Fetal Health Data Set-Model Tuning

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_style('whitegrid')
```

In [2]:

```
fetal=pd.read_csv(r'C:\Users\ADMIN\Desktop\Projects\Data sets\fetal_health.csv')
fetal.head()
```

Out[2]:

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

5 rows × 22 columns

In [3]:

```
fetal.shape
```

Out[3]:

```
(2126, 22)
```

In [4]:

```
fetal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   baseline value                                          2126 non-null   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]:

```python
print(fetal.fetal_health.value_counts())
fetal['fetal_health'].value_counts(normalize=True)
```

```
1.0    1655
2.0     295
3.0     176
Name: fetal_health, dtype: int64
```

Out[5]:

```
1.0    0.778457
2.0    0.138758
3.0    0.082785
Name: fetal_health, dtype: float64
```

In [6]:

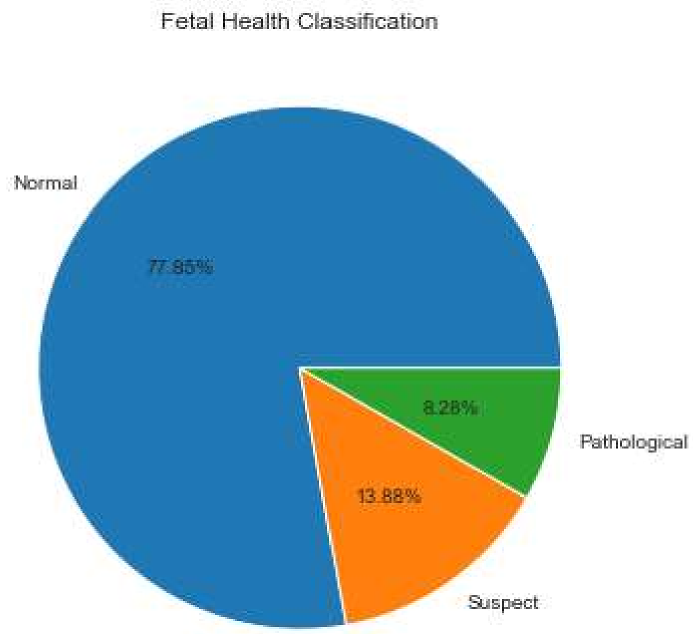
```python
fetal.isnull().sum()
```

Out[6]:

```
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
```

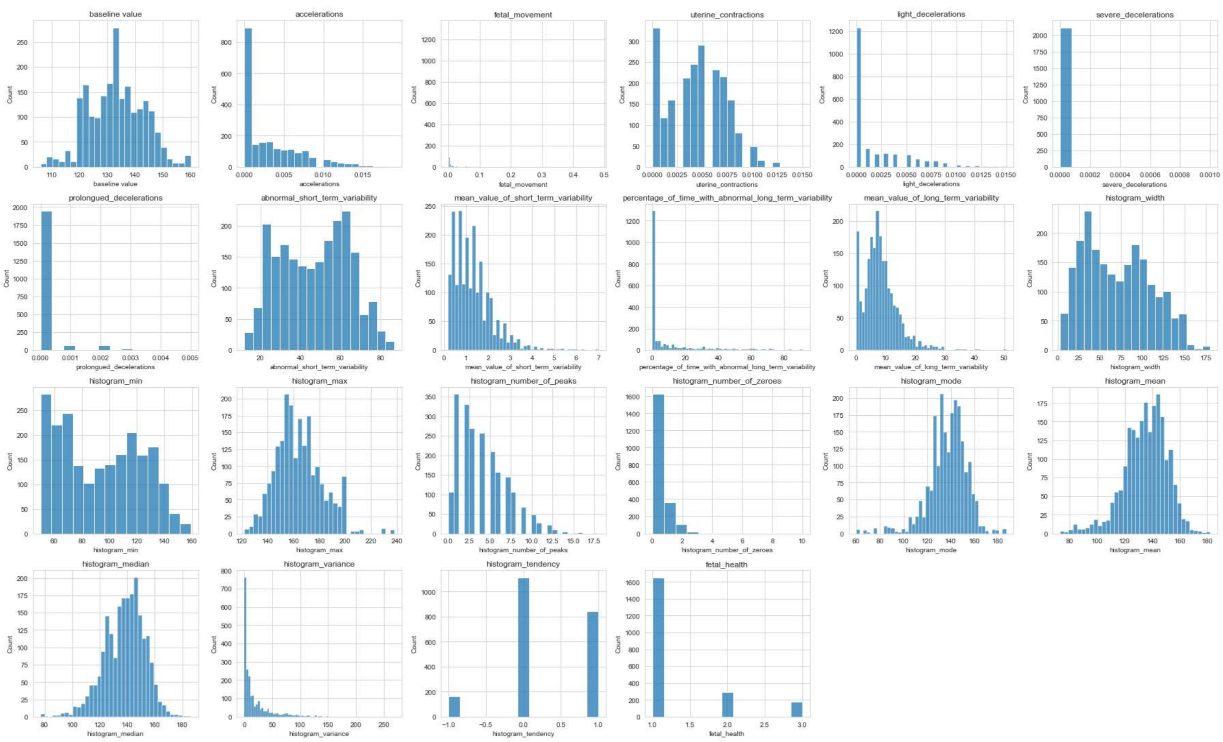
In [7]:

```python
plt.figure(figsize=(14,6))
plt.pie(
    fetal.fetal_health.value_counts(),
    autopct='%.2f%%',
    labels=['Normal','Suspect','Pathological'],
)
plt.title('Fetal Health Classification')
plt.show()
```



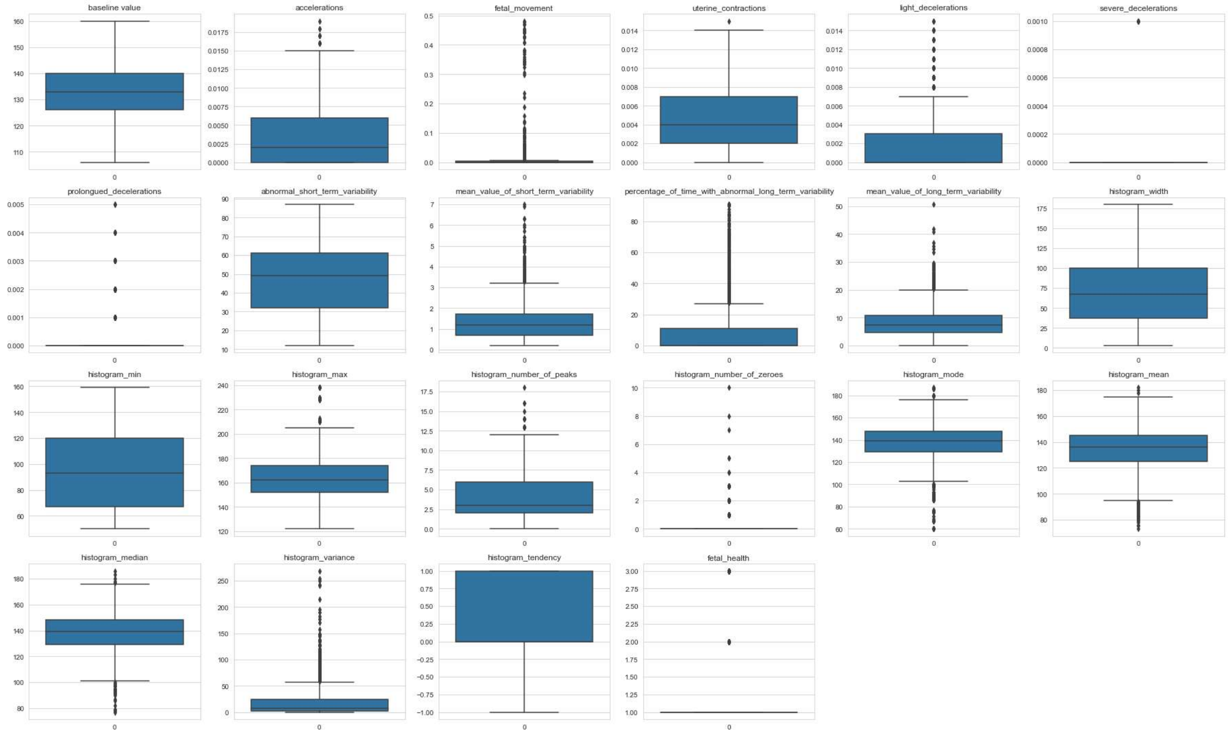
In [8]:

```
plt.figure(figsize=(25,15))
for i,col in enumerate(fetal.columns):
    plt.subplot(4,6,i+1)
    sns.histplot(data=fetal[col])
    plt.title(col)
plt.tight_layout()
plt.show()
```



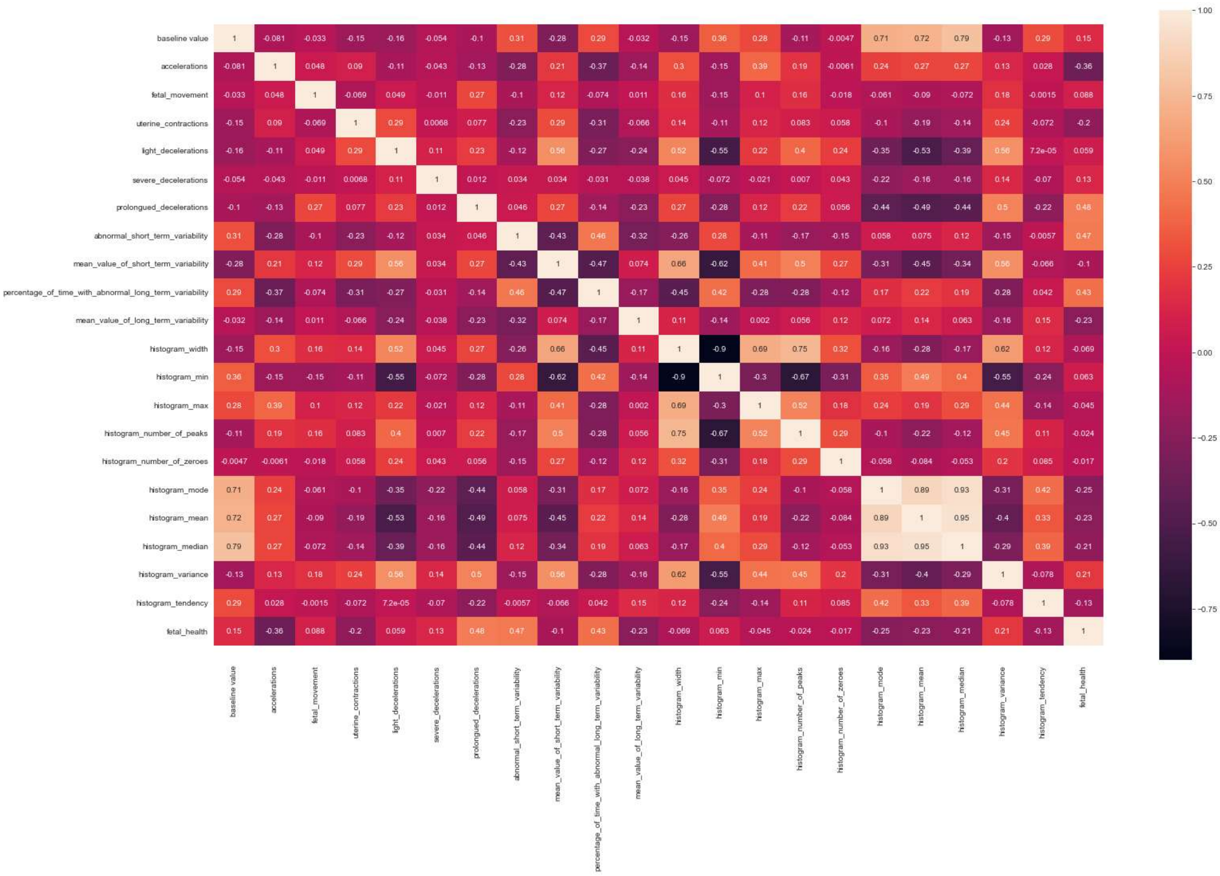
In [9]:

```
plt.figure(figsize=(25,15))
for i,col in enumerate(fetal.columns):
    plt.subplot(4,6,i+1)
    sns.boxplot(data=fetal[col])
    plt.title(col)
plt.tight_layout()
plt.show()
```



In [10]:

```
plt.figure(figsize=(25,15))
cor = fetal.corr()
ax = sns.heatmap(cor,annot=True)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
```



In [11]:

```
cor=fetal.select_dtypes(exclude="object").corr()
Num_feature = cor["fetal_health"].sort_values(ascending=False).head(20).to_frame()

cm = sns.light_palette("#5F9EA0", as_cmap=True)
```

```
style = Num_feature.style.background_gradient(cmap=cm)
style
```

Out[11]:

	fetal_health
fetal_health	1.000000
prolongued_decelerations	0.484859
abnormal_short_term_variability	0.471191
percentage_of_time_with_abnormal_long_term_variability	0.426146
histogram_variance	0.206630
baseline value	0.148151
severe_decelerations	0.131934
fetal_movement	0.088010
histogram_min	0.063175
light_decelerations	0.058870
histogram_number_of_zeroes	-0.016682
histogram_number_of_peaks	-0.023666
histogram_max	-0.045265
histogram_width	-0.068789
mean_value_of_short_term_variability	-0.103382
histogram_tendency	-0.131976
uterine_contractions	-0.204894
histogram_median	-0.205033
mean_value_of_long_term_variability	-0.226797
histogram_mean	-0.226985

We can see three features: "prolongued_decelerations", "abnormal_short_term_variability", "percentage_of_time_with_abnormal_long_term_variability" have high correlation with the target culumn (fetal_health).

In [12]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import mean_squared_error
```

In [13]:

```
X = fetal.iloc[:,:-1]
y = fetal.iloc[:,-1]
```

In [14]:

```
scale = StandardScaler()
sc = scale.fit_transform(X)
```

```
X = pd.DataFrame(sc,columns=fetal.iloc[:,:-1].columns)
X.head()
```

Out[14]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelera
0	-1.352220	-0.822388	-0.20321	-1.482465	-0.638438	-0.0
1	-0.132526	0.730133	-0.20321	0.554627	0.375243	-0.0
2	-0.030884	-0.046128	-0.20321	1.233657	0.375243	-0.0
3	0.070757	-0.046128	-0.20321	1.233657	0.375243	-0.0
4	-0.132526	0.988886	-0.20321	1.233657	-0.638438	-0.0

5 rows × 21 columns

In [15]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_st
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[15]:

```
((1488, 21), (638, 21), (1488,), (638,))
```

In [16]:

```
knn = KNeighborsClassifier()
knn_mod = knn.fit(X_train, y_train)
print(f"Baseline K-Nearest Neighbors: {round(knn_mod.score(X_test, y_test), 3)}")
pred_knn = knn_mod.predict(X_test)
```

```
Baseline K-Nearest Neighbors: 0.876
```

Hyper Parameter Tuning

In [17]:

```
# Cross validate K-Nearest Neighbors model
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score

cv_method = StratifiedKFold(n_splits=3,
                            random_state=42,shuffle=True
                            )

scores_knn = cross_val_score(knn, X_train, y_train, cv = cv_method, n_jobs = 2, scor

print(f"Scores(Cross validate) for K-Nearest Neighbors model:\n{scores_knn}")
print(f"CrossValMeans: {round(scores_knn.mean(), 3)}")
print(f"CrossValStandard Deviation: {round(scores_knn.std(), 3)}")
```

```
Scores(Cross validate) for K-Nearest Neighbors model:
[0.8891129  0.90322581 0.90322581]
CrossValMeans: 0.899
CrossValStandard Deviation: 0.007
```

In [18]:

```
params_knn = {"leaf_size": list(range(1,30)),
              "n_neighbors": list(range(1,21)),
              "p": [1,2]}

GridSearchCV_knn = GridSearchCV(estimator=KNeighborsClassifier(),
                                param_grid=params_knn,
```

```
                                cv=cv_method,
                                verbose=1,
                                n_jobs=-1,
                                scoring="accuracy",
                                return_train_score=True
                                )
```

In [19]:

```
# Fit model with train data
GridSearchCV_knn.fit(X_train, y_train);
```

```
Fitting 3 folds for each of 1160 candidates, totalling 3480 fits
```

In [28]:

```
best_estimator_knn = GridSearchCV_knn.best_estimator_
print(f"Best estimator for KNN model:\n{best_estimator_knn}")
best_params_knn = GridSearchCV_knn.best_params_
print(f"Best parameter values:\n{best_params_knn}")
```

```
Best estimator for KNN model:
KNeighborsClassifier(leaf_size=1, n_neighbors=3)
Best parameter values:
{'leaf_size': 1, 'n_neighbors': 3, 'p': 2}
```

In [21]:

```
best_score_knn = GridSearchCV_knn.best_score_
print(f"Best score for GNB model: {round(best_score_knn, 3)}")
```

```
Best score for GNB model: 0.904
```

In [22]:

```
# Test with new parameter for KNN model
knn = KNeighborsClassifier(leaf_size=1, n_neighbors=3 , p=1)
knn_mod = knn.fit(X_train, y_train)
pred_knn = knn_mod.predict(X_test)

mse_knn = mean_squared_error(y_test, pred_knn)
rmse_knn = np.sqrt(mean_squared_error(y_test, pred_knn))
score_knn_train = knn_mod.score(X_train, y_train)
score_knn_test = knn_mod.score(X_test, y_test)
```

In [23]:

```
print(f"Mean Square Error for K_Nearest Neighbor  = {round(mse_knn, 3)}")
print(f"Root Mean Square Error for K_Nearest Neighbor = {round(rmse_knn, 3)}")
print(f"R^2(coefficient of determination) on training set = {round(score_knn_train,
print(f"R^2(coefficient of determination) on testing set = {round(score_knn_test, 3)
```

```
Mean Square Error for K_Nearest Neighbor  = 0.132
Root Mean Square Error for K_Nearest Neighbor = 0.363
R^2(coefficient of determination) on training set = 0.956
R^2(coefficient of determination) on testing set = 0.897
```

In [24]:

```
print("Classification Report")
print(classification_report(y_test, pred_knn))
```

```
Classification Report
              precision    recall  f1-score   support

         1.0       0.94      0.96      0.95       497
         2.0       0.66      0.66      0.66        88
         3.0       0.88      0.68      0.77        53

    accuracy                           0.90       638
```

```
   macro avg       0.83      0.77      0.79       638
weighted avg       0.90      0.90      0.89       638
```

In [25]:

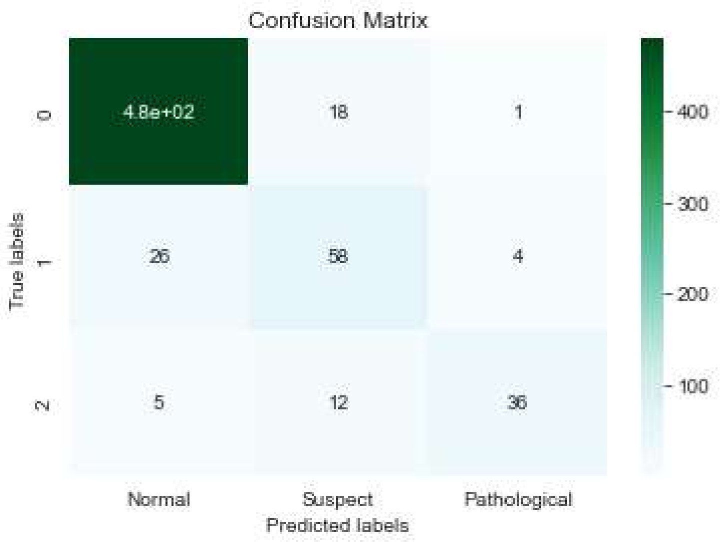
```python
print("Confusion Matrix:")
print(confusion_matrix(y_test, pred_knn))
```

```
Confusion Matrix:
[[478  18   1]
 [ 26  58   4]
 [  5  12  36]]
```

In [26]:

```python
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, pred_knn), annot=True, ax = ax, cmap = "BuGn");

# labels, title and ticks
ax.set_xlabel("Predicted labels");
ax.set_ylabel("True labels");
ax.set_title("Confusion Matrix");
ax.xaxis.set_ticklabels(["Normal", "Suspect", "Pathological"]);
```



In [27]:

```python
from sklearn.metrics import accuracy_score
print('Accuracy Before hyper parameter tuning:',accuracy_score(y_test,pred_knn))
print('Accuracy After hyper parameter tuning:',GridSearchCV_knn .best_score_)
```

```
Accuracy Before hyper parameter tuning: 0.896551724137931
Accuracy After hyper parameter tuning: 0.9038978494623656
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```