Jenna Kobular

DSSA-5302-401

Dr. Baldwin

August 3, 2023

Predicting Fetal Health Using Cardiotocograms

Executive Summary

The reduction of child mortality is considered a key indicator of human progress. Fetal death refers to the intrauterine death of a fetus during pregnancy. Newborn death accounts for 47% of all child deaths under the age of 5. Fetal health is monitored prior to and during labor by using cardiotocography. A cardiotocogram measures fetal heart rate and uterine contractions. The results have generally required a specialist to review them and assess if further action needs to be taken. This study looks to see if machine learning can help accurately predict fetal health. Using the results from 2126 records, it was found that machine learning can be an asset to maternal-fetal health. After implementing five different machine learning algorithms, a random forest classifier was able to predict fetal health with 94.6% accuracy. While machine learning will not replace the need for human review, it can help reduce medical errors and increase cost-effectiveness.

Introduction

Artificial Intelligence (AI) has the potential to make the healthcare system much more efficient and cost-effective. AI and machine learning can also fill holes in access to medical care in poorer nations and reduce medical errors that cost people their lives. In recent years, more algorithms and machine learning models are being introduced into the medical field. One field of medicine that has begun to bring machine learning into the equation is maternal-fetal medicine.

Fetal health can be predicted by using machine learning algorithms. Looking specifically at evaluating cardiotocographs - which is something that previously required an obstetrician's review.

Fetal death refers to the intrauterine death of a fetus at any time during a pregnancy. Deaths that occur later in pregnancy are sometimes referred to as stillbirths. Reduction of child mortality is one of the Sustainable Development Goals of the United Nations and is a key indicator of human progress. There are around 6,700 newborn deaths every day, which amounts to 47% of all child deaths under the age of 5 (World Health Organization, 2021). Cardiotocograms (CTGs) are an effective and inexpensive way for healthcare providers to monitor fetal health. A cardiotocography visually represents fetal heart rate (FHR) and uterine contractions (Pettker and Campbell, 2018). Fetal heart rate is recognized as an important indicator of fetal health. Pregnancies can be complicated if the mother has a medical condition such as diabetes or high blood pressure, which can impact the health and development of the fetus. Cardiotocographs aim to help identify situations where potential complications may occur and then effectively intervene to improve outcomes.

A carditocograph is a continuous electronic record of the fetal heart rate which is obtained via an ultrasound transducer placed on the mother's abdomen (Grivell, Alfirevic, Gyte, and Devane, 2015). Cardiotocography is widely accepted and used in maternity care in the stages of antepartum (before labor) and intrapartum (during labor). If the CTG shows that the fetus is not responding well, it may mean the baby is not getting enough oxygen or that the placenta is not working as it should (Bellani, 2022). If after further testing it is decided the baby is not doing well, the doctor may induce labor or perform a C-section.

When a CTG is performed, generally an obstetrician, a doctor who specializes in pregnancy, childbirth, and a woman's reproductive system, reviews the result and classifies the risk. Whenever humans are involved, there is a risk of error or oversight. Introducing machine learning algorithms may be able to adequately identify suspect and pathological results. Using a dataset comprised of records from 2126 cardiotocograms, which were classified by expert obstetricians into three classes: normal, suspect, and pathological, this paper looks to see if machine learning algorithms can correctly identify cardiotocograms that indicate suspect and pathological results. Machine learning can add value to the medical field and result in better health outcomes for mothers and their babies.

Literature Review

Studies have been carried out in recent years to predict certain risks that may occur during pregnancy and labor. Artificial intelligence has also been used to predict the most suitable birth method using the characteristics of mothers (Islam, Mustafina, Mahmud, and Khan, 2022). The most frequently used machine learning algorithms are neural networks, decision trees, and random forest models. These algorithms have been used to predict the risk of premature birth, predict factors associated with premature birth, and predict the most suitable delivery method.

The most common maternal complications are infection, preeclampsia, gestational diabetes, and prolonged or pre-term labor (Islam et al., 2022). Cardiotocograms consist of fetal heart rate and uterine contractions which are continuously recorded during labor (Ogasawara et al., 2021). Looking at acceleration, baseline heart rate, and heart rate variability fetal status can be diagnosed. Ogasawara et al., used deep neural networks to predict infant outcomes (2021). They found that their model could predict infant outcomes from the last 30 minutes of CTG before delivery. Islam et al. (2023), used seven different machine-learning models to classify the fetal state, including K Nearest Neighbors, a decision tree, a support vector machine, and a random forest. Each of these models was evaluated with an accuracy of 90% or greater.

The use of machine learning in pregnancy diseases and complications has only increased in recent years. The application's goal is not only to diagnose but also to manage, treat, and understand different perinatal alterations (Mennickent et al., 2023). It is expected that in upcoming years, the use of ML will continue to grow in obstetrics and gynecology. Based on the literature, it is feasible to predict fetal health using cardiotocography results.

Methodology

Using data sourced from Kaggle, several machine-learning algorithms were used to predict fetal health outcomes. The dataset from Kaggle originated from a study published in The Journal of Maternal-Fetal Medicine. SisPorto is a program that is used for the automated analysis of cardiotocograms and closely follows the International Federation of Gynecology and Obstetrics (FIGO) guidelines (Ayres-de-campos, et al., 1999). The dataset used in this study was used to evaluate the efficiency of SisPorto. Ayres de Campos et al. (2000) made the dataset they used for analysis openly available. It contains 2126 records of CTG features and classifies them into three fetal health states: normal, suspect, and pathological. The data had no null values and was pre-cleaned so limited pre-processing needed to be performed. The three classes of the target variable, fetal health, were imbalanced, with a majority being 1 (or normal). There were 21 features that were extracted from the CTG exams. The features included the baseline fetal heart rate, accelerations, fetal movement, uterine contractions, decelerations, and heart rate variability.

The features were scaled using scikit-learn’s standard scaler to normalize the data. There were a few outliers that were not removed due to domain relevance. The outliers may have represented genuine and meaningful observations in the context of fetal health. There was not a reasonable enough basis to remove them. After reviewing histograms and summary statistics of the features, it was found that a majority were normally distributed. The dataset was split into training and test sets using 80 percent of the data for training and 20 percent of the data for testing and evaluation.

Using Python and the scikit-learn library, a few multi-class models were built and evaluated. Predictions were generated using several different multi-class models. Two different multinomial logistic regression models were evaluated, as well as a decision tree, K-Nearest-Neighbors, and a random forest classifier. A logistic regression was used since they are a simple and easily interpreted model that works well with multi-class classification problems. K-Nearest-Neighbors are simple and intuitive but do not always perform well with high-dimensional data. Decision trees are another model that is easy to interpret, they are good when using both numerical and categorical data. A random forest model was used since it combines multiple decision trees and can improve the accuracy and robustness of the analysis. Random forests can also handle high-dimensional data with many features with ease. These models were found to be effective in past studies, so it follows that they were chosen to evaluate this dataset.

The random forest model performed the best with a weight average accuracy of 94%. In all but one of the two logistic regression models, all 21 features were used. There is a chance the model has a bias towards over-fitting that was not mitigated. A limitation of the study is the sci-kit learn library does not offer much in the way of mitigating the risk of overfitting. Another limitation is the size of the dataset, there were only 2126 records that were analyzed. The effectiveness of the models may be altered with datasets of larger size and scope.

Data

After analyzing the data using the pandas and numpy libraries in Python, it was found that the target class was imbalanced. The majority of the cardiotocography results were normal. This can be shown in Figure 1 below.

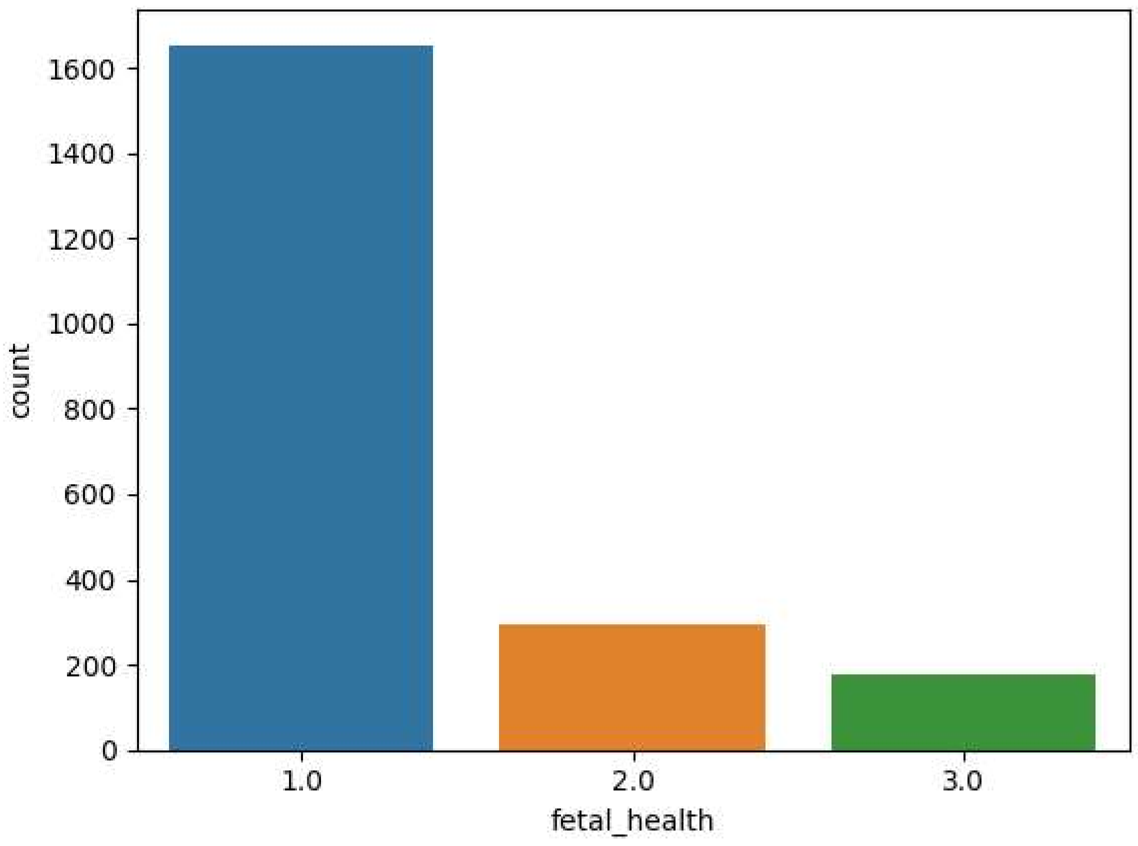


Figure 1

With imbalanced classes, logistic regressions tend to struggle to properly classify data. To see if the features were evenly distributed, summary statistics were reviewed. In addition, histograms for each feature were reviewed which is shown in Figure 2. Attributes were largely normally distributed or mildly skewed. A correlation matrix was also reviewed to look for multicollinearity. There were several features such as that showed a strong correlation. Prolonged decelerations, abnormal short-term variability, and percentage of time with abnormal long-term variability showed were highly correlated with fetal health. The dataset was largely pre-cleaned and did not have any null values. All features had a datatype of float64, which works well with machine learning algorithms. The data was used with integrity, the dataset was made available after being used in a study. The data was compliant with the California Consumer Privacy Act (CCPA) and was analyzed appropriately. With most of the features being normally distributed, after scaling the features, all were used to make predictions.

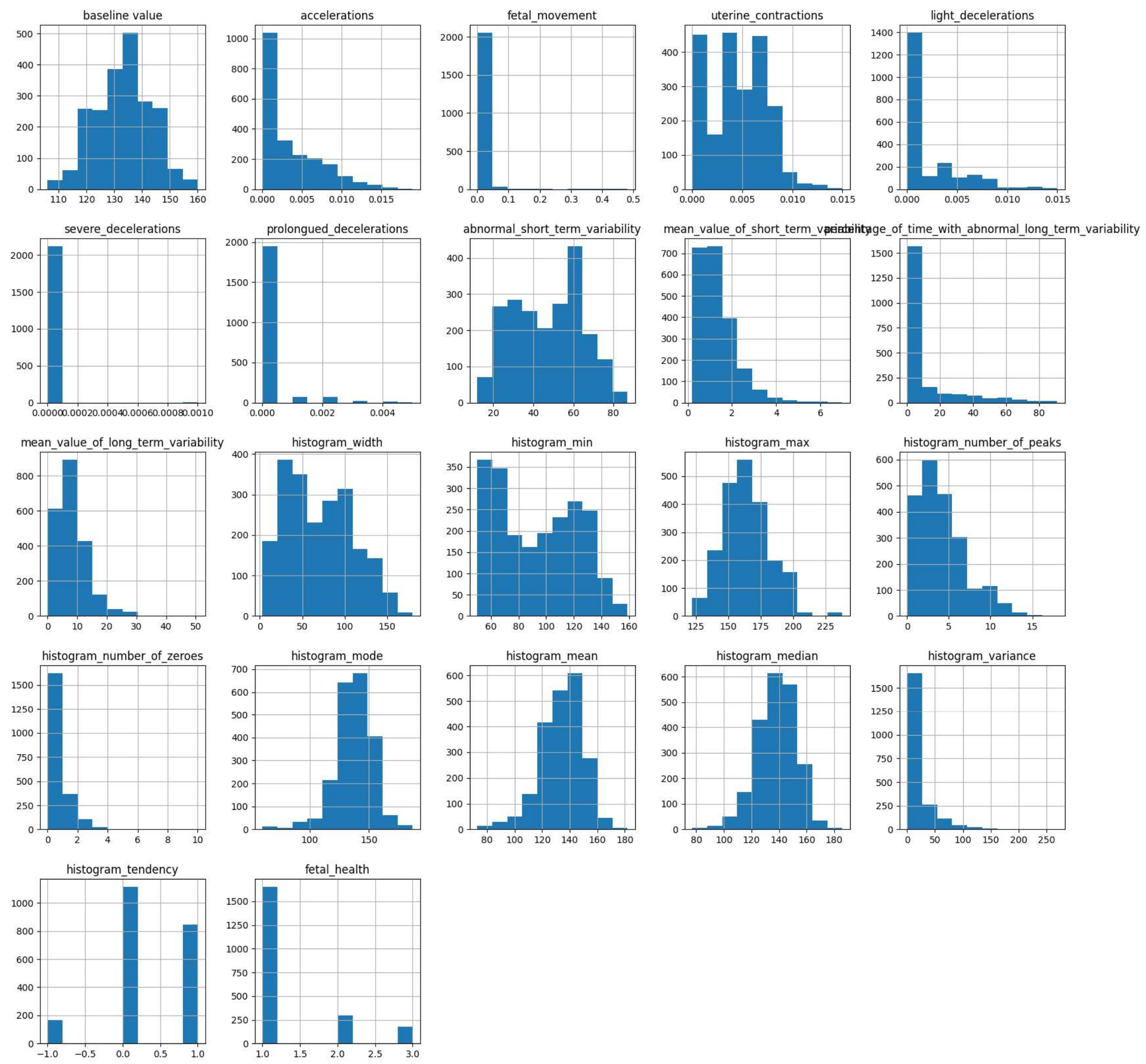


Figure 2

All models had high accuracy at over 85% but it is important to note that performance differs across classes. Class 1, which represents normal fetal health, has the highest precision, recall, F1-score, and accuracy. Each model performed well in classifying when a fetus' health was normal. Table 1 below shows an overview of the performance of each model that was built, using the weighted average between the classes for precision, recall, and F1-score.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Logistic Regression (all features)	87.8%	88%	88%	88%
Decision Tree	93.2%	92.9%	93.1%	92.9%
Logistic Regression (11 features)	86.6%	87%	75%	74%
KNN	92%	92%	85%	86%
Random Forest	94.6%	94%	95%	94%

Table 1

As is shown in Table 1, the random forest and decision tree outperformed the logistic regression and K-Nearest-Neighbors. This could be because logistic regression and KNN models are more simplistic and do not always perform well with many features. Random forests are able to combine multiple decision trees which reduces the risk of overfitting and can improve accuracy and robustness. Figure 3 below shows the confusion matrix for the random forest model.

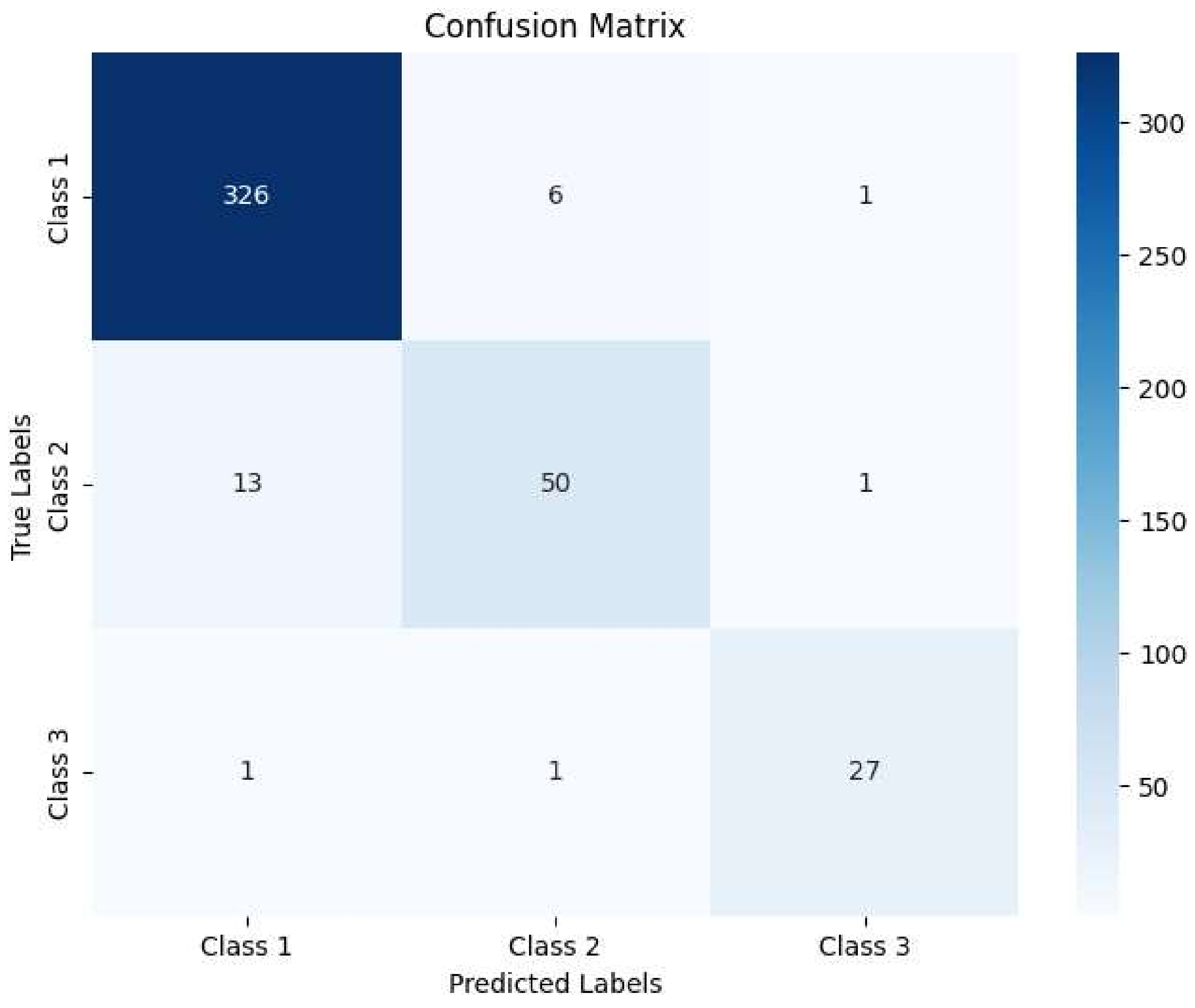


Figure 3

The confusion matrix shows that the random forest was able to successfully predict most of the truly normal records. There were six false positives, which means that either suspect or pathological cases were incorrectly classified as being normal. If the model was used alone, this could result in medical errors or even death. There was one false negative which means that the model inaccurately classified a normal result as being suspect or pathological. Ideally, since dealing with medical data, the model should have higher levels of sensitivity. For best medical outcomes, more false negatives are preferred to having false positives.

For class 2, which indicates suspect fetal health, the model had low precision and recall. The random forest, as well as the other models implemented, struggled to distinguish this class from normal and pathological results. Class 3, pathological fetal health, was balanced. The model correctly identified nearly all true positives and only had one false positive and one false negative. Model performance moving forward should focus on increasing sensitivity so that suspect cases are more likely to be correctly identified.

Conclusion

AI and machine learning are starting to be used in the medical field. In maternal-fetal medicine, it is being used to predict fetal health, predict the best labor method, and predict the chance of a c-section being performed. A cardiotocography looks at fetal heart rate and uterine contractions, obstetricians use cardiotocograms to monitor fetal health. There are currently around 6,700 newborn deaths a day – some due to medical errors or lack of intervention. Specialists or obstetricians have generally done the analysis of CTGs. The introduction of machine learning into the field can improve medical outcomes and result in a reduction in fetal deaths. In this study, the random forest model performed best coming in with 94.6% accuracy. The model performed best on normal fetal health records and did well with class 3 or pathological fetal health results. The random forest model could be improved as it did not handle class 2 or suspect fetal health results with high accuracy. The model sensitivity could be increased so that suspect cases are more likely to be correctly identified. It is preferred that the model have more false negatives than positives. Inaccurately predicting that a fetal health result is normal could have severe implications that could increase the risk of medical errors. Nothing about fetal health makes sense except in the light of cardiotography.

This study had limitations: only 2126 fetal health records were evaluated, and the target variable was imbalanced. This may have caused the models to have some bias towards class 1. Overall, the random forest classifier was able to predict fetal health with 94.6% accuracy. The model was not perfect, and neither are humans. Moving forward, the medical field can introduce machine learning to predict fetal health. A comprehensive approach should include both obstetrician review and machine learning. Incorporating machine learning into the medical field can increase access to care in a time of labor shortages, decrease costs, and improve health outcomes. Future work can look to increase the sensitivity of the random forest model. As datasets increase in size, the models may need to be tweaked to reduce bias and the risk of overfitting. All in all, machine learning can be a valuable addition to medical treatment.

References

Ayres de Campos et al. (2000) SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms. J Matern Fetal Med 5:311-318

Bellani, P. (2022, December 1). What is cardiotocography (CTG) and why do I need it? BabyCenter India. Retrieved July 31, 2023, from https://www.babycenter.in/x1045384/what-is-cardiotocography-ctg-and-why-do-i-need-it

CDC. (2022, October 18). NVSS - Fetal Deaths. CDC. Retrieved July 31, 2023, from https://www.cdc.gov/nchs/nvss/fetal_death.htm

Diogo Ayres-de-campos, João Bernardes, Antonio Garrido, Joaquim Marques-de-sá & Luis Pereira-leite (2000) SisPorto 2.0: A Program for Automated Analysis of Cardiotocograms, Journal of Maternal-Fetal Medicine, 9:5, 311-318, DOI: 10.3109/14767050009053454

Islam, Md & Rokunojjaman, Md & Amin, Al & Akhtar, Md & Sarker, Iqbal. (2023). Diagnosis and Classification of Fetal Health Based on CTG Data Using Machine Learning Techniques. 10.1007/978-3-031-34622-4_1.

Islam, M.M., Rokunojjaman, M., Amin, A., Akhtar, M.N., Sarker, I.H. (2023). Diagnosis and Classification of Fetal Health Based on CTG Data Using Machine Learning Techniques. In: Satu, M.S., Moni, M.A., Kaiser, M.S., Arefin, M.S. (eds) Machine Intelligence and Emerging Technologies. MIET 2022. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 491. Springer, Cham. https://doi.org/10.1007/978-3-031-34622-4_1

Islam, M.N., Mustafina, S.N., Mahmud, T. et al. Machine learning to predict pregnancy outcomes: a systematic review, synthesizing framework and future research agenda. BMC Pregnancy Childbirth 22, 348 (2022). https://doi.org/10.1186/s12884-022-04594-2

Mennickent, D., Rodríguez, A., Opazo, M. C., Riedel, C. A., Castro, E., Eriz-Salinas, A., Appel-Rubio, J., Aguayo, C., Damiano, A. E., Guzmán-Gutiérrez, E., & Araya, J. (2023). Machine learning applied in maternal and fetal health: a narrative review focused on pregnancy diseases and complications. Frontiers in endocrinology, 14, 1130139. https://doi.org/10.3389/fendo.2023.1130139

Powell, A. (2020, November 11). Risks and benefits of an AI revolution in medicine. Harvard Gazette. Retrieved July 31, 2023, from https://news.harvard.edu/gazette/story/2020/11/risks-and-benefits-of-an-ai-revolution-in-medicine/

World Health Organization (WHO). (2023, February 23). A woman dies every two minutes due to pregnancy or childbirth: UN agencies. World Health Organization (WHO). Retrieved July 31, 2023, from https://www.who.int/news/item/23-02-2023-a-woman-dies-every-two-minutes-due-to-pregnancy-or-childbirth--un-agencies

Appendix A

Code for analyzing, visualizing, and making predictions.

```
#data processing
import pandas as pd
import numpy as np
#visualization
import matplotlib.pyplot as plt
import seaborn as sns
# Algorithms
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

In [ ]:

```
#reading in data
df = pd.read_csv("C:/Users/17326/OneDrive - go.Stockton.edu/Documents/dssa/fetal_health.csv")
```

In [ ]:

```
#overview of data
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 [ ]:

```
#summary statistics
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
baseline value	2126.0	133.303857	9.840844	106.0	126.000	133.000	140.000	160.000
accelerations	2126.0	0.003178	0.003866	0.0	0.000	0.002	0.006	0.019
fetal_movement	2126.0	0.009481	0.046666	0.0	0.000	0.000	0.003	0.481
uterine_contractions	2126.0	0.004366	0.002946	0.0	0.002	0.004	0.007	0.015
light_decelerations	2126.0	0.001889	0.002960	0.0	0.000	0.000	0.003	0.015
severe_decelerations	2126.0	0.000003	0.000057	0.0	0.000	0.000	0.000	0.001
prolongued_decelerations	2126.0	0.000159	0.000590	0.0	0.000	0.000	0.000	0.005
abnormal_short_term_variability	2126.0	46.990122	17.192814	12.0	32.000	49.000	61.000	87.000
mean_value_of_short_term_variability	2126.0	1.332785	0.883241	0.2	0.700	1.200	1.700	7.000
percentage_of_time_with_abnormal_long_term_variability	2126.0	9.846660	18.396880	0.0	0.000	0.000	11.000	91.000
mean_value_of_long_term_variability	2126.0	8.187629	5.628247	0.0	4.600	7.400	10.800	50.700
histogram_width	2126.0	70.445908	38.955693	3.0	37.000	67.500	100.000	180.000

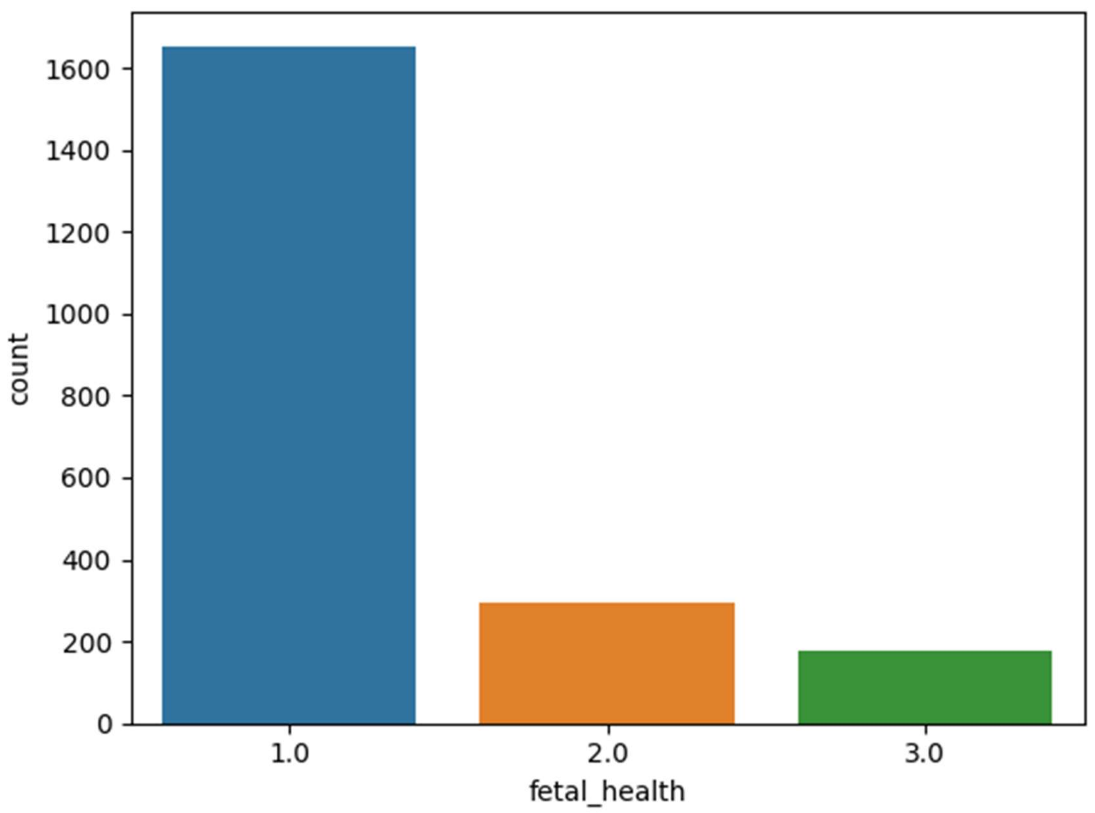
	count	mean	std	min	25%	50%	75%	max
histogram_min	2126.0	93.579492	29.560212	50.0	67.000	93.000	120.000	159.000
histogram_max	2126.0	164.025400	17.944183	122.0	152.000	162.000	174.000	238.000
histogram_number_of_peaks	2126.0	4.068203	2.949386	0.0	2.000	3.000	6.000	18.000
histogram_number_of_zeroes	2126.0	0.323612	0.706059	0.0	0.000	0.000	0.000	10.000
histogram_mode	2126.0	137.452023	16.381289	60.0	129.000	139.000	148.000	187.000
histogram_mean	2126.0	134.610536	15.593596	73.0	125.000	136.000	145.000	182.000
histogram_median	2126.0	138.090310	14.466589	77.0	129.000	139.000	148.000	186.000
histogram_variance	2126.0	18.808090	28.977636	0.0	2.000	7.000	24.000	269.000
histogram_tendency	2126.0	0.320320	0.610829	-1.0	0.000	0.000	1.000	1.000
fetal_health	2126.0	1.304327	0.614377	1.0	1.000	1.000	1.000	3.000

In [ ]:

```
#looking to see if target value (fetal health) is imbalanced
sns.countplot(data = df, x = "fetal_health")
```

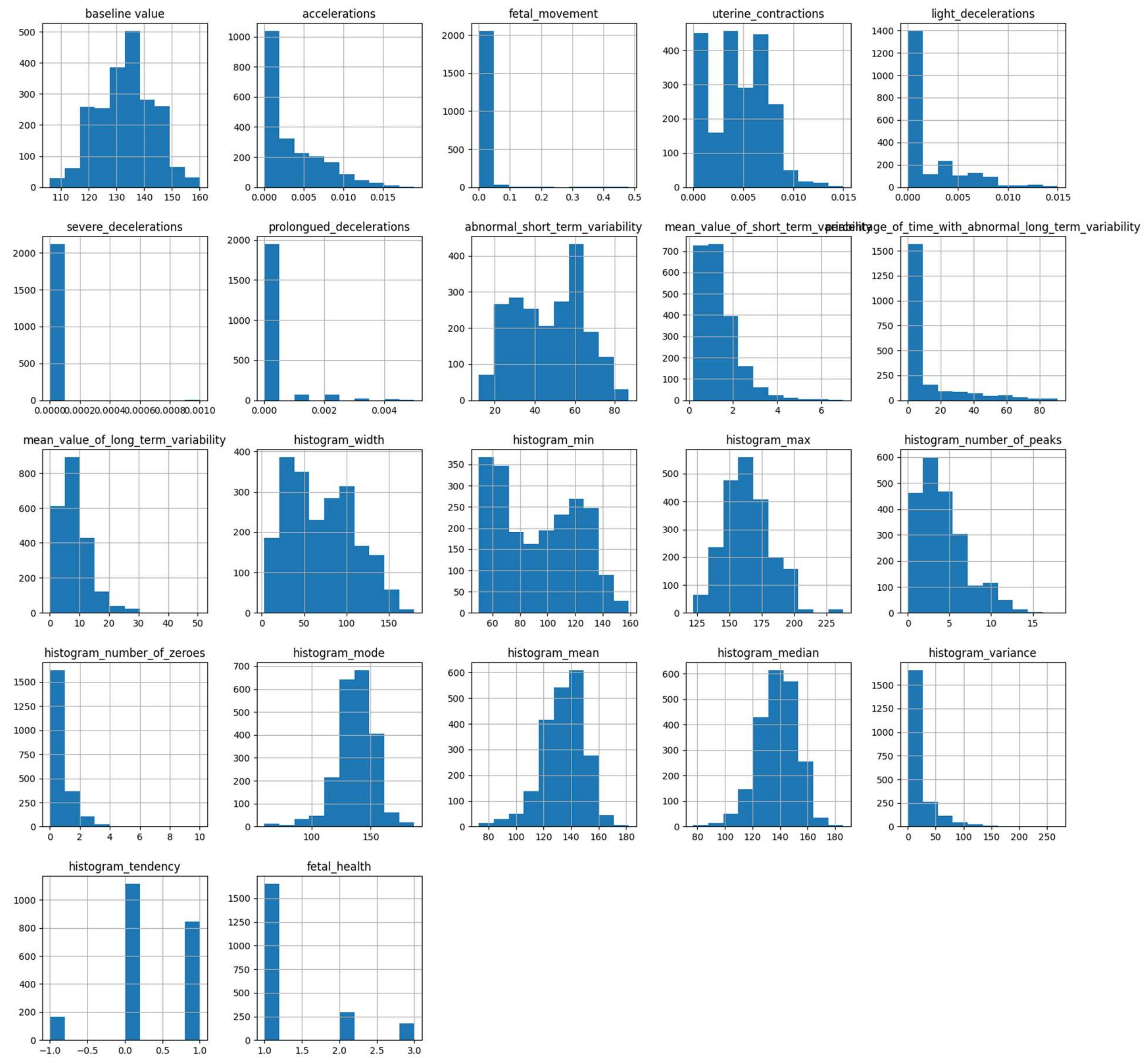
Out[ ]:

```
<AxesSubplot: xlabel='fetal_health', ylabel='count'>
```



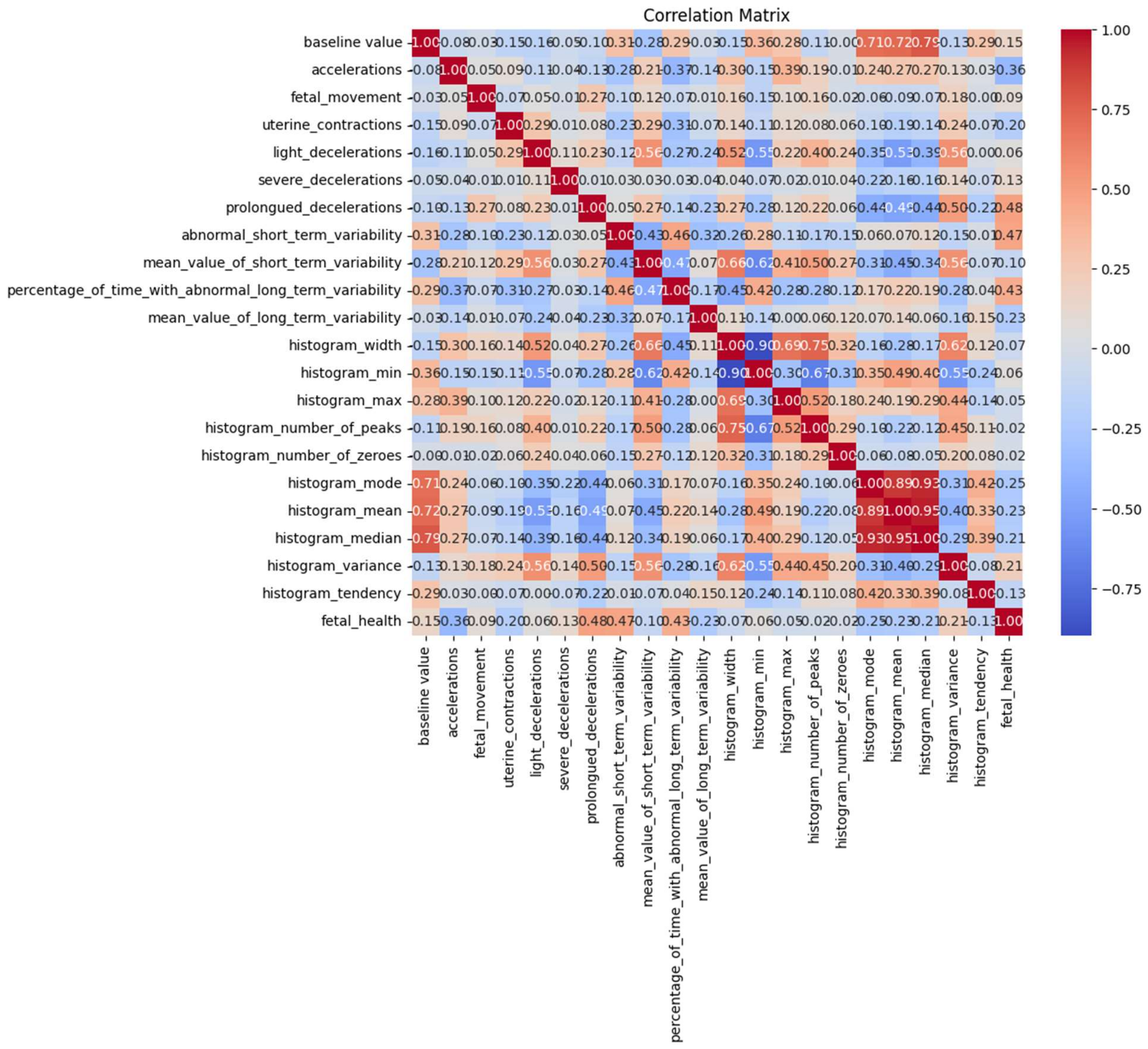
In [ ]:

```
#looking to see how all features are distributed
hist_plot = df.hist(figsize= (20,20))
```



In [ ]:

```
#correlation matrix
corr_matrix = df.corr()

plt.figure(figsize=(10,8))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title("Correlation Matrix")
plt.show()
```



In [ ]:

```
#defining features and target variable
X = df.drop(["fetal_health"], axis= 1)
y = df["fetal_health"]
```

In [ ]:

```
#Stanadrdizing the data
col_names = list(X.columns)
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(X)
X_Scaled = pd.DataFrame(X_Scaled, columns = col_names)
```

In [ ]:

```
X_Scaled.describe().T
```

Out[ ]:

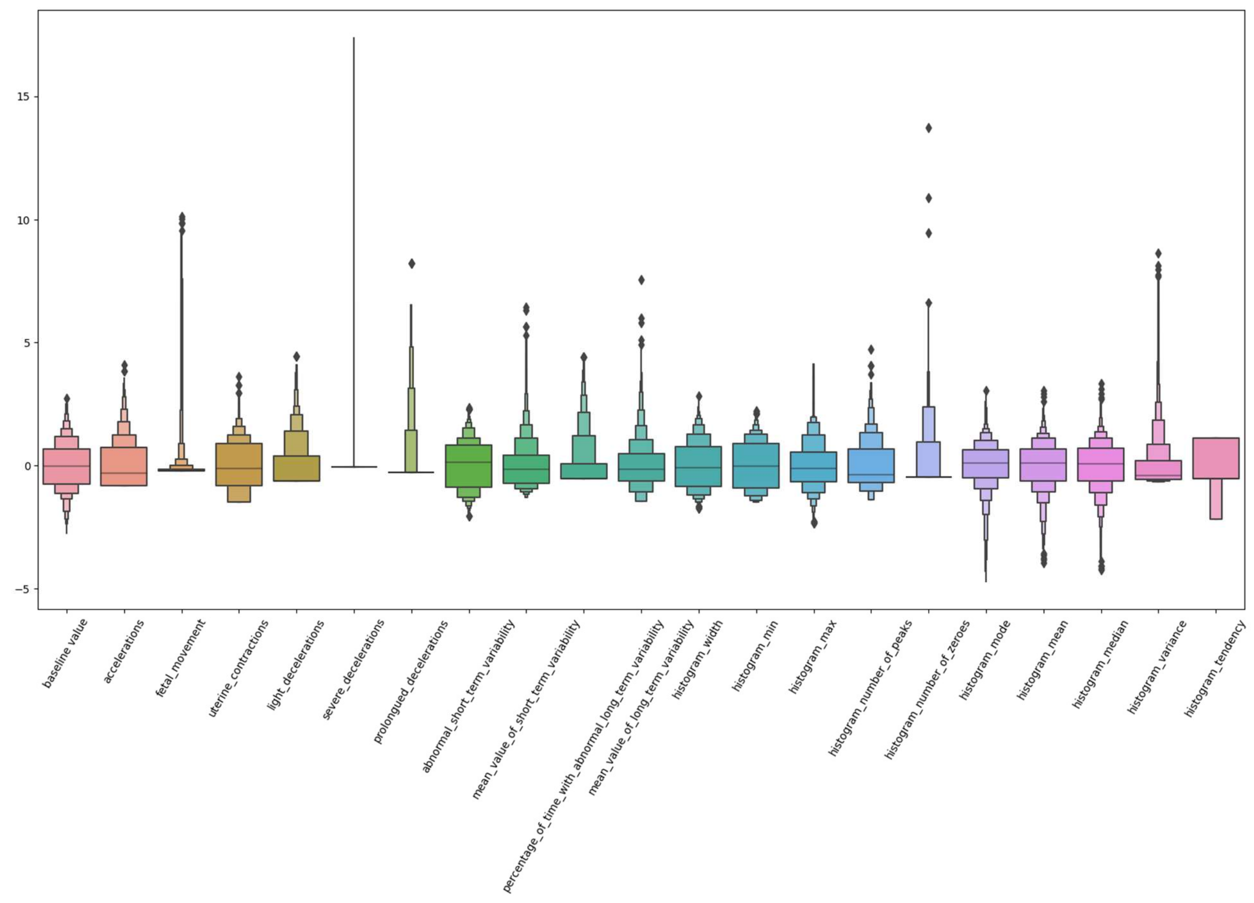
	count	mean	std	min	25%	50%	75%	max
baseline value	2126.0	1.069490e-15	1.000235	-2.775197	-0.742373	-0.030884	0.680604	2.713428
accelerations	2126.0	-4.010589e-17	1.000235	-0.822388	-0.822388	-0.304881	0.730133	4.093929
fetal_movement	2126.0	-1.336863e-17	1.000235	-0.203210	-0.203210	-0.203210	-0.138908	10.106540
uterine_contractions	2126.0	-1.336863e-16	1.000235	-1.482465	-0.803434	-0.124404	0.894142	3.610264
light_decelerations	2126.0	-5.347452e-17	1.000235	-0.638438	-0.638438	-0.638438	0.375243	4.429965
severe_decelerations	2126.0	6.684315e-18	1.000235	-0.057476	-0.057476	-0.057476	-0.057476	17.398686
prolongued_decelerations	2126.0	1.336863e-17	1.000235	-0.268754	-0.268754	-0.268754	-0.268754	8.208570
abnormal_short_term_variability	2126.0	-7.352747e-17	1.000235	-2.035639	-0.872088	0.116930	0.815060	2.327675

	count	mean	std	min	25%	50%	75%	max
mean_value_of_short_term_variability	2126.0	6.684315e-17	1.000235	-1.282833	-0.716603	-0.150373	0.415857	6.417893
percentage_of_time_with_abnormal_long_term_variability	2126.0	-5.347452e-17	1.000235	-0.535361	-0.535361	-0.535361	0.062707	4.412293
mean_value_of_long_term_variability	2126.0	2.406354e-16	1.000235	-1.455081	-0.637583	-0.139975	0.464263	7.555172
histogram_width	2126.0	-3.007942e-17	1.000235	-1.731757	-0.858765	-0.075640	0.758838	2.812936
histogram_min	2126.0	-4.679021e-17	1.000235	-1.474609	-0.899376	-0.019608	0.893996	2.213648
histogram_max	2126.0	-1.203177e-16	1.000235	-2.342558	-0.670314	-0.112899	0.555999	4.123453
histogram_number_of_peaks	2126.0	-1.671079e-16	1.000235	-1.379664	-0.701397	-0.362263	0.655137	4.724738
histogram_number_of_zeroes	2126.0	2.757280e-17	1.000235	-0.458444	-0.458444	-0.458444	-0.458444	13.708003

	count	mean	std	min	25%	50%	75%	max
histogram_mode	2126.0	1.069490e-16	1.000235	-4.729191	-0.516077	0.094519	0.644055	3.025381
histogram_mean	2126.0	-6.684315e-16	1.000235	-3.951945	-0.616458	0.089126	0.666422	3.039749
histogram_median	2126.0	2.673726e-16	1.000235	-4.223849	-0.628514	0.062897	0.685166	3.312527
histogram_variance	2126.0	-5.347452e-17	1.000235	-0.649208	-0.580173	-0.407586	0.179212	8.635997
histogram_tendency	2126.0	-1.069490e-16	1.000235	-2.162031	-0.524526	-0.524526	1.112980	1.112980

In [ ]:

```
#plotting standardized features
plt.figure(figsize=(20,10))
sns.boxenplot(data = X_Scaled)
plt.xticks(rotation=60)
plt.show()
```



In [ ]:

```
#splitting into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y, test_size =0.2, random_state=42) #80/20 training test split
```

In [ ]:

```
#creating a logistic regression model & fitting the training data
logreg_model = LogisticRegression(max_iter=1000, multi_class= 'multinomial')
logreg_model.fit(X_train, y_train)
```

Out[ ]:

```
LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

In [ ]:

```
#making predictions on the test set
y_pred = logreg_model.predict(X_test)
```

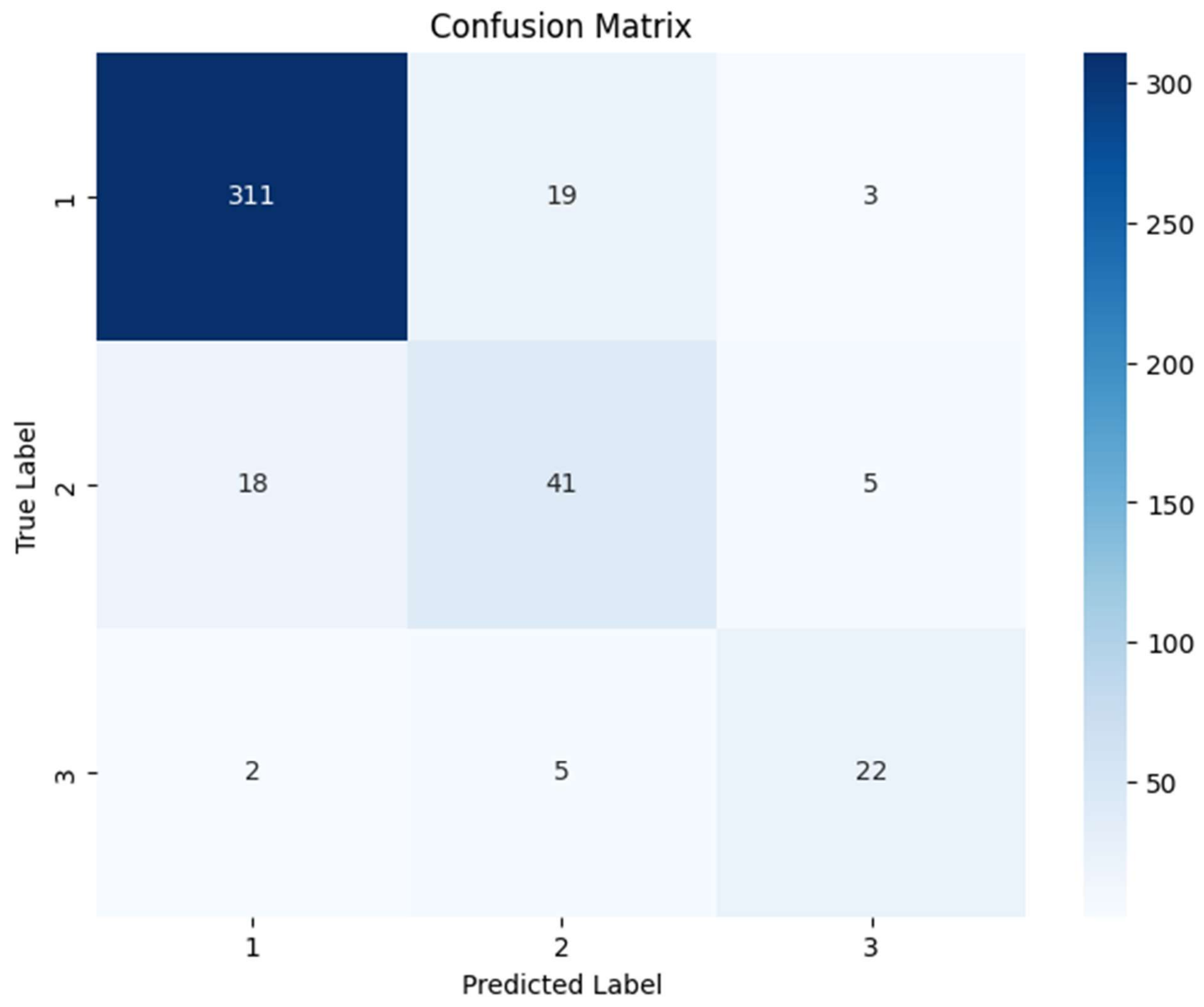
In [ ]:

```
#evaluate model's performance

accuracy = accuracy_score(y_test, y_pred)
classificiation = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

#print results
print("Accuracy: ", accuracy)
print("Classification Report:\n", classificiation)
print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy:  0.8779342723004695
Classification Report:
              precision    recall  f1-score   support

         1.0       0.94      0.93      0.94       333
         2.0       0.63      0.64      0.64        64
         3.0       0.73      0.76      0.75        29

    accuracy                           0.88       426
   macro avg       0.77      0.78      0.77       426
weighted avg       0.88      0.88      0.88       426

Confusion Matrix:
 [[311  19   3]
 [ 18  41   5]
 [  2   5  22]]
```

In [ ]:

```
#visual of log reg confusion matrix

plt.figure(figsize=(8,6))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=[1,2,3], yticklabels=[1,2,3])

plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel('True Label')
plt.show()
```



```
# Algorithms
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
df.drop(df.columns[10:20], axis=1, inplace=True) #dropping histogram values to see how the model is
impacted
```

In [ ]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 12 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  histogram_tendency                                2126 non-null   float64
 11  fetal_health                                      2126 non-null   float64
dtypes: float64(12)
memory usage: 199.4 KB
```

In [ ]:

```
X = df.drop(["fetal_health"], axis= 1) #defining x
y = df["fetal_health"] #defining target variable
```

In [ ]:

```
#Stanadrdizing the data
col_names = list(X.columns)
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(X)
X_Scaled = pd.DataFrame(X_Scaled, columns = col_names)
```

In [ ]:

```
#splitting into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y, test_size =0.2, random_state=42) #80/20 training test split
```

In [ ]:

```
#creating a logistic regression model & fitting the training data
logreg_model = LogisticRegression(max_iter=1000, multi_class= 'multinomial')
logreg_model.fit(X_train, y_train)
```

Out[ ]:

```
LogisticRegression
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

In [ ]:

```
#making predictions on the test set
y_pred = logreg_model.predict(X_test)
```

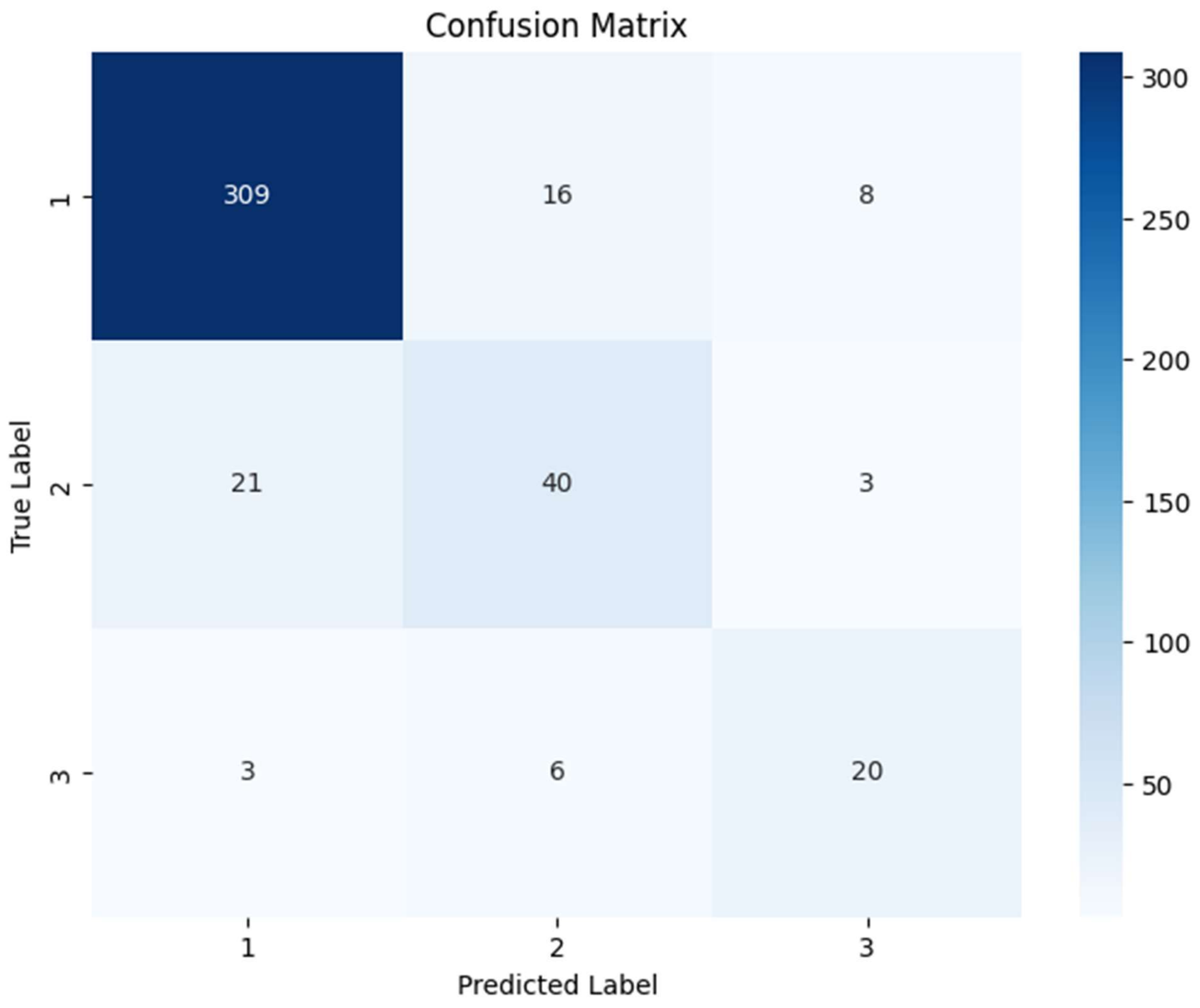
In [ ]:

```
#evaluate model's performance

accuracy = accuracy_score(y_test, y_pred)
classificiation = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

#print results
print("Accuracy: ", accuracy)
print("Classification Report:\n", classificiation)
print("Confusion Matrix:\n", conf_matrix)
Accuracy:  0.8661971830985915
Classification Report:
               precision    recall  f1-score   support

         1.0       0.93      0.93      0.93       333
         2.0       0.65      0.62      0.63        64
         3.0       0.65      0.69      0.67        29

    accuracy                           0.87       426
   macro avg       0.74      0.75      0.74       426
weighted avg       0.87      0.87      0.87       426
```

```
Confusion Matrix:
 [[309  16   8]
 [ 21  40   3]
 [  3   6  20]]
```

In [ ]:

```
#visual of log reg confusion matrix

plt.figure(figsize=(8,6))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=[1,2,3], yticklabels=[1,2,3])

plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel('True Label')
plt.show()
```



```
#models/analysis
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
#defining features and target variable
```

```
X = df.drop(["fetal_health"], axis= 1)
y = df["fetal_health"]
```

In [ ]:

```
#Stanadrdizing the data
col_names = list(X.columns)
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(X)
X_Scaled = pd.DataFrame(X_Scaled, columns = col_names)
```

In [ ]:

```
#train & test
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y, test_size =0.2, random_state=42) #80/20 training test split
```

In [ ]:

```
#create KNN classifier with k=3
knn_model = KNeighborsClassifier(n_neighbors=3)

#fit the model to training data
knn_model.fit(X_train, y_train)
```

Out[ ]:

```
☑ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [ ]:

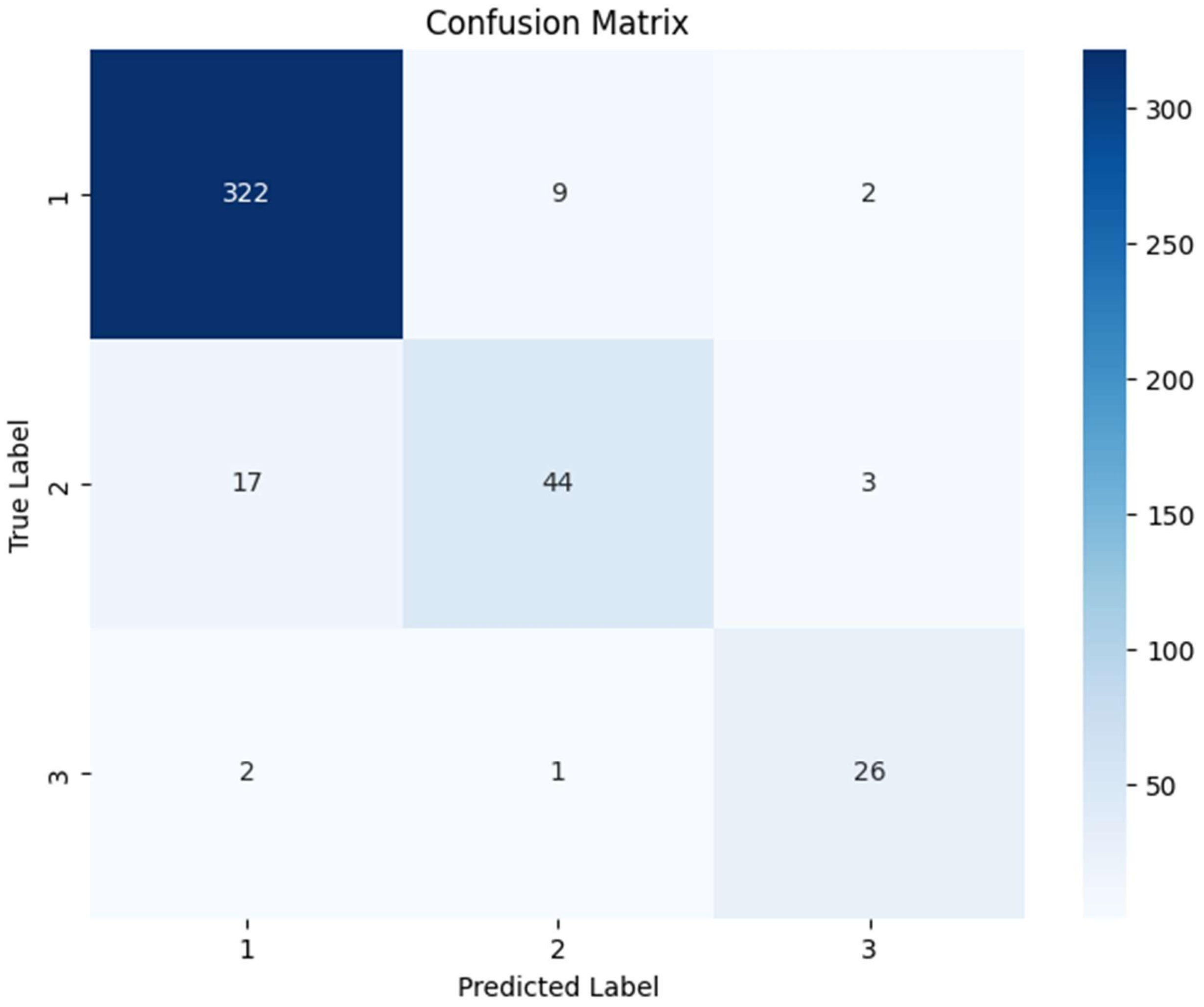
```
# Make predictions on the test set
y_pred = knn_model.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy: 0.92018779342723
Classification Report:
              precision    recall  f1-score   support

         1.0       0.94      0.97      0.96       333
         2.0       0.81      0.69      0.75        64
         3.0       0.84      0.90      0.87        29

    accuracy                           0.92       426
   macro avg       0.87      0.85      0.86       426
weighted avg       0.92      0.92      0.92       426

Confusion Matrix:
 [[322   9   2]
 [ 17  44   3]
 [  2   1  26]]
```

In [ ]:

```
#visual of KNN confusion matrix

plt.figure(figsize=(8,6))
```

```
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=[1,2,3], yticklabels=[1,2,3])

plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel('True Label')
plt.show()
```



```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.feature_selection import RFE
#defining features and target variable
X = df.drop(["fetal_health"], axis= 1)
y = df["fetal_health"]
```

In [ ]:

```
#Stanadrdizing the data
col_names = list(X.columns)
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(X)
X_Scaled = pd.DataFrame(X_Scaled, columns = col_names)
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y, test_size =0.2, random_state=42) #80/20 training
test split
```

In [ ]:

```
#creating random forest classifier and fitting the model to training set

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

rf_model.fit(X_train, y_train)
```

Out[ ]:

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

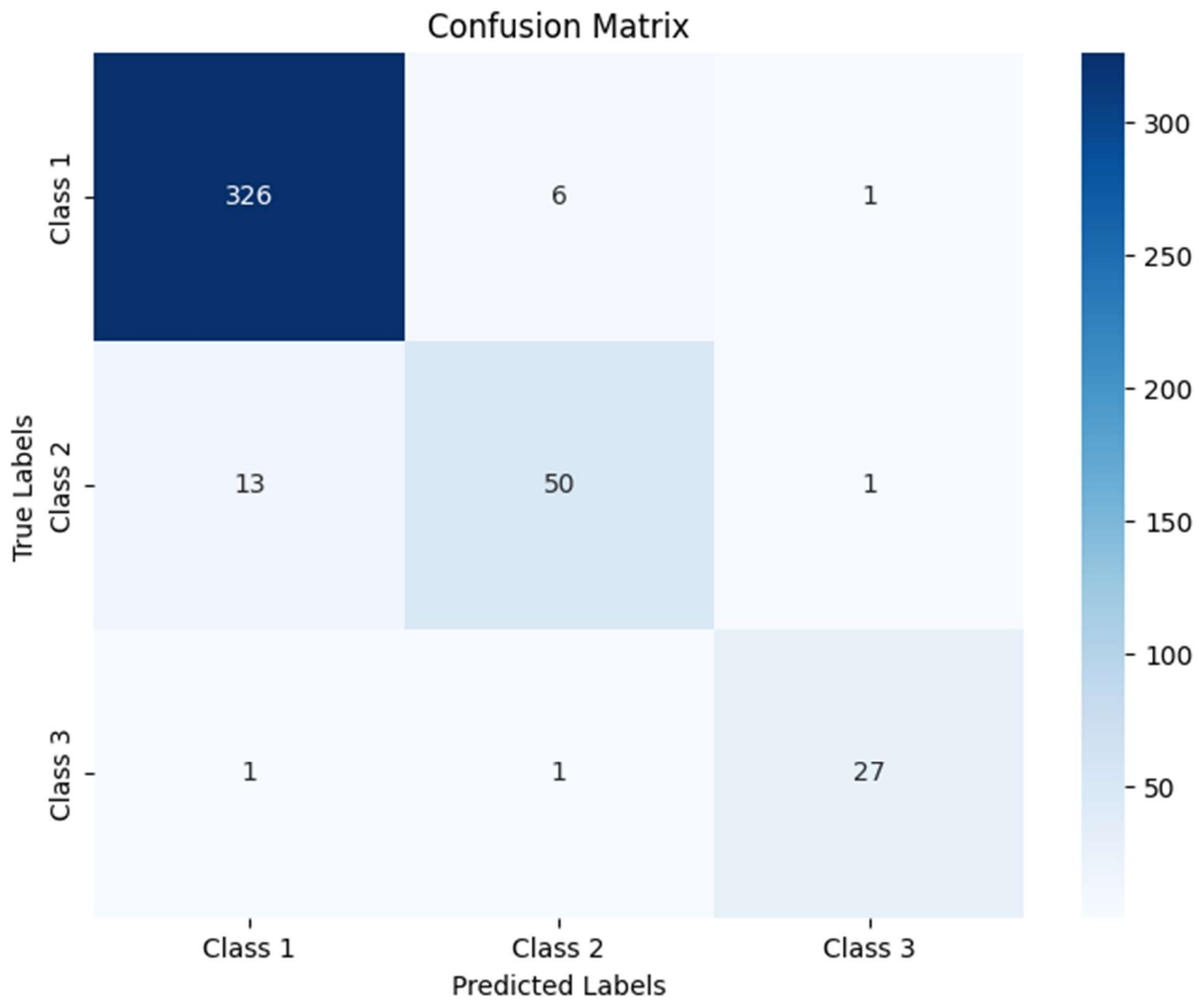
In [ ]:

```
# make predictions on the test set
y_pred = rf_model.predict(X_test)

# evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy: 0.9460093896713615
Classification Report:
              precision    recall  f1-score   support

         1.0       0.96      0.98      0.97       333
         2.0       0.88      0.78      0.83        64
         3.0       0.93      0.93      0.93        29

    accuracy                           0.95       426
   macro avg       0.92      0.90      0.91       426
weighted avg       0.94      0.95      0.94       426

Confusion Matrix:
 [[326   6   1]
 [ 13  50   1]
 [  1   1  27]]
```

In [ ]:

```
# Assuming you have your confusion matrix stored as a numpy array

# Create a list of class labels (replace these with your actual class labels)
class_labels = ['Class 1', 'Class 2', 'Class 3']

# Create the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d', xticklabels=class_labels,
yticklabels=class_labels)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [ ]:

```
#create RFE to rank features
rfe = RFE(estimator=rf_model, n_features_to_select=1)
rfe.fit(X_train,y_train)
```

Out[ ]:

```
RFE
estimator: RandomForestClassifier
RandomForestClassifier
```

In [ ]:

```
ranking_df = pd.DataFrame({'Feature' : X.columns, 'Ranking' : rfe.ranking_})
ranking_df = ranking_df.sort_values(by='Ranking', ascending=True)
print(ranking_df)
```

```
                                   Feature  Ranking
17                          histogram_mean        1
7          abnormal_short_term_variability        2
8       mean_value_of_short_term_variability        3
9  percentage_of_time_with_abnormal_long_term_var...        4
10       mean_value_of_long_term_variability        5
18                        histogram_median        6
```

```
0                               baseline value       7
16                              histogram_mode       8
11                             histogram_width       9
1                                accelerations      10
6                       prolongued_decelerations      11
12                               histogram_min      12
3                          uterine_contractions      13
13                               histogram_max      14
19                          histogram_variance      15
14                     histogram_number_of_peaks      16
2                               fetal_movement      17
4                           light_decelerations      18
20                           histogram_tendency      19
15                    histogram_number_of_zeroes      20
5                          severe_decelerations      21
```

```
#defining features and target variable
X = df.drop(["fetal_health"], axis= 1)
y = df["fetal_health"]
```

In [ ]:

```
#Stanadrdizing the data
col_names = list(X.columns)
scaler = StandardScaler()
X_Scaled = scaler.fit_transform(X)
X_Scaled = pd.DataFrame(X_Scaled, columns = col_names)
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X_Scaled, y, test_size =0.2, random_state=42) #80/20 training test split
```

In [ ]:

```
#create decision tree classifier

dt_model = DecisionTreeClassifier(random_state=42, max_depth=5)

#fit the model w/ train set
dt_model.fit(X_train, y_train)
```

Out[ ]:

```
☑ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=42)
```

In [ ]:

```
y_pred = dt_model.predict(X_test)
```

In [ ]:

```
# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, dt_model.predict_proba(X_test), multi_class='ovr')  # For multiclass classification

# Print the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1)
print("Confusion Matrix:\n", conf_matrix)
print("ROC AUC:", roc_auc)
Accuracy: 0.931924882629108
Precision: 0.9298875578210575
Recall: 0.931924882629108
F1-score: 0.9294106565074547
Confusion Matrix:
 [[324   7   2]
 [ 18  45   1]
 [  1   0  28]]
ROC AUC: 0.9367483589720708
```

In [ ]:

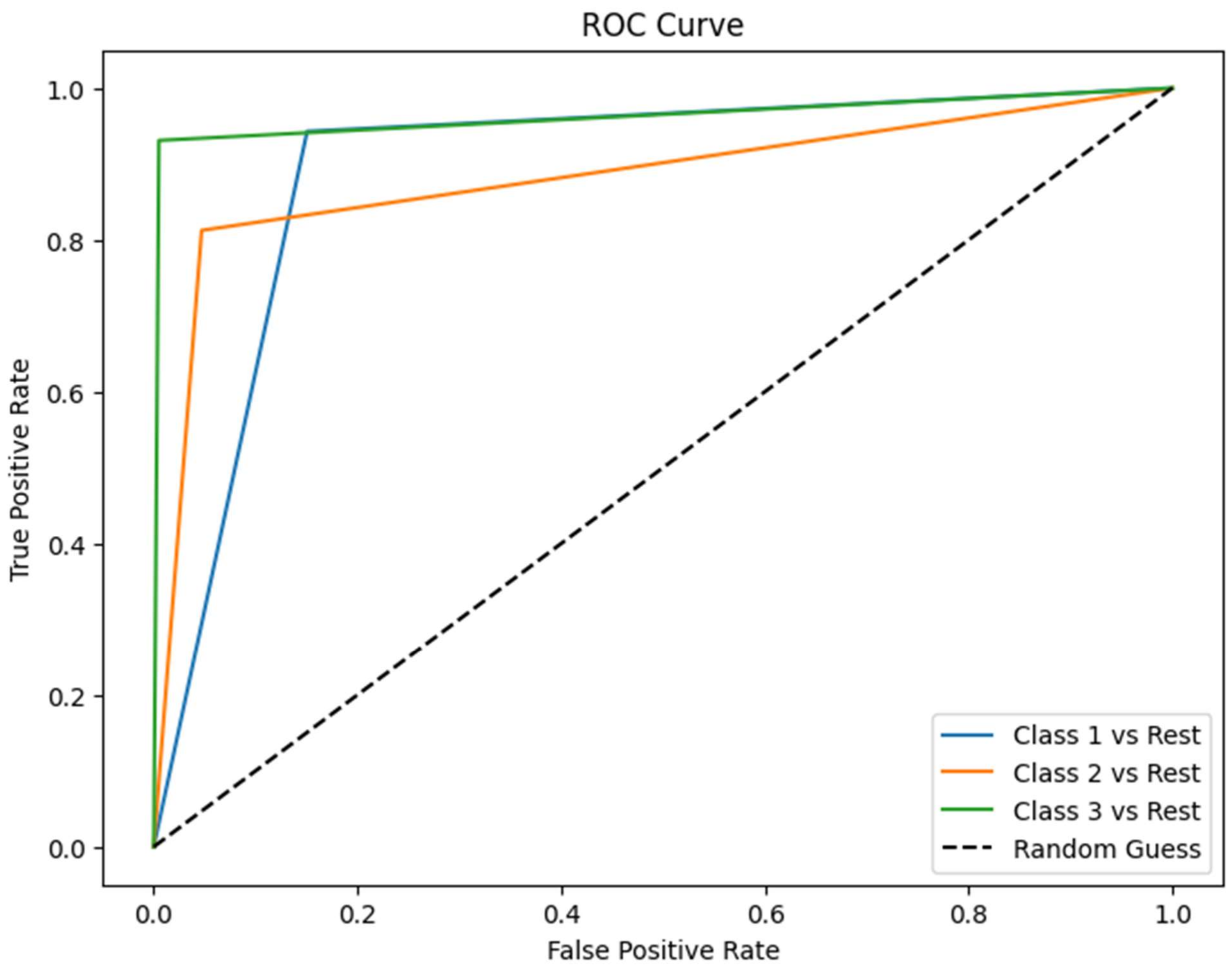
```
#get feature importance

feature_importance = dt_model.feature_importances_

#creating df to display importance scores

importance_df =pd.DataFrame({'Feature' : X.columns, 'Importance': feature_importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

print(importance_df)
                                 Feature  Importance
8          mean_value_of_short_term_variability    0.303693
17                              histogram_mean    0.261645
7              abnormal_short_term_variability    0.146010
9   percentage_of_time_with_abnormal_long_term_var...    0.133263
1                                accelerations    0.033909
0                               baseline value    0.032834
6                      prolongued_decelerations    0.022871
3                         uterine_contractions    0.022704
13                               histogram_max    0.018619
19                          histogram_variance    0.009623
11                             histogram_width    0.008293
12                               histogram_min    0.003627
20                          histogram_tendency    0.002116
4                         light_decelerations    0.000793
5                        severe_decelerations    0.000000
14                   histogram_number_of_peaks    0.000000
15                   histogram_number_of_zeroes    0.000000
16                              histogram_mode    0.000000
18                            histogram_median    0.000000
2                               fetal_movement    0.000000
10         mean_value_of_long_term_variability    0.000000
```

In [ ]:

```
# Calculate ROC AUC for each class separately (one-vs-rest strategy)
y_pred_prob = dt_model.predict_proba(X_test)
roc_auc = roc_auc_score(y_test, y_pred_prob, multi_class='ovr')
```

In [ ]:

```
# Plot the ROC curve for each class separately (one-vs-rest strategy)
plt.figure(figsize=(8, 6))
for class_label in range(1, 4):  # Assuming 3 classes (1, 2, 3)
    fpr, tpr, _ = roc_curve(y_test == class_label, y_pred_prob[:, class_label - 1])  # Adjust for 0-based index
    plt.plot(fpr, tpr, label=f'Class {class_label} vs Rest')
plt.plot([0, 1], [0, 1], 'k--', label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```



Appendix B

Link to dataset: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification

Link to GitHub repository: https://github.com/jennakobular/predicting_fetal_health