SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
Faculty of Engineering and Technology

FETAL HEALTH CLASSIFICATION USING MACHINE LEARNING

MACHINE LEARNING IN HEALTHCARE

CA4 PROJECT REPORT

Submitted by

HARISH S – E6221007

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

(Data Science)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai -600116

Jan,2024

1.Problem Statement:

Maternal and fetal health monitoring during pregnancy is crucial to ensure the well-being of both the mother and the unborn child. Traditional methods of fetal health assessment often rely on manual interpretation of various medical parameters, leading to subjective results and potential delays in identifying critical conditions.

To address these challenges, our project focuses on leveraging Machine Learning (ML) techniques to develop an automated system for fetal health classification. The primary goal is to enhance the accuracy, efficiency, and timely detection of potential issues, providing healthcare professionals with a reliable tool to make informed decisions.

2.Dataset:

Title: Fetal Health Classification Dataset

This dataset, crucial for fetal health classification, is sourced from Kaggle, a leading platform for data science and machine learning.

Origin: Kaggle

The dataset was obtained from Kaggle, a platform known for hosting diverse and high-quality datasets for data science and machine learning projects.

Number of Rows: Exactly 2126

The dataset comprises 2126 instances, providing a robust foundation for training and evaluating machine learning models.

Key Column:

Fetal Heart Rate (FHR):

Description: This column typically represents the fetal heart rate, measured in beats per minute (bpm). FHR is a crucial parameter as it provides insights into the well-being of the fetus. Normal fetal heart rate ranges vary depending on the gestational age, and deviations from the norm can indicate potential health issues.

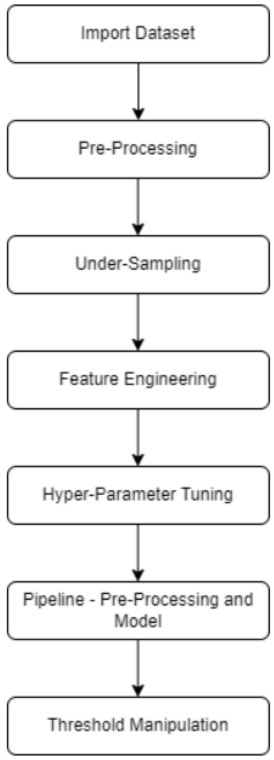
Uterine Contractions (UC):

Description: The Uterine Contractions column usually indicates the frequency and intensity of contractions in the mother's uterus. Monitoring uterine contractions is essential as excessive or insufficient contractions can impact fetal oxygen supply and overall well-being. The intensity and duration of contractions are crucial factors in assessing the uterine environment.

Fetal Movement (FM):

Description: The Fetal Movement column often represents the perceived or measured movements of the fetus. This can include kicks, rolls, or other observable motions. Monitoring fetal movement is crucial as it provides insights into the neurological development and overall well-being of the fetus. Changes in fetal movement patterns can be indicative of fetal distress or other health concerns.

3.Workflow:



4. Required libraries:

```
#import the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report
import matplotlib.pyplot as plt
import seaborn as sns
```

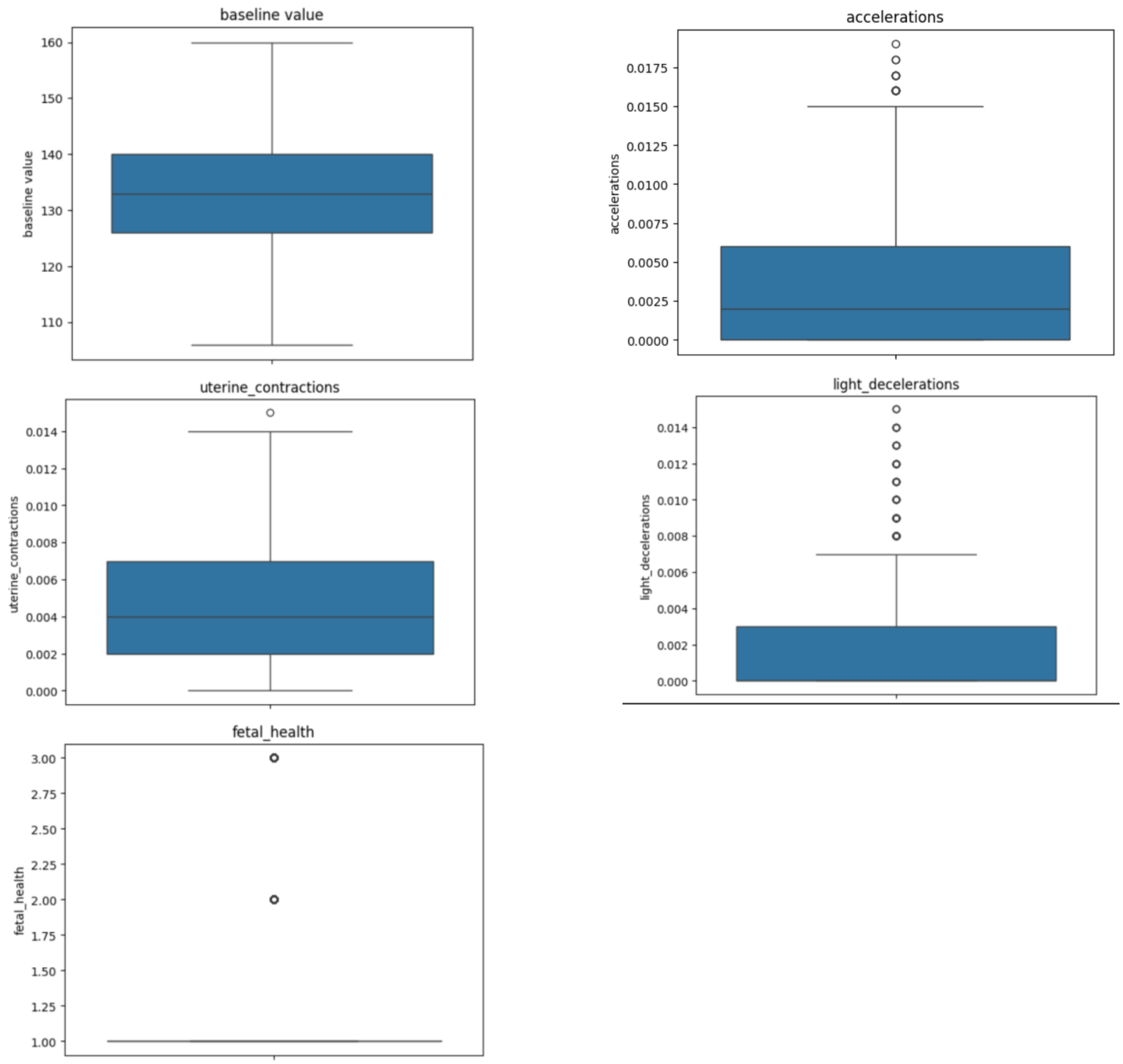
5. Data cleaning and pre processing

```
#load the dataset
df = pd.read_csv("C:\\Users\\Harish S\\Downloads\\fetal_health.csv")
```

```
df.isnull().sum()
```

```
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
```

```
#box plot for gathering the outlier info
plt.figure(figsize=(6,6))
for i in df.columns.values:
    sns.boxplot(df[i])
    plt.title(f'{i}')
    plt.show()
```



5.1 Exploratory Data Analysis:

```
df.head()
```

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

5 rows × 22 columns

```python
df.columns
```

```
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')
```

```python
df.shape
```

```
(2126, 22)
```

```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
```

```
df.describe()
```

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations
count	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
std	9.840844	0.003866	0.046666	0.002946	0.002960	0.000057	0.000590
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	0.000000
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000

8 rows × 22 columns

```
df.corr()
```

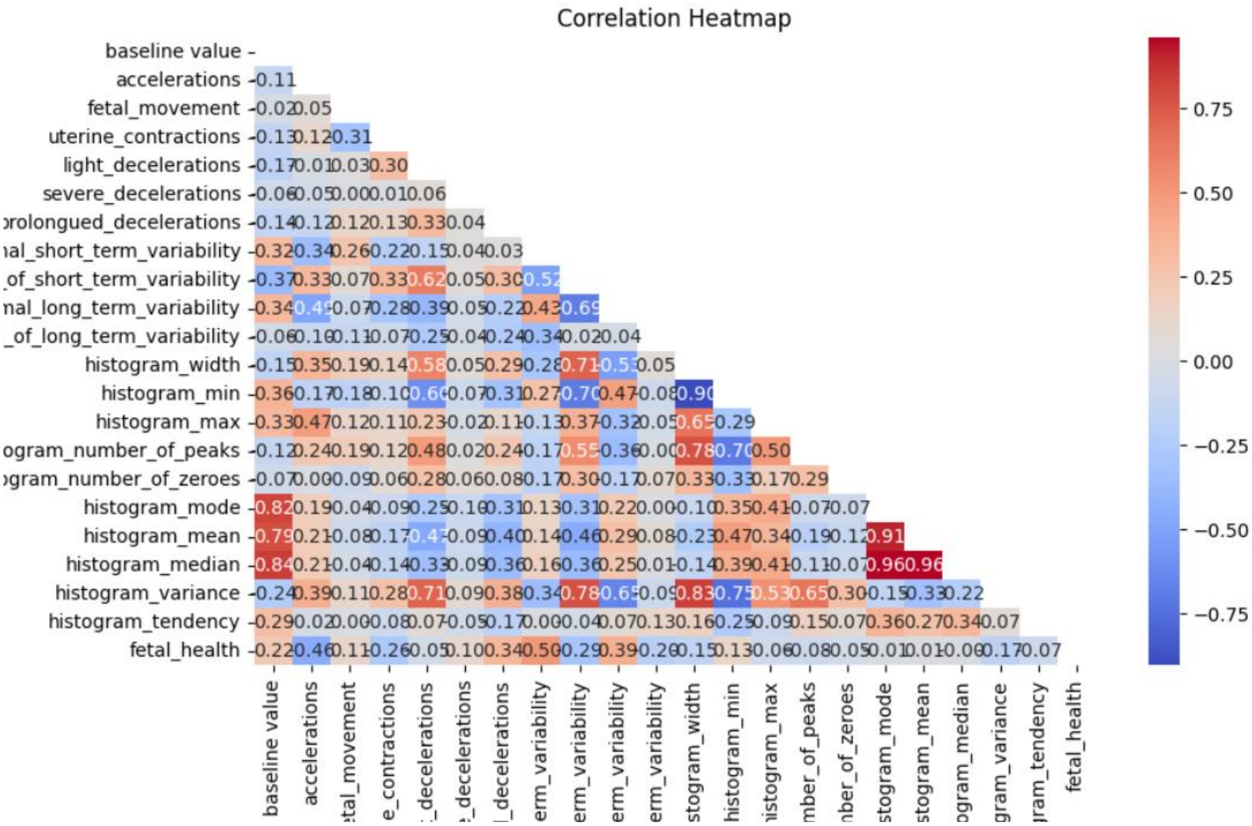
	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations
baseline value	1.000000	-0.080560	-0.033436	-0.146373	-0.159032
accelerations	-0.080560	1.000000	0.048235	0.089674	-0.108615
fetal_movement	-0.033436	0.048235	1.000000	-0.068779	0.049228
uterine_contractions	-0.146373	0.089674	-0.068779	1.000000	0.285079
light_decelerations	-0.159032	-0.108615	0.049228	0.285079	1.000000
severe_decelerations	-0.053518	-0.043018	-0.010976	0.006788	0.107573
prolongued_decelerations	-0.104597	-0.127749	0.265922	0.077036	0.225611
abnormal_short_term_variability	0.305570	-0.279577	-0.103715	-0.232811	-0.119152
mean_value_of_short_term_variability	-0.279607	0.207170	0.121314	0.289679	0.562170
percentage_of_time_with_abnormal_long_term_variability	0.285630	-0.373943	-0.074096	-0.306608	-0.271282
mean_value_of_long_term_variability	-0.032091	-0.142363	0.011047	-0.066058	-0.242932

5.2 Target Engineering:

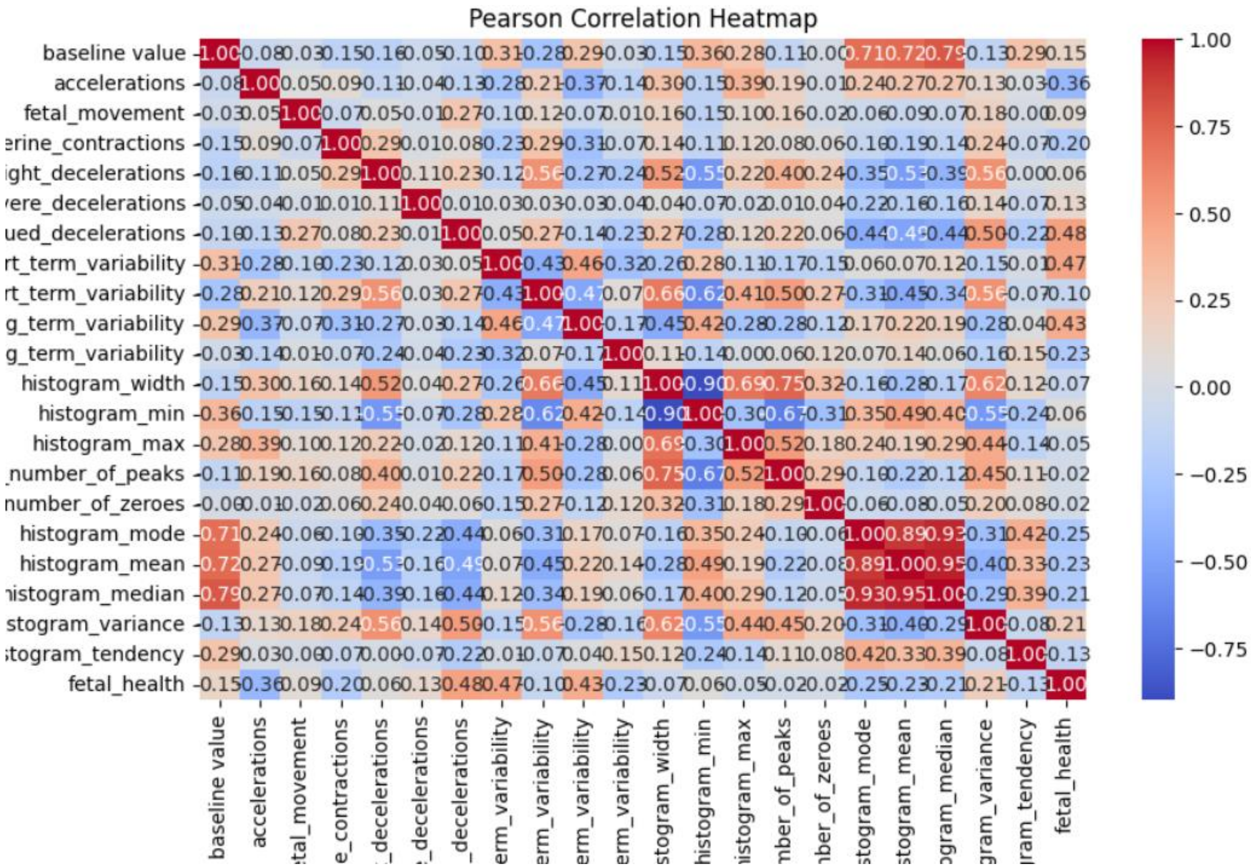
```
#value count
for i in df.columns.values:
    print(df[i].value_counts())
```

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

```python
# Calculate the correlation matrix
plt.figure(figsize=(10, 6))
correlation_matrix = df.corr(method = 'spearman')
mask_spearman = np.triu(correlation_matrix)
# Create a heatmap using seaborn
sns.heatmap(correlation_matrix, mask = mask_spearman,annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```




```python
# Calculate the Pearson correlation matrix
plt.figure(figsize=(10, 8))
correlation_matrix = df.corr(method='pearson')
# Create a heatmap using seaborn
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Pearson Correlation Heatmap')
plt.show()
```

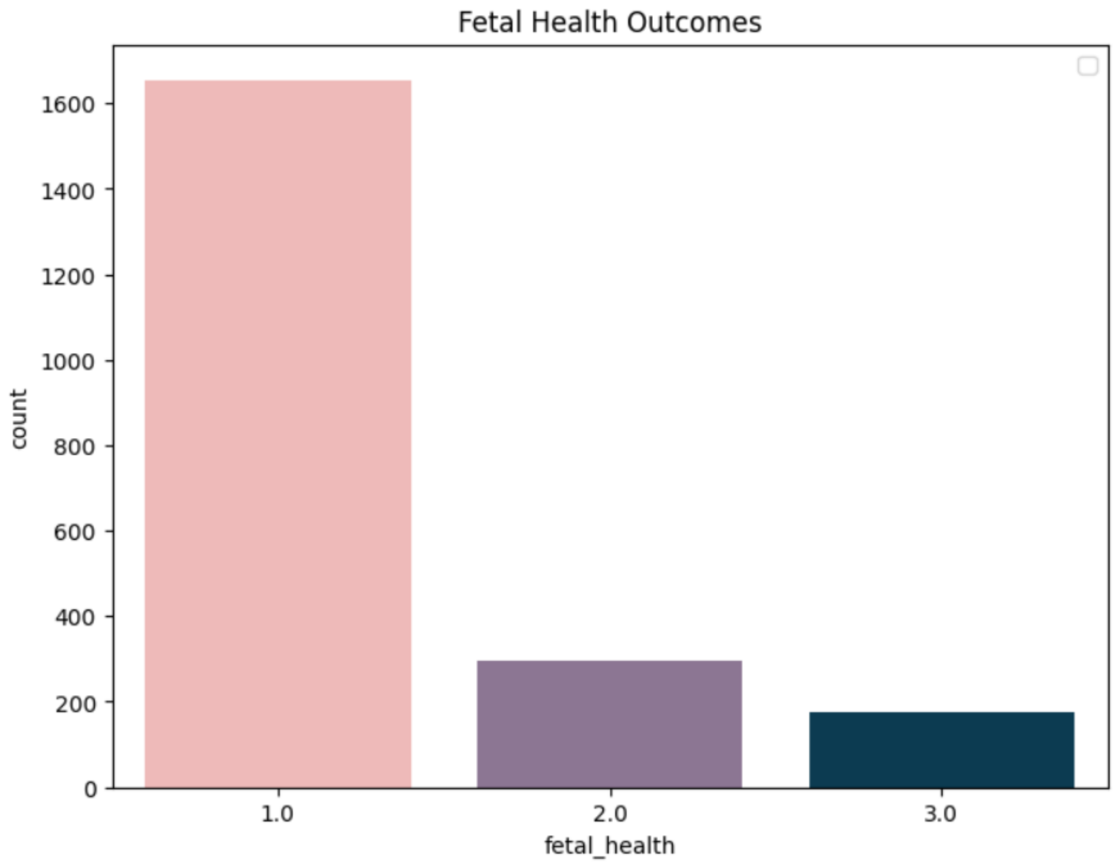


5.3 Splitting train, test and valid set:

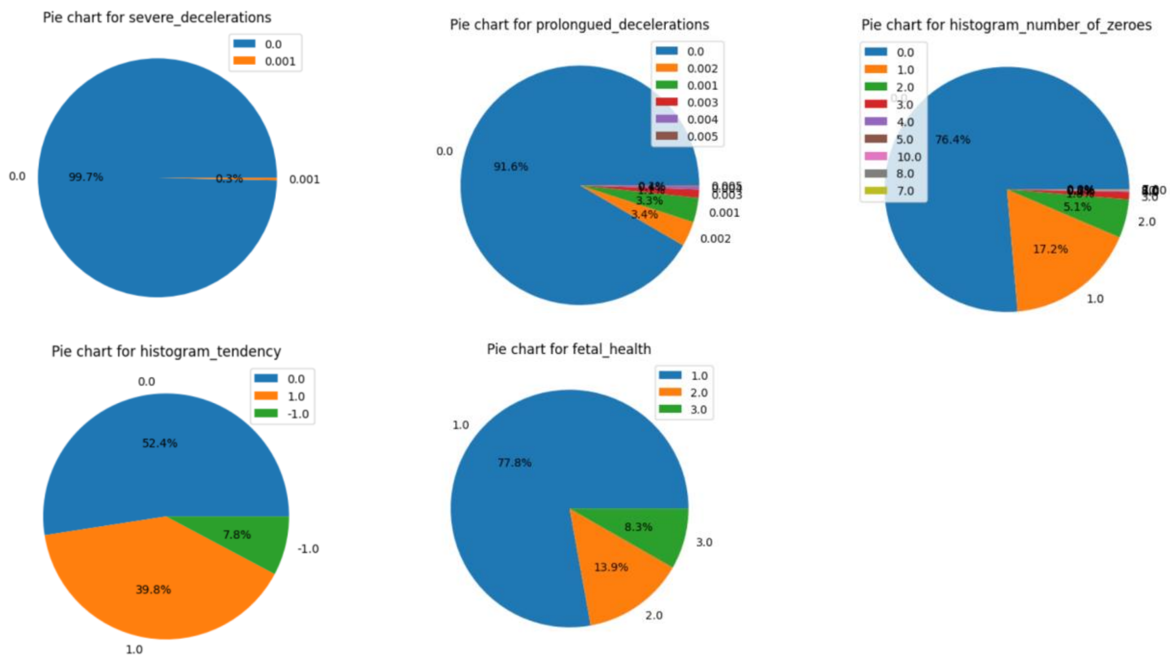
```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. Visualization:

```
#first of all let us evaluate the target and find out if our data is imbalanced or not
plt.figure(figsize=(8, 6))
plt.title('Fetal Health Outcomes')
plt.legend()
colours=["#f7b2b0","#8f7198", "#003f5c"]
sns.countplot(data= df, x="fetal_health",palette=colours)
```




```
#pie chart
for column in df.columns:
    if df[column].nunique()<=10:
        values = df[column].value_counts().values
        labels = df[column].value_counts().index
        plt.pie(values,labels = labels,autopct = "%1.1f%%")
        plt.title(f"Pie chart for {column}")
        plt.legend()
        plt.show()
```



7. Machine Learning Models

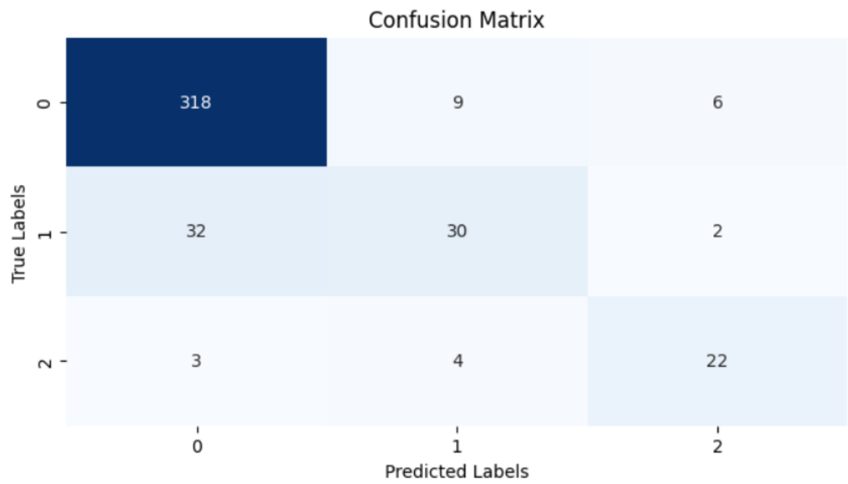
7.1 Logistic Regression:

```
#Logistic Regression
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming the target variable is 'fetal_health'
X = df.drop('fetal_health', axis=1)
y = df['fetal_health']

# Create a Logistic Regression model
logreg_model = LogisticRegression()

# Train the model on the training data
logreg_model.fit(X_train, y_train)

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

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
#confusion matrix as a heatmap
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

O/p:

```
Accuracy: 0.87
```



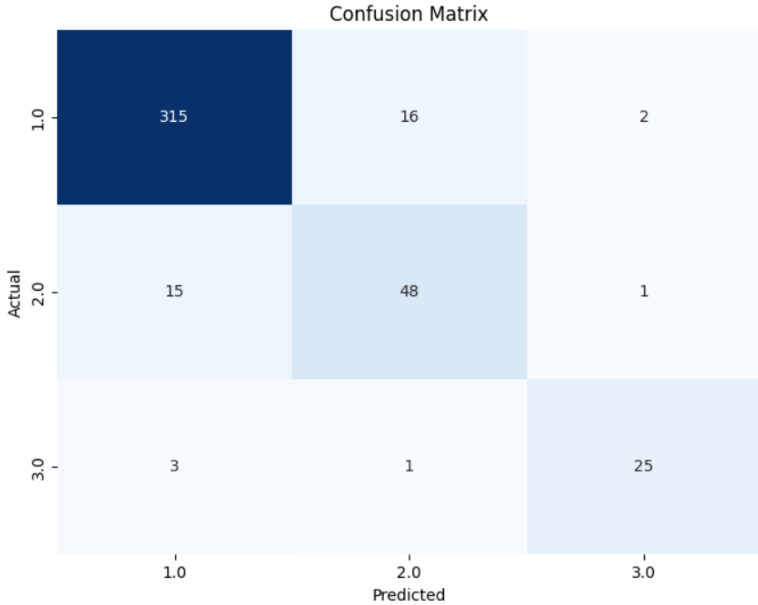
7.2 Decision Tree Classifier:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Assuming the target variable is 'fetal_health'
X = df.drop('fetal_health', axis=1)
y = df['fetal_health']

# Initialize the Decision Tree classifier
decision_tree_classifier = DecisionTreeClassifier(random_state=42)

# Train the classifier
decision_tree_classifier.fit(X_train, y_train)

# Make predictions on the test set
predictions = decision_tree_classifier.predict(X_test)

# Evaluate the accuracy
accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy}')
```

```
# Display confusion matrix
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=decision_tree_classifier.classes_, yticklabels=decision_tree_classifier.classes_)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

O\P:

```
Accuracy: 0.9225352112676056
```



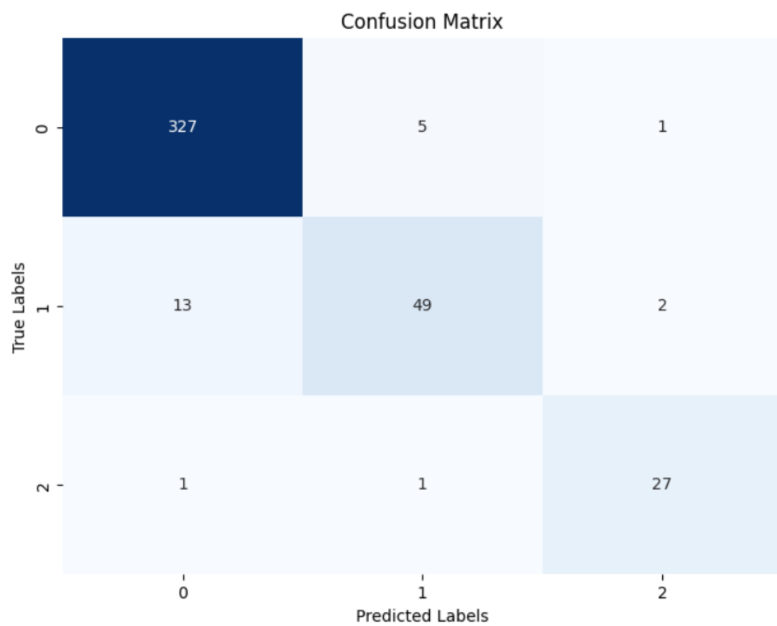
7.3 Random Forest Classifier:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Assuming the target variable is 'fetal_health'
X = df.drop('fetal_health', axis=1)
y = df['fetal_health']

# Create a Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training data
rf_classifier.fit(X_train, y_train)

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

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

O/P:

```
Accuracy: 0.95
```



7.4 Ada Boosting

```
#Assign vaue for x and y
X = df.iloc[:,:-1]
y = df["fetal_health"]

# Create an AdaBoost classifier
ada_classifier = AdaBoostClassifier(n_estimators=50, random_state=42)

# Train the classifier
ada_classifier.fit(X_train, y_train)

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

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

O/P:

```
Accuracy: 0.91
```