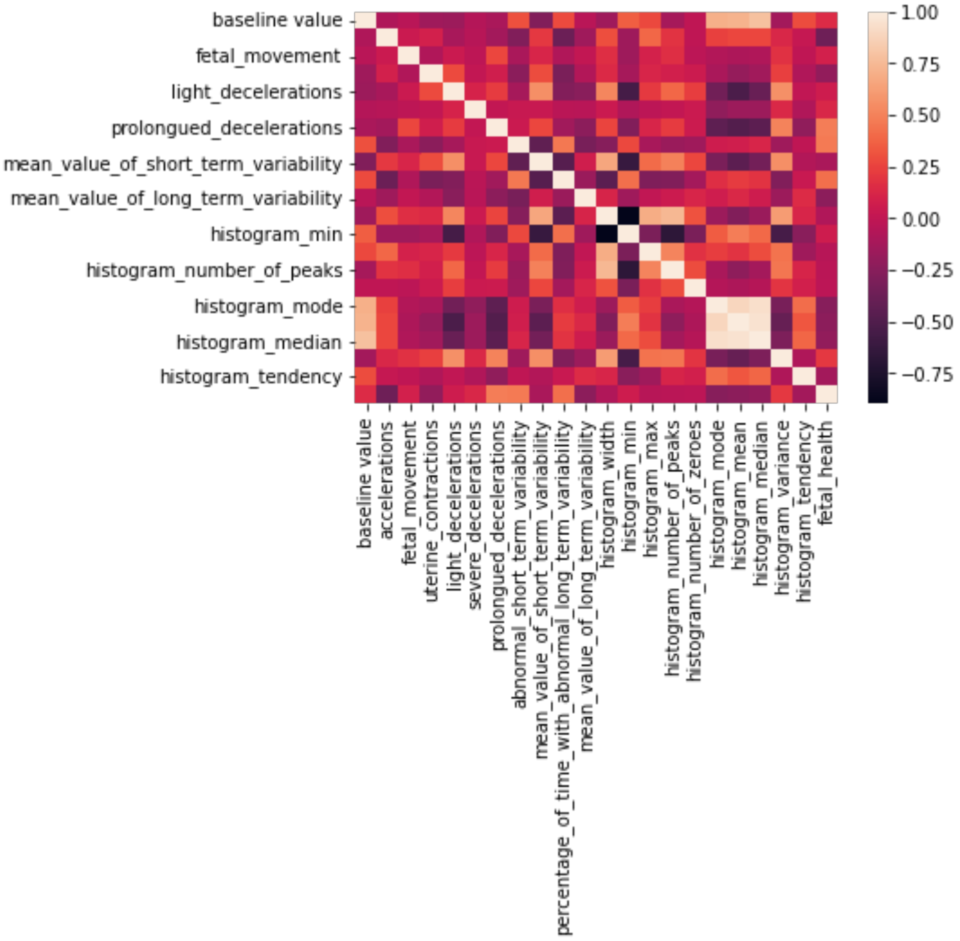
```
baseline value                                            0
accelerations                                             0
fetal_movement                                            0
uterine_contractions                                      0
light_decelerations                                       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 [53]:

```
# CORRELATION PLOT
import seaborn as snc
snc.heatmap(df.corr())
```

Out[53]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f474def8640>
```



In [54]:

```
# SEPERATING INPUT X AND OUTPUT Y
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [55]:

```
# SEPERATING TRAIN & TEST DATA
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_st
ate=42)
```

In [56]:

```
# NORMALIZATION
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

1.KNN

In [57]:

```
# ML ALGORITHM
from sklearn.neighbors import KNeighborsClassifier
model_knn=KNeighborsClassifier(n_neighbors=7)
model_knn.fit(x_train,y_train)
y_pred_knn=model_knn.predict(x_test)
```

In [58]:

```
# PERFORMANCE EVALUATION
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay,accurac
y_score,classification_report
mat=confusion_matrix(y_pred_knn,y_test)
mat
```

Out[58]:

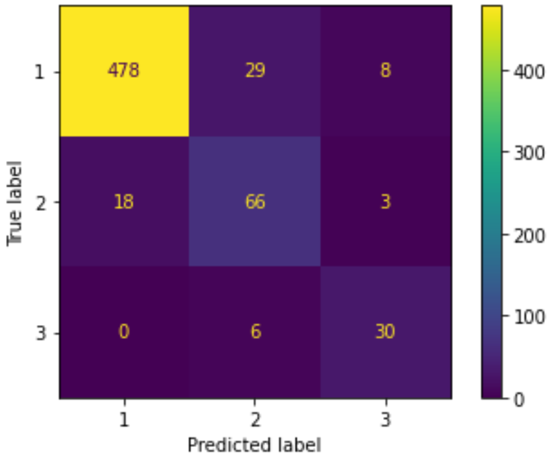
```
array([[478,  29,   8],
       [ 18,  66,   3],
       [  0,   6,  30]])
```

In [59]:

```
label=['1','2','3']
cmd=ConfusionMatrixDisplay(mat,display_labels=label)
cmd.plot()
```

Out[59]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f47533
d1fa0>
```



In [60]:

```
score_knn=accuracy_score(y_pred_knn,y_test)
score_knn
```

Out[60]:

```
0.8996865203761756
```

```
In [61]: report=classification_report(y_test,y_pred_knn)
         print(report)
```

```
              precision    recall  f1-score   support

         1.0       0.93      0.96      0.95       496
         2.0       0.76      0.65      0.70       101
         3.0       0.83      0.73      0.78        41

    accuracy                           0.90       638
   macro avg       0.84      0.78      0.81       638
weighted avg       0.90      0.90      0.90       638
```

2.NAIVE_BAYES

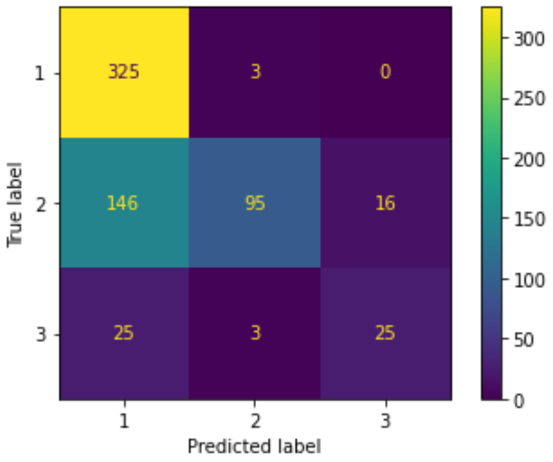
```
In [62]: # ML ALGORITHM
         from sklearn.naive_bayes import GaussianNB
         model_nb=GaussianNB()
         model_nb.fit(x_train,y_train)
         y_pred_nb=model_nb.predict(x_test)
```

```
In [63]: # PERFORMANCE EVALUATION
         mat_nb=confusion_matrix(y_pred_nb,y_test)
         mat_nb
```

```
Out[63]: array([[325,   3,   0],
                [146,  95,  16],
                [ 25,   3,  25]])
```

```
In [64]: cmd_nb=ConfusionMatrixDisplay(mat_nb,display_labels=label)
         cmd_nb.plot()
```

```
Out[64]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f474d5
         8e310>
```



```
In [65]: score_nb=accuracy_score(y_pred_nb,y_test)
         score_nb
```

```
Out[65]: 0.6974921630094044
```

In [66]:
```
report_nb=classification_report(y_test,y_pred_nb)
print(report_nb)
```

```
              precision    recall  f1-score   support

         1.0       0.99      0.66      0.79       496
         2.0       0.37      0.94      0.53       101
         3.0       0.47      0.61      0.53        41

    accuracy                           0.70       638
   macro avg       0.61      0.74      0.62       638
weighted avg       0.86      0.70      0.73       638
```

3.SVM

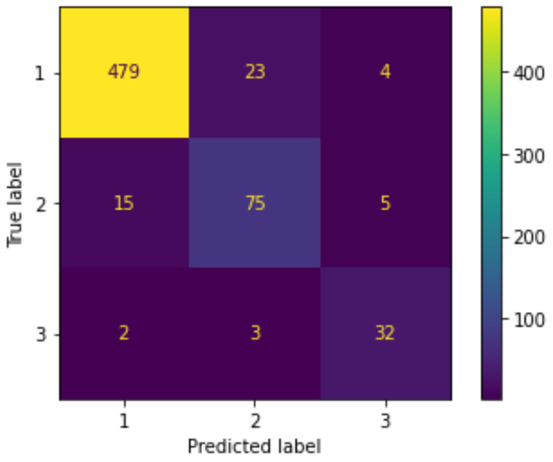
In [67]:
```
# ML ALGORITHM
from sklearn.svm import SVC
model_svm=SVC()
model_svm.fit(x_train,y_train)
y_pred_svm=model_svm.predict(x_test)
```

In [68]:
```
# PERFORMANCE EVALUATION
mat_svm=confusion_matrix(y_pred_svm,y_test)
mat_svm
```

Out[68]:
```
array([[479,  23,   4],
       [ 15,  75,   5],
       [  2,   3,  32]])
```

In [69]:
```
cmd_svm=ConfusionMatrixDisplay(mat_svm,display_labels=label)
cmd_svm.plot()
```

Out[69]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f474d5
8ed90>
```



In [70]:
```
score_svm=accuracy_score(y_pred_svm,y_test)
score_svm
```

Out[70]:
```
0.9184952978056427
```

```
In [71]: report_svm=classification_report(y_test,y_pred_svm)
         print(report_svm)
```

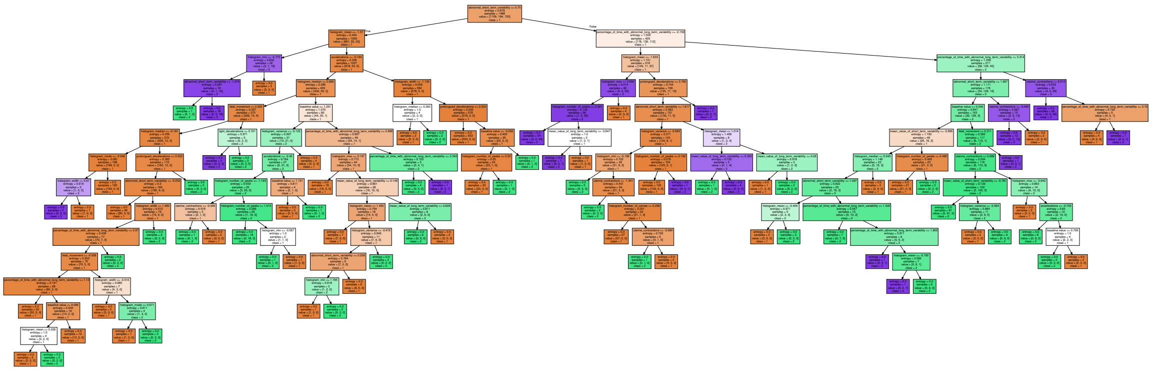
```
              precision    recall  f1-score   support

         1.0       0.95      0.97      0.96       496
         2.0       0.79      0.74      0.77       101
         3.0       0.86      0.78      0.82        41

    accuracy                           0.92       638
   macro avg       0.87      0.83      0.85       638
weighted avg       0.92      0.92      0.92       638
```

4.DECISION TREE

```
In [72]: # ML ALGORITHM
         from sklearn.tree import DecisionTreeClassifier
         model_tree=DecisionTreeClassifier(criterion='entropy')
         model_tree.fit(x_train,y_train)
         y_pred_tree=model_tree.predict(x_test)
```

```
In [75]: # PLOT DECISION TREE
         import graphviz
         from sklearn import tree
         dot_data=tree.export_graphviz(model_tree,out_file=None,\
                                       feature_names=['baseline value', 'acceleratio
         ns', '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'],\
                class_names=['1','2','3'],\
                filled=True)
         graph = graphviz.Source(dot_data, format="png")
         graph
```

Out[75]:

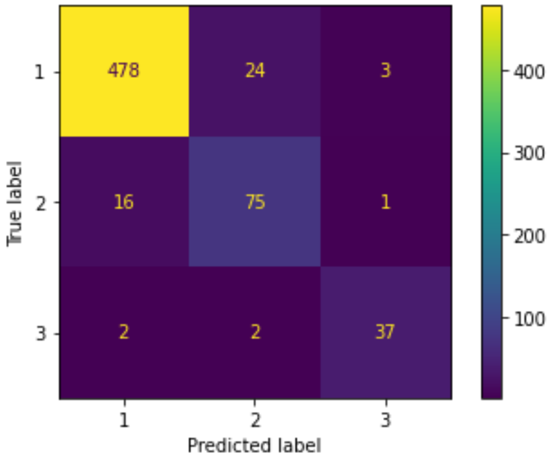


```
In [76]: # PERFORMANCE EVALUATION
         mat_tree=confusion_matrix(y_pred_tree,y_test)
         mat_tree
```

```
Out[76]: array([[478,  24,   3],
                [ 16,  75,   1],
                [  2,   2,  37]])
```

```
In [77]: cmd_tree=ConfusionMatrixDisplay(mat_tree,display_labels=label)
         cmd_tree.plot()
```

```
Out[77]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f474d5
         1c4c0>
```



```
In [78]: score_tree=accuracy_score(y_pred_tree,y_test)
         score_tree
```

```
Out[78]: 0.9247648902821317
```

```
In [79]: report_tree=classification_report(y_test,y_pred_tree)
         print(report_tree)
```

```
              precision    recall  f1-score   support

         1.0       0.95      0.96      0.96       496
         2.0       0.82      0.74      0.78       101
         3.0       0.90      0.90      0.90        41

    accuracy                           0.92       638
   macro avg       0.89      0.87      0.88       638
weighted avg       0.92      0.92      0.92       638
```

5.RANDOM FOREST

```
In [80]: # ML ALGORITHM
         from sklearn.ensemble import RandomForestClassifier
         model_random=RandomForestClassifier(n_estimators=5,criterion='entropy')
         model_random.fit(x_train,y_train)
         y_pred_random=model_random.predict(x_test)
```

```
In [81]: # PERFORMANCE EVALUATION
         mat_random=confusion_matrix(y_pred_random,y_test)
         mat_random
```

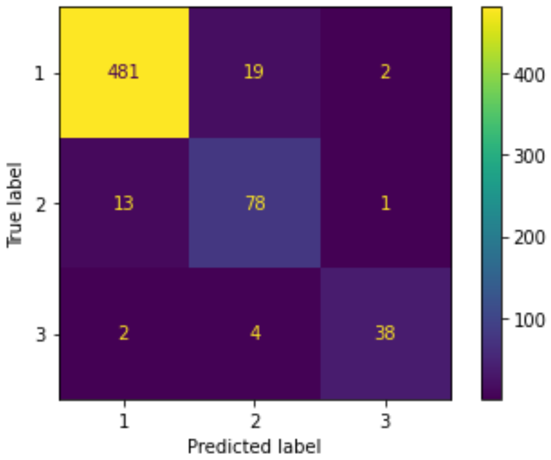
```
Out[81]: array([[481,  19,   2],
                [ 13,  78,   1],
                [  2,   4,  38]])
```

In [82]:

```
cmd_random=ConfusionMatrixDisplay(mat_random,display_labels=label)
cmd_random.plot()
```

Out[82]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f474df
a2f40>
```



In [83]:

```
score_random=accuracy_score(y_pred_random,y_test)
score_random
```

Out[83]:

```
0.9357366771159875
```

In [84]:

```
report_random=classification_report(y_test,y_pred_random)
print(report_random)
```

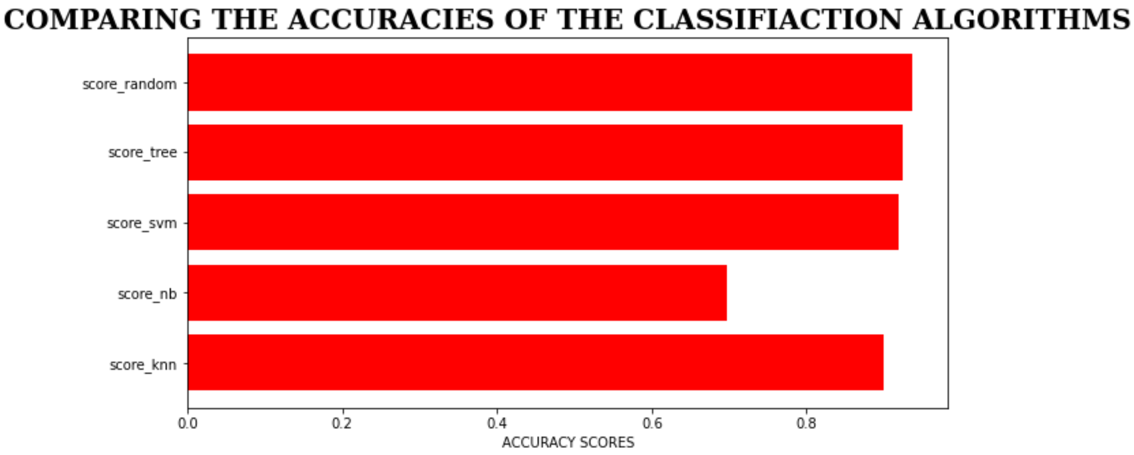
```
              precision    recall  f1-score   support

         1.0       0.96      0.97      0.96       496
         2.0       0.85      0.77      0.81       101
         3.0       0.86      0.93      0.89        41

    accuracy                           0.94       638
   macro avg       0.89      0.89      0.89       638
weighted avg       0.93      0.94      0.93       638
```

In [85]:

```
x_lst=["score_knn","score_nb","score_svm","score_tree","score_random"]
y_lst=[score_knn,score_nb,score_svm,score_tree,score_random]
```

In [95]:

```
# COMPARE ACCURACIES
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.barh(x_lst,y_lst,color='red')
font={'family':'serif','color':'black','size':20,'fontweight':'bold'}
plt.title("COMPARING THE ACCURACIES OF THE CLASSIFIACTION ALGORITHMS",fontd
ict=font)
plt.xlabel("ACCURACY SCORES")
plt.show()
```


COMPARING THE ACCURACIES OF THE CLASSIFIACTION ALGORITHMS
score_random
score_tree
score_svm
score_nb
score_knn
ACCURACY SCORES