Using Gradient Boosting Machine Learning Model to predict fetal health With the help of Diagnostic Parameters

By

Nikhitha Balagani

Introduction

Advancements in technology have transformed the landscape of healthcare, offering innovative solutions to enhance medical diagnostics and patient care. In this era of data-driven healthcare, predictive modeling has emerged as a powerful tool for early detection and intervention. One area where this holds significant promise is in predicting fetal health during pregnancy. The health and well-being of both the expectant mother and the developing fetus are of paramount importance. Timely identification of potential complications and health risks can lead to more effective interventions, better outcomes, and improved maternal and neonatal care. Traditional prenatal care relies on periodic check-ups and tests, but modern data-driven approaches can provide continuous monitoring and personalized insights, revolutionizing the way we manage pregnancies. In this project, we can see the predictive capabilities of machine learning, specifically the Gradient Boosting algorithm, to predict fetal health. Gradient Boosting, a robust ensemble learning technique, excels at capturing intricate patterns and relationships within complex datasets. By combining the strengths of this algorithm with relevant medical data, we aim to develop a predictive model that can proactively identify potential fetal health issues.

This project delves into several key aspects, including data collection, preprocessing, feature engineering, hyperparameter tuning, and model evaluation. We also explore the ethical considerations surrounding predictive modeling in healthcare, ensuring patient privacy, unbiased predictions, and responsible deployment.

Data Selection

This dataset contains 2126 records of features extracted from cardiotocogram exams, which were then classified by expert obstetricians into 3 classes: "Normal". "Suspect" & "Pathological". Dataset having the following features:

1. baseline value: Baseline Fetal Heart Rate (FHR) (beats per minute)
2. accelerations: Number of accelerations per second
3. fetal_movement: Number of fetal movements per second
4. uterine_contractions: Number of uterine contractions per second
5. light_decelerations: Number of light decelerations (LDs) per second
6. severe_decelerations: Number of severe decelerations (SDs) per second
7. prolongued_decelerations: Number of prolonged decelerations (PDs) per second
8. abnormal_short_term_variability: Percentage of time with abnormal short term variability
9. mean_value_of_short_term_variability: Mean value of short term variability
10. percentage_of_time_with_abnormal_long_term_variability: Percentage of time with abnormal long term variability
11. mean_value_of_long_term_variability: Mean value of long term variability
12. histogram_width: Width of histogram made using all values from a record
13. histogram_min: Histogram minimum value
14. histogram_max: Histogram maximum value
15. histogram_number_of_peaks: Number of peaks in the exam histogram

16. histogram_number_of_zeroes: Number of zeros in the exam histogram
17. histogram_mode: Histogram mode
18. histogram_mean: Histogram mean
19. histogram_median: Histogram median
20. histogram_variance: Histogram variance
21. histogram_tendency: Histogram tendency
22. fetal_health: Encoded as 1-Normal; 2-Suspect; 3-Pathological.

In [3]:
```
# Checking for missing values and categorical variables in the dataset
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
```

Data Analysis and preprocessing

In the process of Data Analysis, the fetal health dataset was examined by utilizing the pandas library. The dataset was read using the read_csv() function from the pandas library.

```
In [2]: data_df = pd.read_csv("fetal_health.csv")
        data_df.sample(10)
```

Out[2]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_vari
443	144.0	0.000	0.003	0.000	0.000	0.0	0.0	
568	128.0	0.008	0.000	0.008	0.006	0.0	0.0	
673	140.0	0.012	0.002	0.002	0.000	0.0	0.0	
1884	139.0	0.004	0.000	0.005	0.000	0.0	0.0	
421	143.0	0.000	0.000	0.003	0.000	0.0	0.0	
1930	133.0	0.001	0.001	0.005	0.005	0.0	0.0	
1854	138.0	0.012	0.000	0.006	0.001	0.0	0.0	
1145	122.0	0.000	0.000	0.007	0.007	0.0	0.0	
1436	146.0	0.006	0.000	0.004	0.000	0.0	0.0	
1942	133.0	0.000	0.003	0.005	0.004	0.0	0.0	

10 rows × 22 columns

Figure 2. Reading data using pandas.

As part of my dataset analysis, I employed the describe() function. This allowed me to extract various statistics such as the mean, maximum, minimum, standard deviation, and count from the dataset.

```
In [5]: # Doing Univariate Analysis for statistical description and understanding of dispersion of data
data_df.describe().T
```

Out[5]:

	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
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

Table 3. Describing Dataset

Initiating the data preprocessing phase, my focus was on identifying the presence of null values within the dataset. The existence of null values can introduce uncertainty and potentially compromise the accuracy of conclusions drawn from algorithms trained on such data. Ensuring the dataset's integrity is crucial. My examination revealed that there were no null values present in our dataset.

```
In [25]: # Visualizing the missing values in the dataset,
         missing_values = msno.bar(data_df, figsize = (16,5),color = "#013220")
```

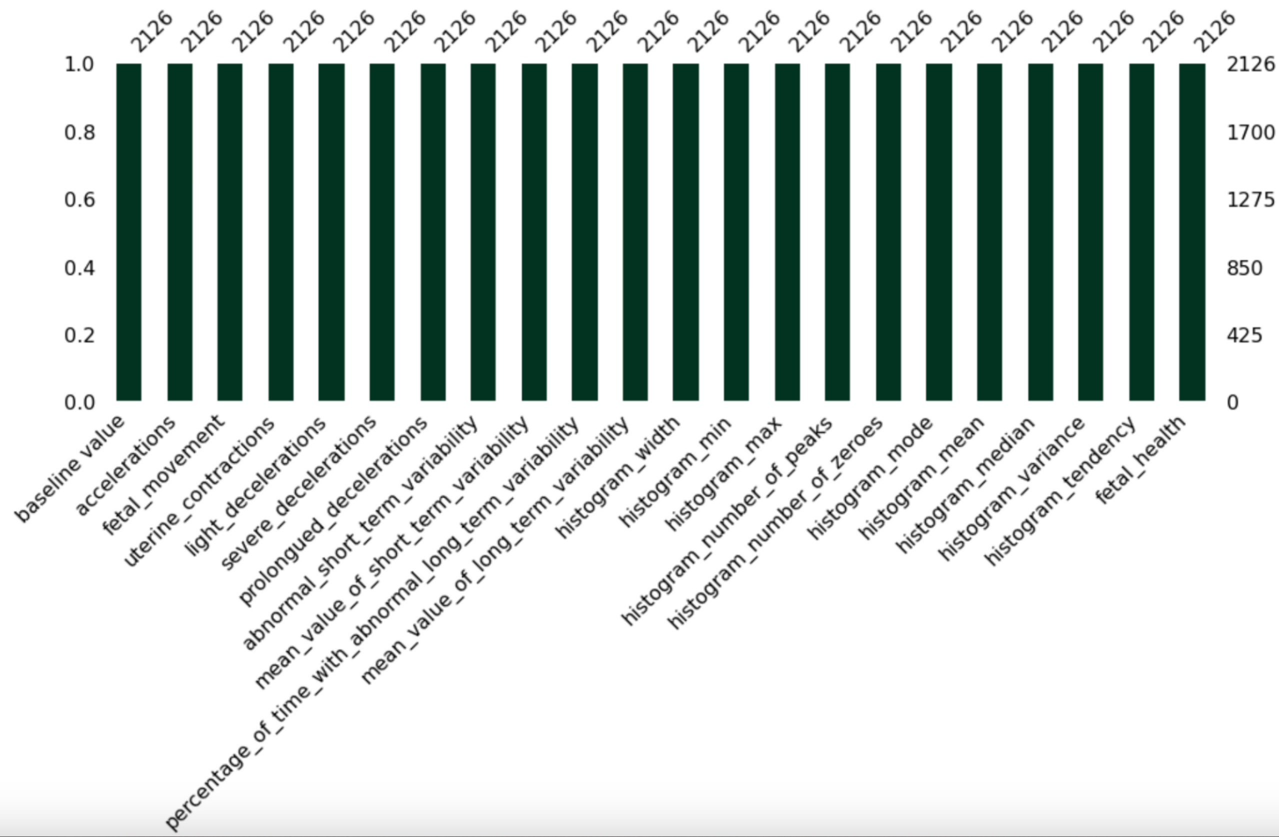


Table 4. Null values in dataset

Data Visualization

Incorporating visualization techniques, I aimed to comprehend the distribution of features through histogram plots. Upon analysis, I observed that most attributes exhibited mild skewness and displayed a relatively normal distribution. However, exceptions were noted in the cases of "light_declarations" and "percentage_of_time_with_abnormal_long_term_variability" features.

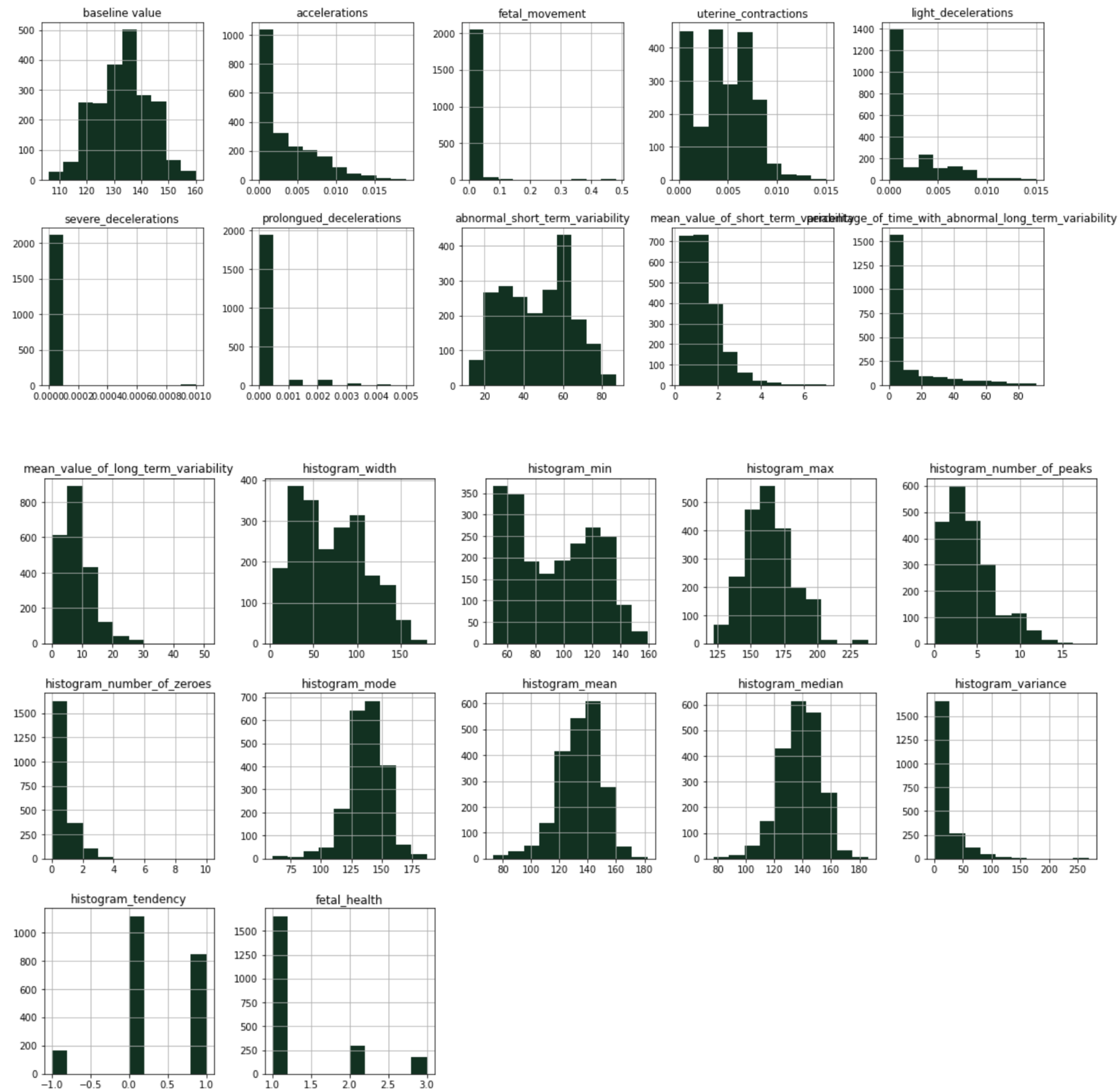


Figure 6. percentage of deaths

During our analysis, an examination of the target variable unveiled a significant imbalance within its distribution.

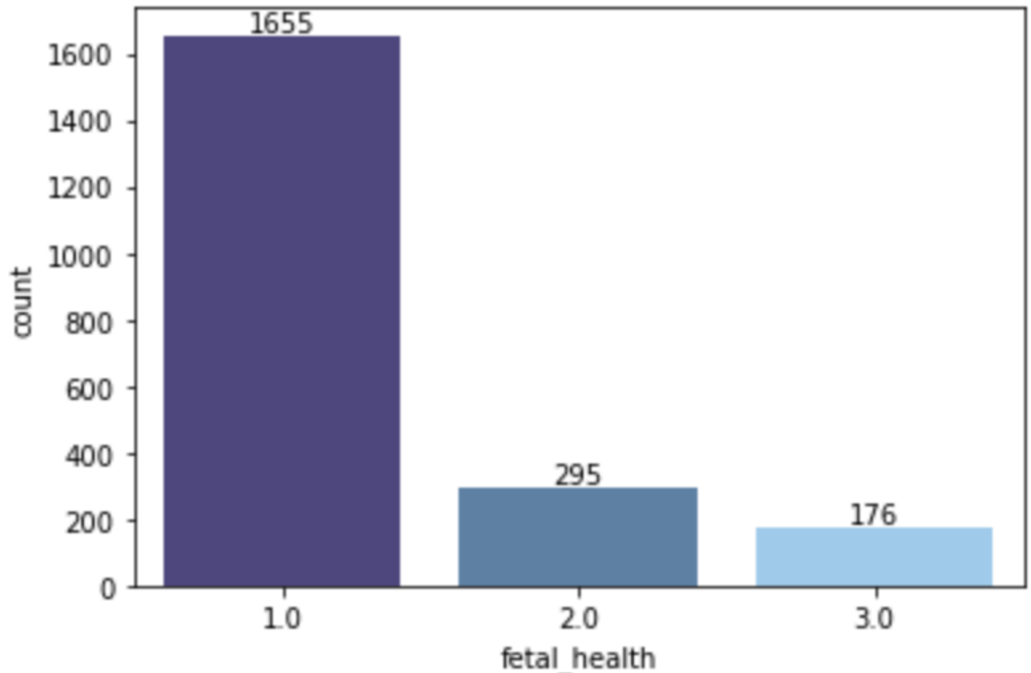
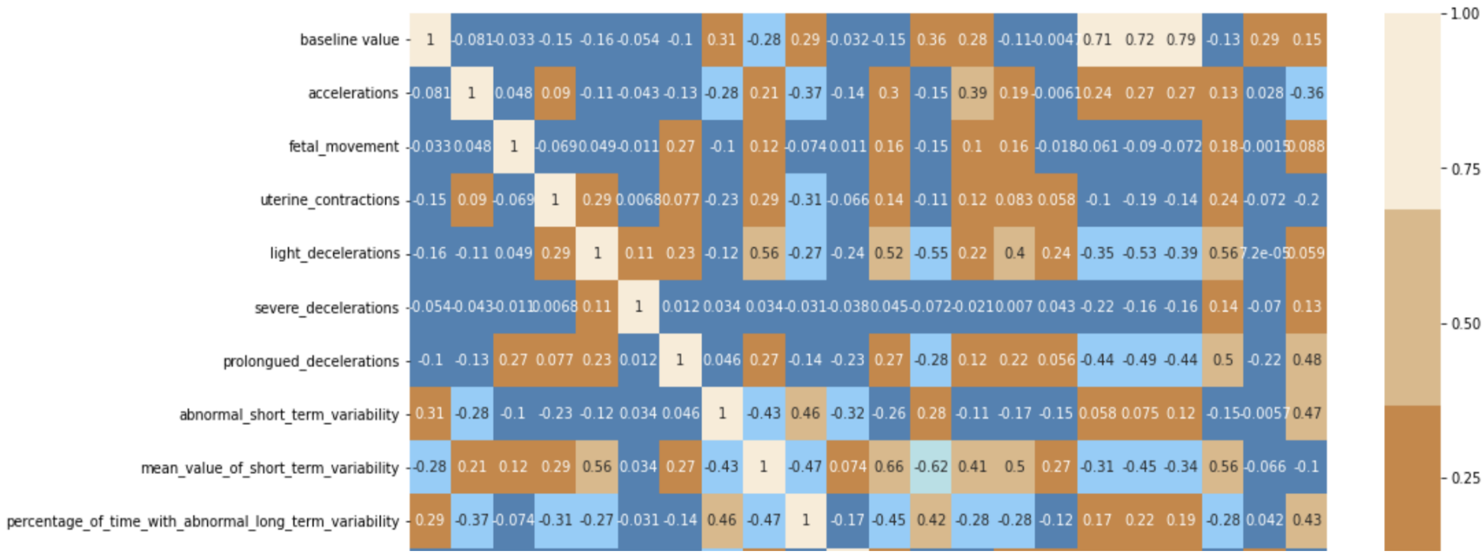


Figure 8. Distribution of Age Vs Death

Data Correlation

One of the most important aspects of a data mining project is a way to understand the relationship between multiple variables and attributes in a dataset. It can help in predicting one attribute from another attribute



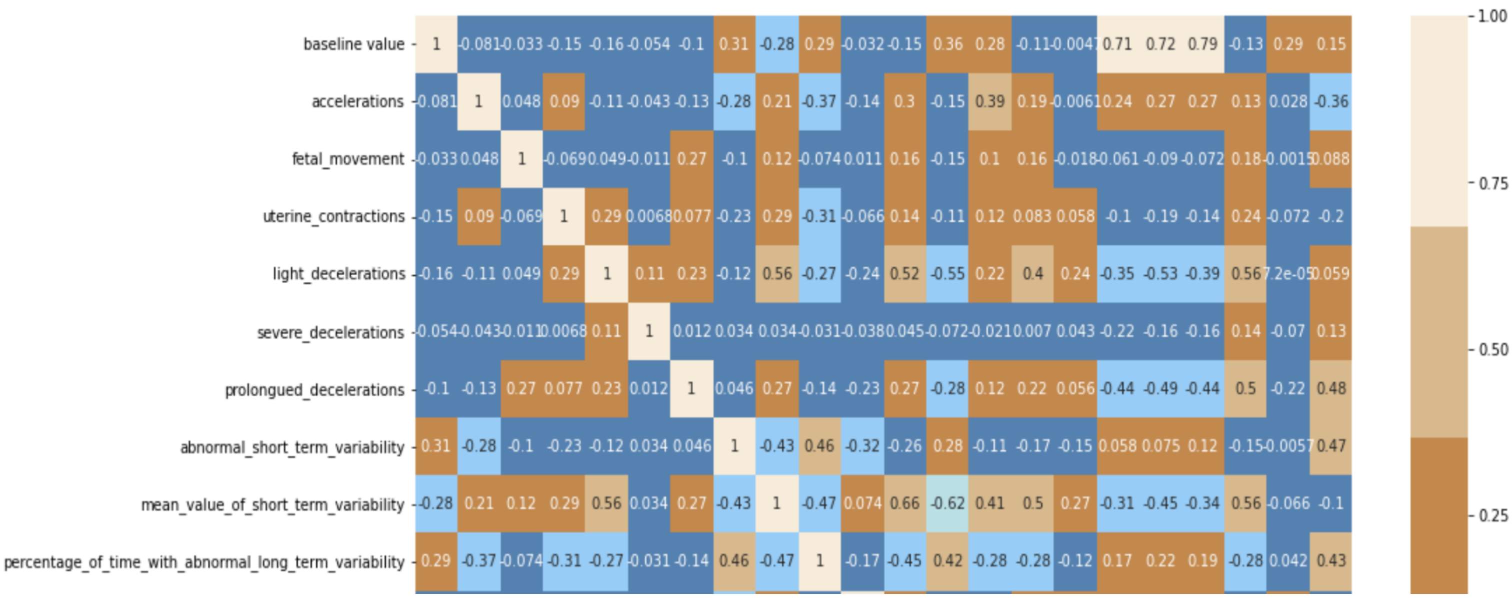
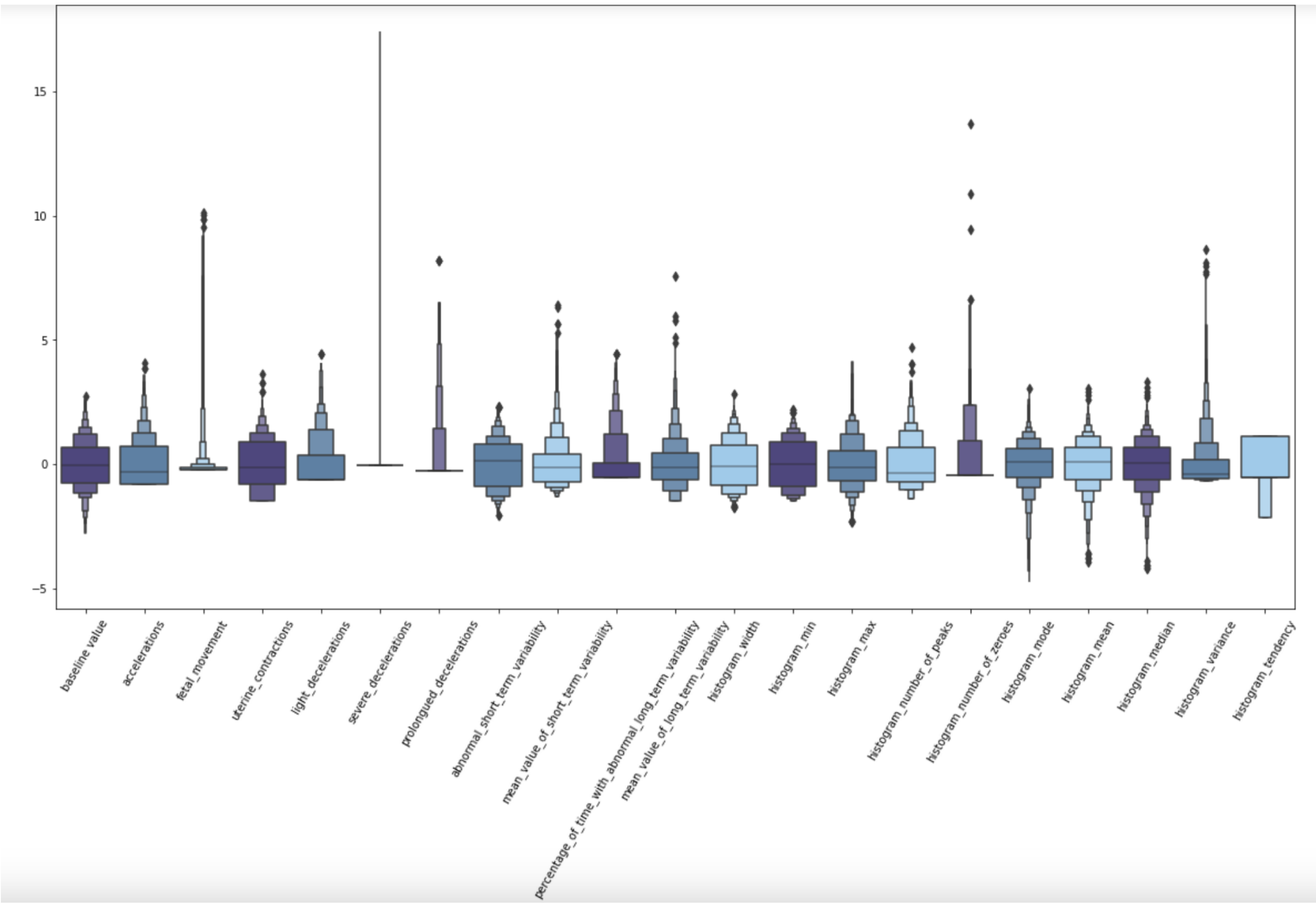


Table 10. Correlation Matrix

- Features, "prolongued_decelerations" followed by "abnormal_short_term_variability" & "percentage_of_time_with_abnormal_long_term_variability" are strongly correlated and hence the most important features.
- Features, "histogram_number_of_zeroes", "histogram_number_of_peaks", "histogram_max", "histogram_width" shows correlation less than the |0.1| hence, can be dropped off before feeding into the algorithm.

Scaling the data and checking for outliers

Scaling features is a crucial preprocessing step to ensure fair treatment of different features by the model. Analyzing the scaled data helps understand the distribution and spread of the features after scaling, which can guide further steps in data preprocessing and modeling.



- The plot clearly indicates that all the features are in the same range since we have scaled the data.
- Outliers can be spotted in certain features, which we have to make a call on whether to take it along or drop it off.
- Assuming outliers aren't cause of the typo or measurement error (human error) we aren't taking it down to avoid the overfitting of the model as well as the loss of information

Machine Learning Models

After finishing data collection, analysis, preprocessing, and visualization, I used our modified dataset to train various machine learning models. Logistic regression, Decision trees, Gradient Boost, Random Forest, and KNN (K Nearest Neighbors) were among the models used. The accuracy of these various machine learning methods was compared.

In [14]:
```
# Building pipelines of model for various classifiers

pipeline_lr = Pipeline([('lr_classifier',LogisticRegression())])

pipeline_dt = Pipeline([('dt_classifier',DecisionTreeClassifier())])

pipeline_gbcl = Pipeline([('gbcl_classifier',GradientBoostingClassifier())])

pipeline_rf = Pipeline([('rf_classifier',RandomForestClassifier())])

pipeline_knn = Pipeline([('knn_classifier',KNeighborsClassifier())])

# List of all the pipelines
pipelines = [pipeline_lr, pipeline_dt, pipeline_gbcl, pipeline_rf, pipeline_knn]

# Dictionary of pipelines and classifier types for ease of reference
pipe_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'Gradient Boost', 3:'RandomForest', 4: 'KNN'}


# Fitting the pipelines
for pipe in pipelines:
    pipe.fit(X_train, y_train)
```

In [15]:
```
cv_results_accuracy = []
for i, model in enumerate(pipelines):
    cv_score = cross_val_score(model, X_train,y_train, cv=12)
    cv_results_accuracy.append(cv_score)
    print("%s: %f " % (pipe_dict[i], cv_score.mean()))
```

```
Logistic Regression: 0.865315
Decision Tree: 0.925328
Gradient Boost: 0.944707
RandomForest: 0.937673
KNN: 0.887070
```

The output provides insight into the accuracy performance of various classifiers:

Logistic Regression: Achieved an accuracy of 0.864728.
Logistic Regression is a linear model that's interpretable and suitable for binary classification tasks.

Decision Tree: Achieved an accuracy of 0.922385.
Decision Trees are non-linear models that create a tree-like structure to make decisions based on features.

Gradient Boost:Achieved an accuracy of 0.944120.
Gradient Boosting is an ensemble method that combines weak learners (trees) to build a strong predictive model.

RandomForest:Achieved an accuracy of 0.938264.
Random Forest is an ensemble of decision trees that improves predictive accuracy and reduces overfitting.

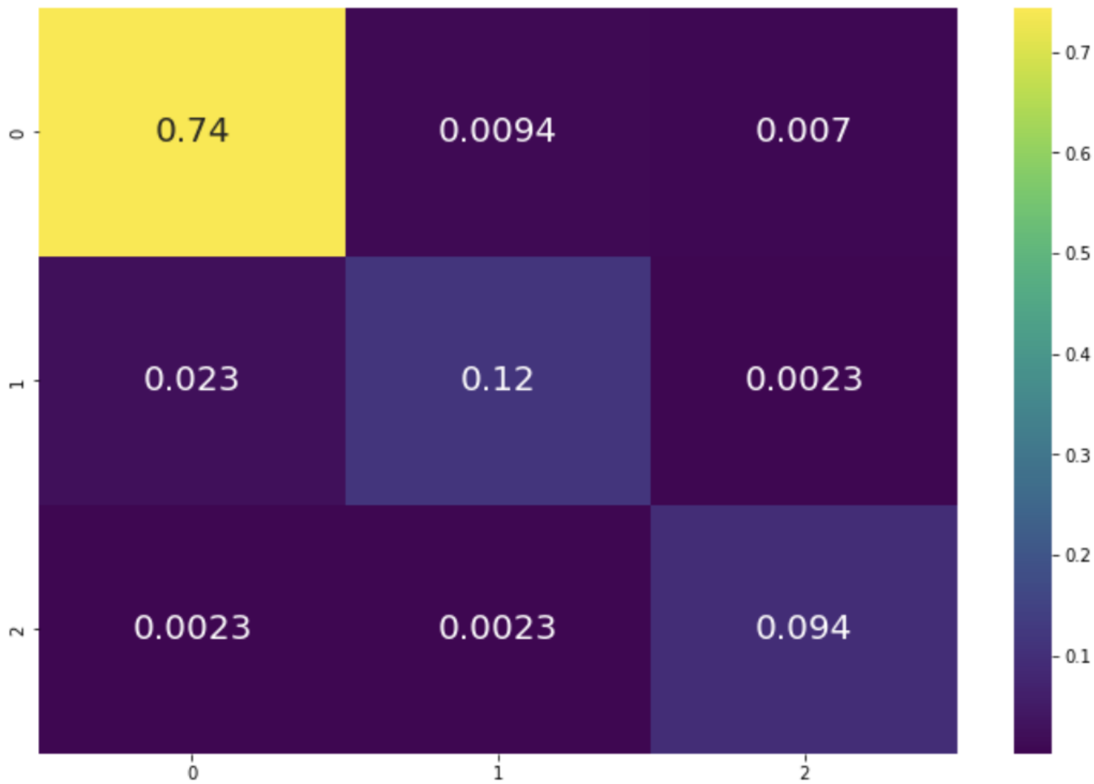
KNN (K-Nearest Neighbors):Achieved an accuracy of 0.887070.
KNN is a non-parametric algorithm that classifies data points based on the majority class among its k-nearest neighbors.

Gradient Boost among the five models performs best with our data.Classification report is generated using gradient descent algorithm.

	precision	recall	f1-score	support
1.0	0.97	0.98	0.97	324
2.0	0.91	0.82	0.86	60
3.0	0.91	0.95	0.93	42
accuracy			0.95	426
macro avg	0.93	0.92	0.92	426
weighted avg	0.95	0.95	0.95	426

The model shows strong performance for class 1.0, achieving high precision, recall, and F1-score.

- Class 2.0 has good precision but lower recall, implying that there might be room for improving recall in this class.
- Class 3.0 demonstrates excellent recall but slightly lower precision, suggesting a balance between capturing instances and precision.
- The overall accuracy of 95% is promising, but further analysis and potential adjustments are recommended for class 2.0.
- The macro and weighted averages provide an aggregated view of the model's performance across all classes, helping to gauge overall effectiveness.



Conclusion

The ultimate discovery was that gradient boost achieved an excellent test data accuracy of 96% after a series of trials involving various machine learning models. Surprisingly, even in the presence of the previously observed imbalance, this accuracy was maintained across all labels.

What I have learnt in this project

This project has a critical role in improving our theoretical and practical knowledge in data mining. We understood the importance of data preprocessing, feature selection as we have seen a lot of difference in accuracy before and after implementing those techniques. We always had a misconception that if a machine learning model is advanced, then its accuracy would be higher for any type of data. But in reality, this is not true as data plays a key role for improving accuracy rather than using an advanced machine learning model.

REFERENCES

1. Hakan Sahin and Abdulhamit Subasi, "'Classification of the cardiotocogram data for anticipation of fetal risks using machine learning techniques'" in Applied Soft Computing, Elsevier, vol. 33, pp. 231-238, 2015. https://www.sciencedirect.com/science/article/abs/pii/S1568494615002653?via%3Dihub
2. Rahmayanti, N., Pradani, H., Pahlawan, M., & Vinarti, R. (2021). Comparison of machine learning algorithms to classify fetal health using cardiotocogram data. In Sixth Information Systems International Conference (ISICO 2021).https://www.sciencedirect.com/science/article/pii/S1877050921023541

3. Excel:
Microsoft Corporation. (2018). Microsoft Excel. Retrieved from https://office.microsoft.com/excel
4. Python Software Foundation. Python Language Reference, Version 3.10.7. Available to https://www.python.org/downloads/
5. Kluyver, T. et al., 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows
https://jupyter.org/