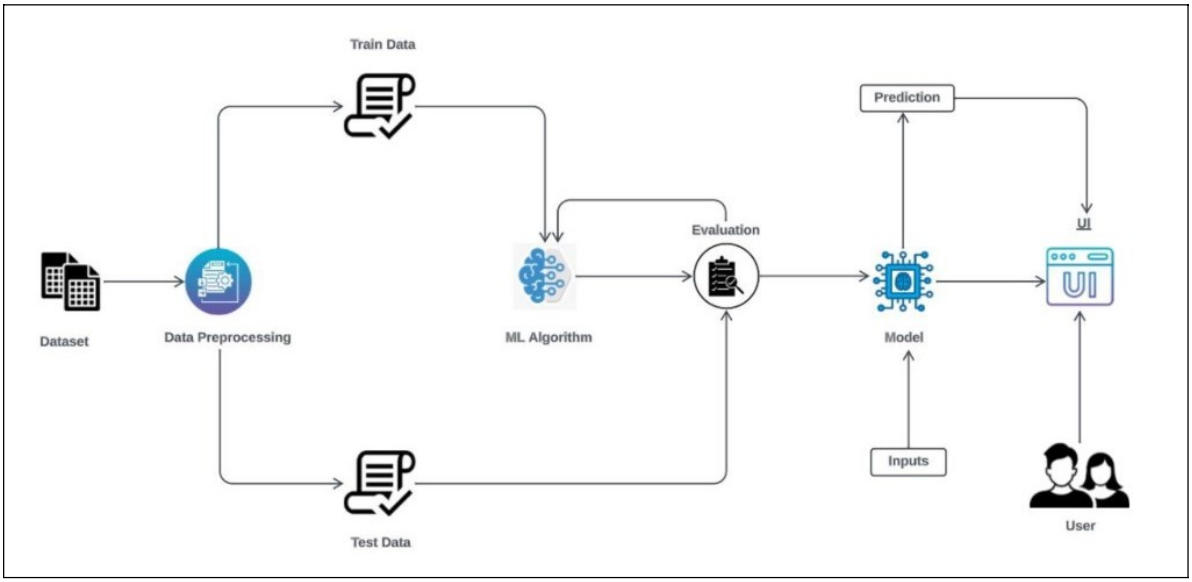
FetalAI: Using Machine Learning to predict and monitor Fetal Health

Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress.The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under 5 mortality to at least as low as 25 per 1,000 live births. Parallel to the notion of child mortality is of course maternal mortality, which accounts for 295 000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented.In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more. In this project, we have some characteristics of Fetal Health as a dataset. The target variable of this dataset is Fetal Health. Since it is a multi class classification, the classes are represented by 'Normal', 'Pathological' and 'Suspect'.

Technical Architecture



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyzes the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below

1) Define Problem / Problem Understanding

- Specify the business problem
- Business requirements
- Literature Survey
- Social or Business Impact.

2) Data Collection & Preparation

- Collect the dataset

3) Exploratory Data Analysis

- Descriptive statistical
- Visual Analysis
- Feature Selection
- Scaling the data
- Checking if the dataset is balanced or not

4) Model Building

- Splitting data into train and test
- Applying SMOTE for balancing the data
- Training the model after applying SMOTE
- Training the model in multiple algorithms
- Testing the model

5) Performance Testing

- Create dataframe of model performance
- Bar plot for model performance

6) Model Deployment

1. Save the best model
2. Integrate with Web Framework

Project Demonstration & Documentation

○ Record explanation Video for project end to end solution
○ Project Documentation-Step by step project development procedure

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress.The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under- 5 mortality to at least as low as 25 per 1,000 live births.Parallel to notion of child mortality is of course maternal mortality, which accounts for 295 000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented.In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more.

Activity 2: Business requirements

A Fetal Health classification project can have a variety of business requirements in the healthcare sector depending on the specific goals and objectives of the project. The business requirement for fetal health classification typically arises in the healthcare industry, specifically in the obstetrics and gynecology (OB/GYN) department. The classification of fetal health is necessary to ensure the well-being of the unborn baby and to make informed decisions regarding pregnancy management. There are several reasons why a healthcare provider may need to classify fetal health. For example, fetal health classification can be used to identify fetuses at risk of preterm labor, intrauterine growth restriction, or other medical conditions that could require early intervention or special care. Additionally, fetal health classification can help healthcare providers monitor the health of the fetus during pregnancy, identify any abnormalities or complications that may arise, and make informed decisions regarding delivery. In order to classify fetal health, healthcare providers typically use a variety of tools and techniques, including ultrasound, fetal monitoring, and other diagnostic tests. Machine learning and artificial intelligence algorithms can also be used to help classify fetal health based on various parameters such as heart rate, movement, and other physiological measures. These techniques can help healthcare providers to make more accurate and timely diagnosis and treatment decisions, leading to better health outcomes for both the mother and baby.

Activity 3: Literature Survey

Vankadari, P., & Verma, S. (2018). Fetal ECG extraction and QRS detection using Hilbert transform and empirical mode decomposition for fetal monitoring. This study focuses on extracting fetal ECG signals from abdominal recordings and uses machine learning techniques for QRS detection, a key element in fetal health assessment.

Peng, Y., Zhang, C., & Chen, H. (2019). A novel approach for fetal health monitoring using random forests. This paper introduces a novel approach using random forests for fetal health monitoring, with a focus on classifying fetal status based on various features.

Koprinska, I., Georgieva, A., & Jutten, C. (2008). Fetal QRS complex detection in abdominal recordings using second-order cyclostationary analysis. This work deals with fetal QRS complex detection, which is crucial for monitoring fetal well-being. Machine learning techniques may be integrated for classification.

Sammour, M. H., & Yao, J. (2019). Fetal heart rate variability analysis for early detection of fetal distress. The paper explores fetal heart rate variability and its potential for early detection of fetal distress, with implications for machine learning-based classification.

Activity 4: Social or Business Impact

Social Impact:

FetalAI aims to enhance prenatal care by reducing the exposure of pregnant mothers to invasive procedures and mitigating health risks for healthcare workers. Furthermore, the implementation of this advanced technology can create new job opportunities, including roles related to system maintenance, supervision, and oversight. It also promotes an increased understanding of prenatal health, recycling a culture of responsible prenatal care. Ultimately, FetalAI promises long-term benefits for fetal health and improved maternal well-being

Business Model/Impact:

The primary revenue stream for FetalAI is centered around the development, licensing, and sale of the automated fetal health monitoring technology. Potential revenue sources include equipment sales, maintenance services, collaboration with healthcare institutions for adoption, public awareness campaigns, and strategic partnerships with government agencies and non-governmental organizations. The project's business model emphasizes sustainability and a positive social impact.

Milestone 2: Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

● There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

● In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

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

● As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

● Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.3,random_state=0)
```

```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
```

```
from sklearn.neighbors import KNeighborsClassifier
model_knn=KNeighborsClassifier(n_neighbors=5)
model_knn.fit(x_train_smote,y_train_smote)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
from sklearn.metrics._plot.confusion_matrix import ConfusionMatrixDisplay
# print("Accuracy of KNeighborsClassifier: ",KNN_model.score(x_test,y_test))
cm=confusion_matrix(y_test,y_pred_knn)
cm_display=ConfusionMatrixDisplay(cm).plot()
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from six import StringIO
from IPython.display import Image
import pydotplus
from sklearn.tree import export_graphviz
```

```
from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier(criterion='entropy',n_estimators=10,random_state=0)
```

```
from sklearn.ensemble import GradientBoostingClassifier
gboost_model = GradientBoostingClassifier(n_estimators = 150, max_depth = 10, random_state = 10)
gboost_model.fit(x_train_smote, y_train_smote)
```

```
from sklearn.svm import SVC
```

```
from sklearn.model_selection import GridSearchCV
```

```
from tensorflow.keras.utils import to_categorical
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GRU, Dense , Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GRU, Dense, Flatten , Reshape
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.optimizers import Adam
from keras.regularizers import l2
from keras.regularizers import l1
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.
In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
In [5]: df=pd.read_csv("/content/fetal_health.csv")
        df
```

Out[5]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_ter
0	120.0	0.000	0.000	0.000	0.000	0.0	0.0	
1	132.0	0.006	0.000	0.006	0.003	0.0	0.0	
2	133.0	0.003	0.000	0.008	0.003	0.0	0.0	
3	134.0	0.003	0.000	0.008	0.003	0.0	0.0	
4	132.0	0.007	0.000	0.008	0.000	0.0	0.0	
...	...	...	...	...	...	...	...	
2121	140.0	0.000	0.000	0.007	0.000	0.0	0.0	
2122	140.0	0.001	0.000	0.007	0.000	0.0	0.0	
2123	140.0	0.001	0.000	0.007	0.000	0.0	0.0	
2124	140.0	0.001	0.000	0.006	0.000	0.0	0.0	
2125	142.0	0.002	0.002	0.008	0.000	0.0	0.0	

2126 rows × 22 columns

```
In [6]: df.head()
```

Out[6]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_
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

```
In [7]: df.shape
Out[7]: (2126, 22)
```

Activity 2: Data Preparation

As we have understood how the data is, let us pre-process the collected data. The Machine Learning model cannot be trained on the imported data directly. The dataset might have randomness, we might have to clean the dataset and bring it in the right form. This activity involves the following steps:

- Handling Missing Values
- Encoding data
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Handling Missing Values:

```
In [8]: 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
```

Null values checking:

```
In [9]: df.isnull().any()
```

```
Out[9]: baseline value                                          False
accelerations                                           False
fetal_movement                                          False
uterine_contractions                                    False
light_decelerations                                     False
severe_decelerations                                    False
prolongued_decelerations                                False
abnormal_short_term_variability                         False
mean_value_of_short_term_variability                    False
percentage_of_time_with_abnormal_long_term_variability  False
mean_value_of_long_term_variability                     False
histogram_width                                         False
histogram_min                                           False
histogram_max                                           False
histogram_number_of_peaks                               False
histogram_number_of_zeroes                              False
histogram_mode                                          False
histogram_mean                                          False
histogram_median                                        False
histogram_variance                                      False
histogram_tendency                                      False
fetal_health                                            False
dtype: bool
```

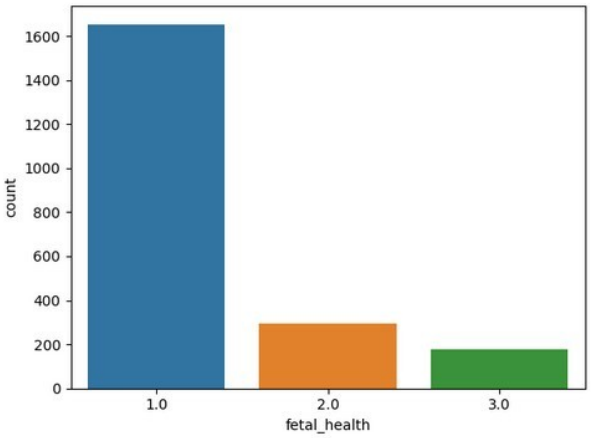
```
In [10]: df.isnull().sum()
```

```
Out[10]: 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
```

There are no missing values in the dataset. That is why we can skip this step.

Handling Imbalance Data

```
In [17]: df.fetal_health.value_counts()
Out[17]: 1.0    1655
         2.0     295
         3.0     176
         Name: fetal_health, dtype: int64

In [18]: sns.countplot(data=df,x="fetal_health")
Out[18]: <Axes: xlabel='fetal_health', ylabel='count'>
```



After checking, we get to know that the dataset is highly imbalanced. So in the later stages we have balanced the dataset before training the model.

Milestone 3: Exploratory Data Analysis

Activity 1: Descriptive statistical analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

```
In [10]: df.describe()
```

Out[10]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2
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

```
In [11]: df.nunique()
```

Out[11]:

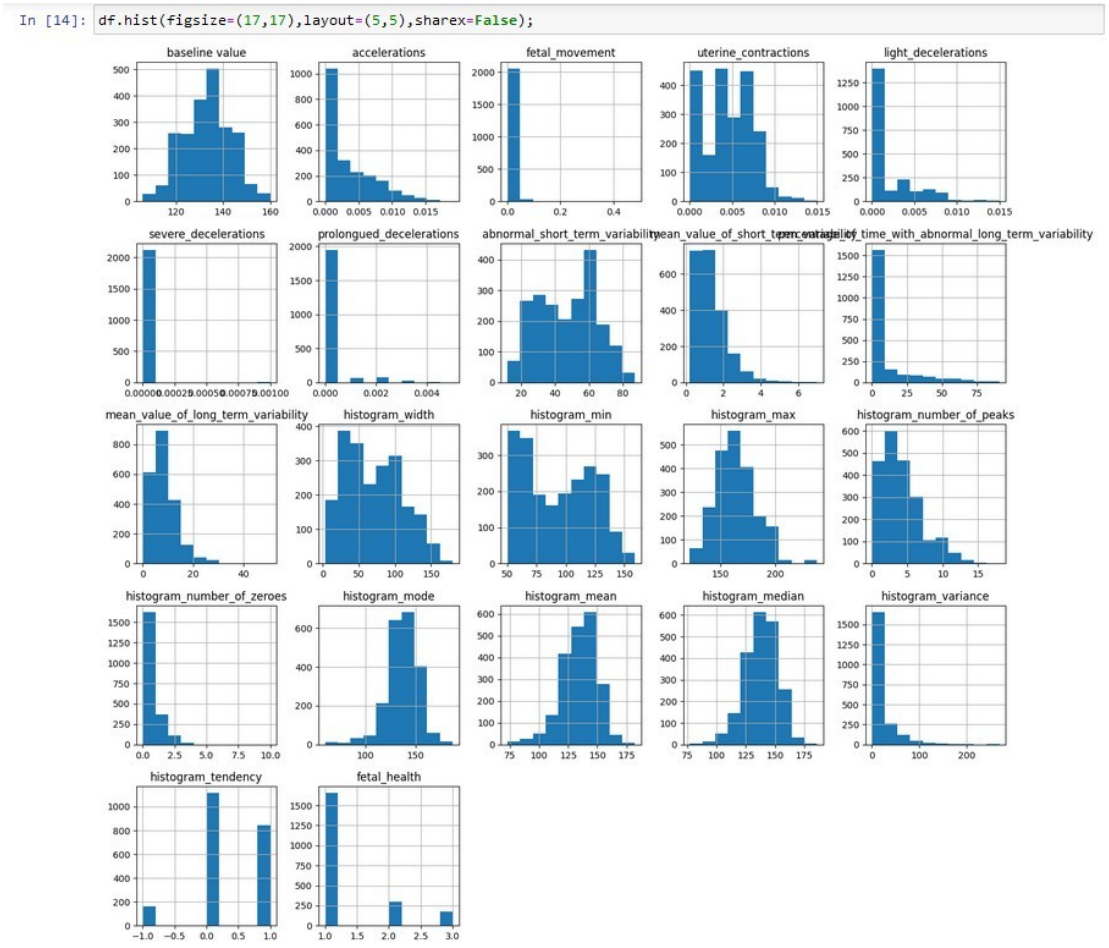
```
baseline value                                          48
accelerations                                           20
fetal_movement                                         102
uterine_contractions                                    16
light_decelerations                                     16
severe_decelerations                                     2
prolongued_decelerations                                 6
abnormal_short_term_variability                         75
mean_value_of_short_term_variability                    57
percentage_of_time_with_abnormal_long_term_variability  87
mean_value_of_long_term_variability                    249
histogram_width                                        154
histogram_min                                          109
histogram_max                                           86
histogram_number_of_peaks                               18
histogram_number_of_zeroes                               9
histogram_mode                                          88
histogram_mean                                         103
histogram_median                                        95
histogram_variance                                     133
histogram_tendency                                       3
fetal_health                                             3
dtype: int64
```

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

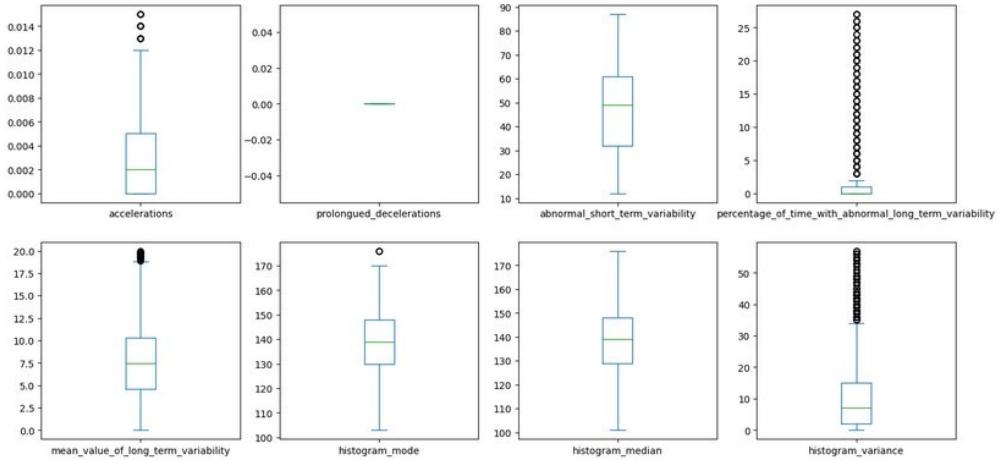
Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed different graphs such as histogram and boxplot. The Seaborn and matplotlib package provides a wonderful functions histogram and boxplot. With the help of histogram and boxplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.



```
In [42]: x.plot(kind='box',figsize=(17,17),layout=(4,4),sharex=False,subplots=True)
```

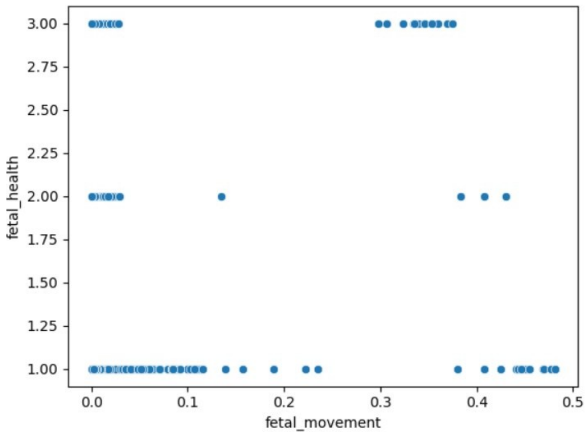
```
Out[42]: accelerations                                          Axes(0.125,0.712609;0.168478x0.167391)
         prolongued_decelerations                               Axes(0.327174,0.712609;0.168478x0.167391)
         abnormal_short_term_variability                        Axes(0.529348,0.712609;0.168478x0.167391)
         percentage_of_time_with_abnormal_long_term_variability Axes(0.731522,0.712609;0.168478x0.167391)
         mean_value_of_long_term_variability                    Axes(0.125,0.511739;0.168478x0.167391)
         histogram_mode                                         Axes(0.327174,0.511739;0.168478x0.167391)
         histogram_median                                       Axes(0.529348,0.511739;0.168478x0.167391)
         histogram_variance                                     Axes(0.731522,0.511739;0.168478x0.167391)
         dtype: object
```



Bivariate Analysis:

```
In [24]: sns.scatterplot(x=df['fetal_movement'],y=df['fetal_health'])
```

```
Out[24]: <Axes: xlabel='fetal_movement', ylabel='fetal_health'>
```

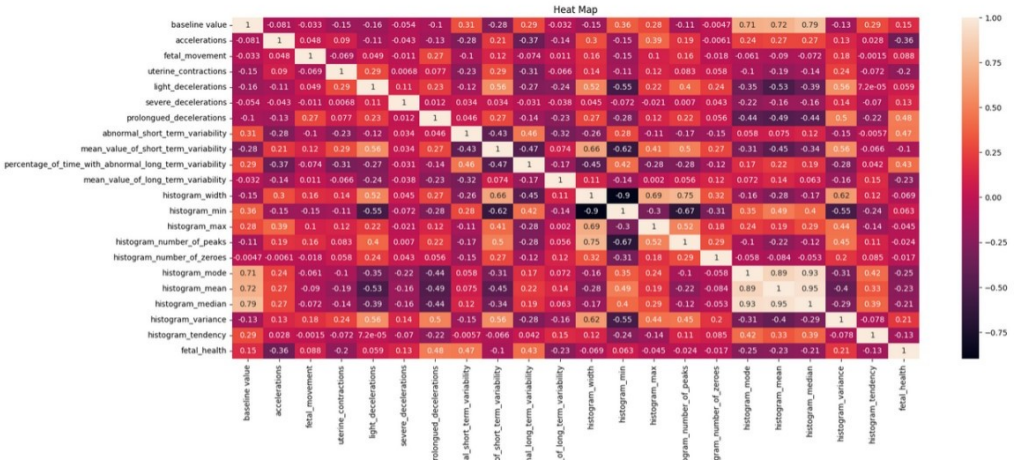


Activity 2.2: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used correlation matrix

```
In [29]: plt.figure(figsize=(20,8))
         sns.heatmap(df.corr(),annot=True)
         plt.title("Heat Map")
```

```
Out[29]: Text(0.5, 1.0, 'Heat Map')
```



Activity 3: Feature Selection

```
In [30]: df=df.drop(columns=['histogram_mean'],axis=1) # multicollinearity between histogram_mean and histogram_median
```

```
In [33]: df.corr().fetal_health.sort_values(ascending=False)
```

```
Out[33]: 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_mode                                           -0.250412
         accelerations                                            -0.364066
         Name: fetal_health, dtype: float64
```

Activity 4: Scaling the data

In [34]:
```
# Calculating correlation between features and target column
correlations = df.drop(columns=['fetal_health']).apply(lambda x: x.corr(df['fetal_health']))

# Dropping columns with correlation < 0.205
columns_to_drop = correlations[abs(correlations) < 0.205].index
df.drop(columns=columns_to_drop,inplace=True)

# print("Filtered DataFrame:")
df.head()
```

Out[34]:

	accelerations	prolongued_decelerations	abnormal_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	mean_value_of_long_term_varial
0	0.000	0.0	73.0	43.0	
1	0.006	0.0	17.0	0.0	
2	0.003	0.0	16.0	0.0	
3	0.003	0.0	16.0	0.0	
4	0.007	0.0	16.0	0.0	

Activity 1: Splitting data into train and test

In [50]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.3,random_state=0)
```

In [51]:
```
y_train.value_counts()
```

Out[51]:
```
1.0    1161
2.0     209
3.0     118
Name: fetal_health, dtype: int64
```

In [52]:
```
x_train.shape
```

Out[52]:
```
(1488, 8)
```

In [53]:
```
x_test.shape
```

Out[53]:
```
(638, 8)
```

In [54]:
```
y_train.shape
```

Out[54]:
```
(1488,)
```

In [55]:
```
y_test.shape
```

Out[55]:
```
(638,)
```

Activity 2: Applying SMOTE for balancing the data

In [56]:
```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
```

In [57]:
```
x_train_smote,y_train_smote=smote.fit_resample(x_train,y_train)
```

In [58]:
```
y_train_smote.value_counts()
```

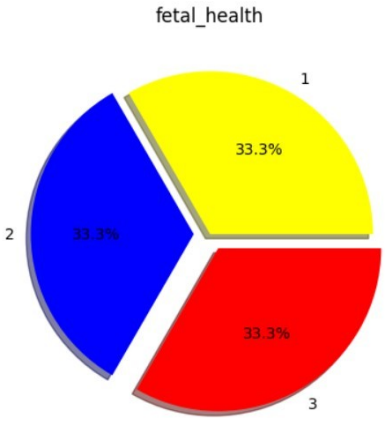
Out[58]:
```
1.0    1161
3.0    1161
2.0    1161
Name: fetal_health, dtype: int64
```

In [59]:
```
y_train_smote
```

Out[59]:
```
0       1.0
1       1.0
2       1.0
3       1.0
4       1.0
       ...
3478    3.0
3479    3.0
3480    3.0
3481    3.0
3482    3.0
Name: fetal_health, Length: 3483, dtype: float64
```

```
In [60]: plt.pie(y_train_smote.value_counts(),[0,0.1,0.1],labels=['1','2','3'],colors=['yellow','blue','red'],autopct="%1.1f%%",shadow='Tr
         print('After SMOTE :')
         plt.title("fetal_health")
         plt.show()
```

```
After SMOTE :
```



Milestone 4: Model Building

Training the model after applying SMOTE

Activity 4: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Activity 4.1: Random forest model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done

```
from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier(criterion='entropy',n_estimators=10,random_state=0)
```

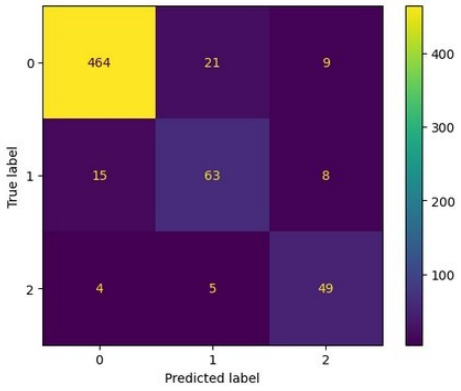
```
model_rf.fit(x_train_smote,y_train_smote)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

In [88]:
```
print('Testing_accuracy= ',accuracy_score(y_test,y_pred_rf))
print('Training_accuracy= ',accuracy_score(y_train_smote,y_pred_rf_train))
```

```
Testing_accuracy=  0.9294670846394985
Training_accuracy=  0.9979902383003159
```

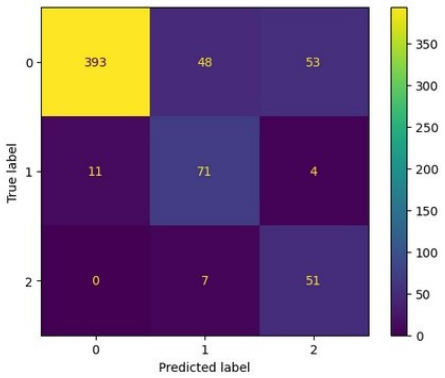
In [87]:
```
cm=confusion_matrix(y_test,y_pred_rf)
cm_display=ConfusionMatrixDisplay(cm).plot()
plt.show()
```



Activity 4.2: Decision Tree

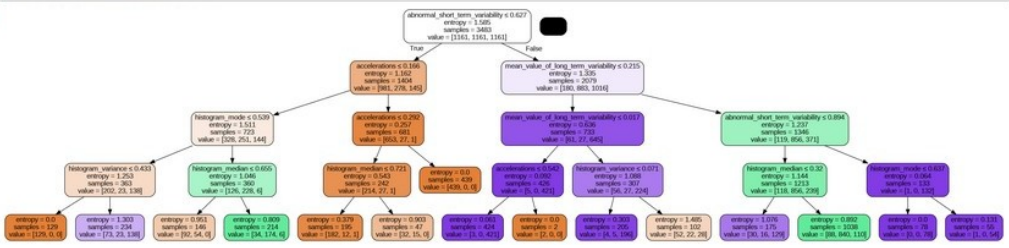
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
In [66]: from sklearn.tree import DecisionTreeClassifier

In [67]: model_dt=DecisionTreeClassifier(max_depth=4,splitter='best',criterion='entropy')

In [68]: model_dt.fit(x_train_smote,y_train_smote)

Out[68]: DecisionTreeClassifier(criterion='entropy', max_depth=4)
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [69]: y_pred_dt=model_dt.predict(x_test)
         y_pred_dt

Out[69]: array([1., 1., 1., 2., 2., 1., 1., 1., 1., 2., 3., 1., 1., 1., 3., 1., 3.,
                1., 1., 2., 1., 1., 1., 3., 1., 1., 1., 2., 1., 2., 1., 1., 3., 1.,
                3., 1., 2., 1., 2., 1., 1., 2., 1., 1., 2., 1., 1., 1., 1., 1.,
                2., 2., 1., 1., 1., 2., 1., 1., 1., 1., 2., 1., 2., 1., 2.,
                1., 1., 2., 1., 1., 1., 1., 1., 1., 1., 1., 3., 2., 1., 1., 1., 1.,
                1., 3., 1., 2., 1., 1., 1., 1., 1., 2., 1., 3., 1., 1., 2., 1., 1.,

In [70]: y_pred_dt_train=model_dt.predict(x_train_smote)
         y_pred_dt_train

Out[70]: array([1., 1., 1., ..., 3., 3., 3.])

In [71]: print('Testing_accuracy= ',accuracy_score(y_test,y_pred_dt))
         print('Training_accuracy= ',accuracy_score(y_train_smote,y_pred_dt_train))

         Testing_accuracy=  0.8072100313479624
         Training_accuracy=  0.8492678725236865

In [73]: cm=confusion_matrix(y_test,y_pred_dt)
         cm_display=ConfusionMatrixDisplay(cm).plot()
         plt.show()
```




```
In [76]: from six import StringIO
         from IPython.display import Image
         import pydotplus
         from sklearn.tree import export_graphviz

In [77]: dot_data=StringIO()
         export_graphviz(model_dt,out_file=dot_data,feature_names=x_train_smote.columns,filled=True,rounded=True,special_charac
         graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
         Image(graph.create_png())
```

Out[77]:



Activity 4.3: KNN

A function named KNeighborsClassifier() is created and train and test data are passed as the parameters. Inside the function, KNeighbors algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

In [56]:
```
from sklearn.neighbors import KNeighborsClassifier
model_knn=KNeighborsClassifier(n_neighbors=5)
model_knn.fit(x_train_smote,y_train_smote)
```

Out[56]:
```
KNeighborsClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [57]:
```
y_pred_knn=model_knn.predict(x_test)
y_pred_knn
```

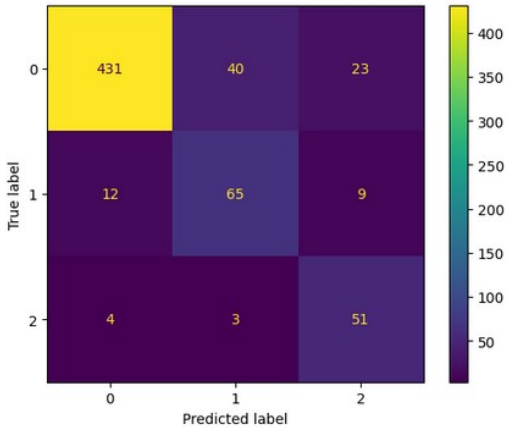
Out[57]:
```
array([1., 1., 1., 1., 2., 1., 1., 1., 1., 2., 3., 1., 1., 1., 1., 1., 1.,
       1., 1., 2., 1., 3., 1., 3., 1., 1., 1., 2., 1., 2., 1., 1., 1., 1.,
       1., 1., 2., 1., 2., 1., 1., 3., 1., 1., 2., 1., 1., 1., 1., 1., 1.,
       3., 2., 1., 1., 1., 1., 1., 1., 1., 1., 1., 2., 1., 2., 1., 2.,
```

In [59]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [60]:
```
print('Testing accuracy: ',accuracy_score(y_test,y_pred_knn))
print('Training accuracy: ',accuracy_score(y_train_smote,y_pred_train_knn))
```

```
Testing accuracy:  0.8573667711598746
Training accuracy:  0.9623887453344818
```

In [62]:
```
from sklearn.metrics._plot.confusion_matrix import ConfusionMatrixDisplay
# print("Accuracy of KNeighborsClassifier: ",KNN_model.score(x_test,y_test))
cm=confusion_matrix(y_test,y_pred_knn)
cm_display=ConfusionMatrixDisplay(cm).plot()
```



Activity 4.4: GradientBoosting

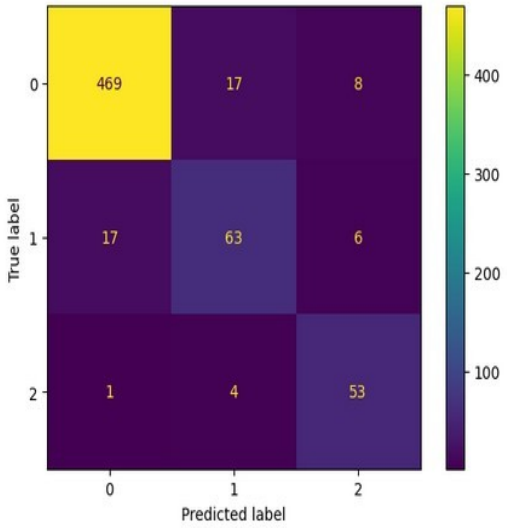
In [90]:
```
from sklearn.ensemble import GradientBoostingClassifier
gboost_model = GradientBoostingClassifier(n_estimators = 150, max_depth = 10, random_state = 10)
gboost_model.fit(x_train_smote, y_train_smote)
```

Out[90]:
```
GradientBoostingClassifier(max_depth=10, n_estimators=150, random_state=10)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [91]:
```
y_pred_gb=gboost_model.predict(x_test)
y_pred_train_gb=gboost_model.predict(x_train_smote)
print(accuracy_score(y_test,y_pred_gb))
print(accuracy_score(y_train_smote,y_pred_train_gb))
```

```
0.9169278996865203
0.9994257823715188
```

In [92]:
```
# print("Accuracy of GradientBoostingClassifier: ",gboost_model.score(x_test,y_test))
cm=confusion_matrix(y_test,y_pred_gb)
cm_display=ConfusionMatrixDisplay(cm).plot()
plt.show()
```



Activity 4.5: Support Vector Classifier

In [95]:
```
from sklearn.svm import SVC
```

In [96]:
```
model_svc=SVC()
```

In [97]:
```
model_svc.fit(x_train_smote,y_train_smote)
```

Out[97]:
```
SVC()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [98]:
```
y_pred_svc=model_svc.predict(x_test)
```

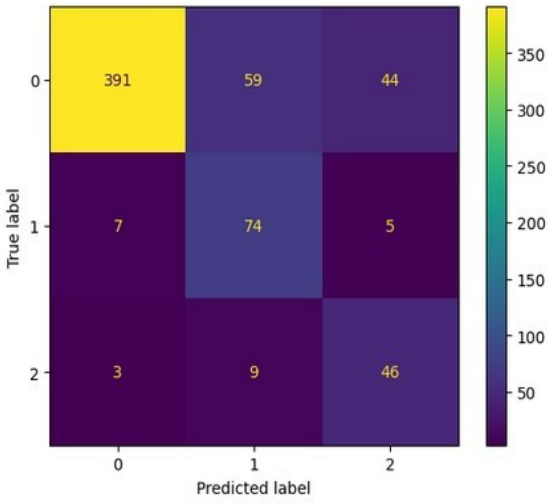
In [99]:
```
y_pred_svc
```

Out[99]:
```
array([1., 1., 1., 2., 2., 1., 3., 1., 1., 2., 3., 1., 1., 1., 3., 1., 3.,
       1., 1., 2., 1., 1., 1., 3., 1., 1., 1., 2., 1., 2., 1., 1., 3., 1.,
```

In [101]:
```
print('Testing_accuracy= ',accuracy_score(y_test,y_pred_svc))
print('Training_accuracy= ',accuracy_score(y_train_smote,y_pred_train_svc))
```

```
Testing_accuracy=  0.8009404388714734
Training_accuracy=  0.8507034165948895
```

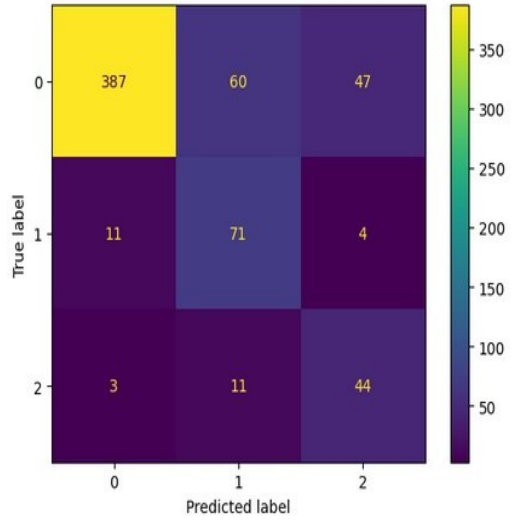
In [102]:
```
cm=confusion_matrix(y_test,y_pred_svc)
cm_display=ConfusionMatrixDisplay(cm).plot()
plt.show()
```



Activity 4.6: GridSearch CV(hyper parameter tuning)

```
In [105]: from sklearn.model_selection import GridSearchCV

In [106]: parameters={'kernel':['linear','rbf'],'C':[0.1,0.2,0.5,1.0],'gamma':['scale','auto']}

In [107]: clf=GridSearchCV(model_svc,param_grid=parameters,verbose=2)

In [108]: clf.fit(x_train_smote,y_train_smote)
```

```
In [115]: print('Testing accuracy: ',accuracy_score(y_test,y_pred_gscv))
          print('Training accuracy: ',accuracy_score(y_train_smote,y_pred_train_gscv))

          Testing accuracy:  0.786833855799373
          Training accuracy:  0.7967269595176572

In [116]: cm=confusion_matrix(y_test,y_pred_gscv)
          cm_display=ConfusionMatrixDisplay(cm).plot()
          plt.show()
```



Activity 4.7: CNN

In [131]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, GRU, Dense , Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GRU, Dense, Flatten , Reshape
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.optimizers import Adam
from keras.regularizers import l2
from keras.regularizers import l1
# Assuming the input data has the shape (num_samples, num_timesteps, num_features)
# In your case, num_timesteps will be 1 as you have 5 features for each sample.


num_timesteps = X_train.shape[1]



# Assuming the input data has the shape (num_samples, num_timesteps, num_features)
# In your case, num_timesteps will be 1 as you have 5 features for each sample.

models = Sequential([
    Conv1D(32, 3,  activation=LeakyReLU(alpha=0.01)   , kernel_regularizer=l1(0.01) , padding='same',     input_shape=
    MaxPooling1D(2, strides=2),
    Conv1D(64, 3, activation=LeakyReLU(alpha=0.01), padding='same'),
    MaxPooling1D(2, strides=2),
    Flatten(),  # Flatten the output of the Conv1D layers
       Reshape((-1, 64)),
    GRU(64  , dropout=0.4),
    Dense(64, activation=LeakyReLU(alpha=0.01)),
    Dense(3, activation='softmax')  # Output layer with 3 target labels
])


models.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])


models.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 32)             128

 max_pooling1d (MaxPooling1  (None, 4, 32)             0
 D)

 conv1d_1 (Conv1D)           (None, 4, 64)             6208

 max_pooling1d_1 (MaxPoolin  (None, 2, 64)             0
 g1D)

 flatten (Flatten)           (None, 128)               0

 reshape (Reshape)           (None, 2, 64)             0

 gru (GRU)                   (None, 64)                24960

 dense (Dense)               (None, 64)                4160

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 35651 (139.26 KB)
Trainable params: 35651 (139.26 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [132]:
```
# Train the model
# batch_size = 64
# epochs = 20

statics=models.fit(X_train, Y_train, batch_size=64, epochs=20 , validation_split=0.2)
```

```
Epoch 1/20
44/44 [==============================] - 4s 23ms/step - loss: 1.1101 - accuracy: 0.5029 - val_loss: 1.7896 - val_accu
racy: 0.0000e+00
Epoch 2/20
44/44 [==============================] - 0s 9ms/step - loss: 0.8545 - accuracy: 0.7046 - val_loss: 1.3125 - val_accur
acy: 0.3702
Epoch 3/20
44/44 [==============================] - 0s 9ms/step - loss: 0.6495 - accuracy: 0.7836 - val_loss: 1.0277 - val_accur
acy: 0.5882
Epoch 4/20
44/44 [==============================] - 0s 9ms/step - loss: 0.6109 - accuracy: 0.7940 - val_loss: 1.2076 - val_accur
acy: 0.5122
Epoch 5/20
44/44 [==============================] - 0s 9ms/step - loss: 0.5857 - accuracy: 0.8073 - val_loss: 1.1203 - val_accur
acy: 0.5567
Epoch 6/20
44/44 [==============================] - 0s 11ms/step - loss: 0.5821 - accuracy: 0.8087 - val_loss: 1.1605 - val_accu
racy: 0.5208
Epoch 7/20
44/44 [==============================] - 1s 14ms/step - loss: 0.5694 - accuracy: 0.8169 - val_loss: 0.7603 - val_accu
```

In [133]:
```
# Evaluate the model on the test set
loss, accuracy_cnn = models.evaluate(x_test, y_test_cnn)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy_cnn)
```

```
20/20 [==============================] - 0s 4ms/step - loss: 0.4996 - accuracy: 0.8260
Test Loss: 0.49955734610557556
Test Accuracy: 0.8260188102722168
```

Activity 5: Testing the model

```
[87] model_rf.predict([[0.006,0.0,60.0,2.3,15.0,133.0,151.0,23.0]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
array([2.])
```

```
[88] model_rf.predict([[0.000,0.0,78.0,0.0,3.0,100.0,123.0,1.0]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
array([2.])
```

```
[89] model_rf.predict([[0.006,0.0,47.0,5,2.2,141.0,138.0,13.0]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
array([2.])
```

Milestone 5: Performance Testing

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Compare the model

```
In [153]: performance=pd.DataFrame()
          performance['model_name']=model_name
          performance['accuracy_score_of_model']=accuracy_score_of_model
```

```
In [154]: performance
```

Out[154]:

	model_name	accuracy_score_of_model
0	KNN	0.849530
1	Decision Tree Classifier	0.786834
2	Random Forest Classifier	0.929467
3	Gradient Boosting Classifier	0.912226
4	Support Vector Classifier	0.808777
5	GridSearchCV(Hyper Parameter Tuning)	0.789969
6	CNN	0.833856

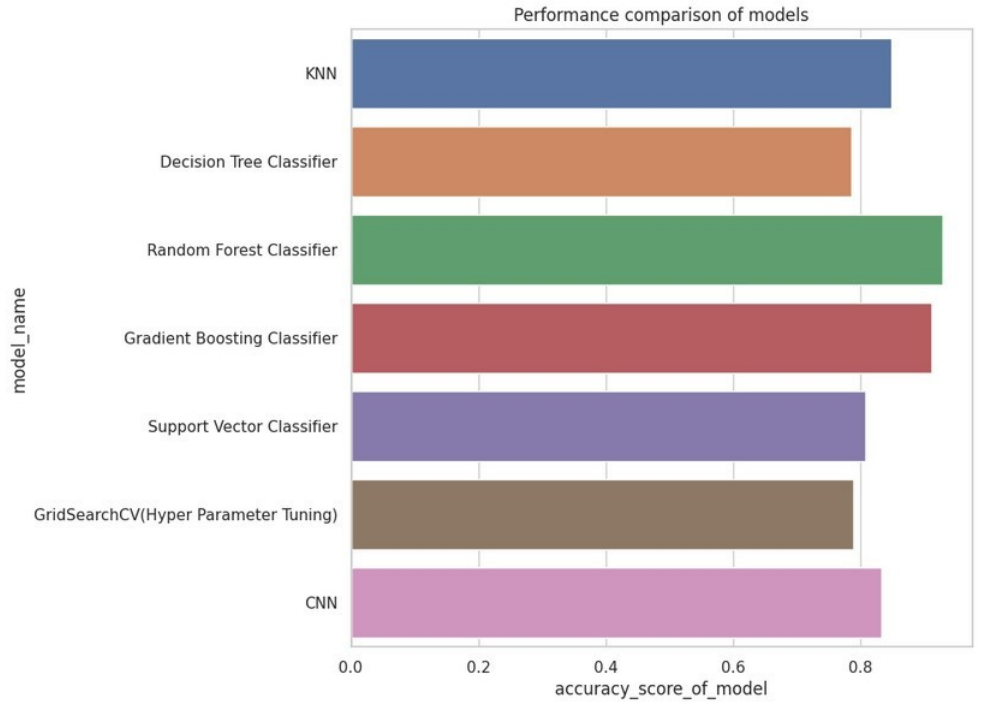
Activity 1.1: Adding colors to the dataframe

```
[142] CM=sns.light_palette('red',as_cmap=True)
      C=performance.style.background_gradient(cmap=CM)
      C
```

	model_name	accuracy_score_of_model
0	KNN	0.851097
1	Decision Tree Classifier	0.797806
2	Random Forest Classifier	0.913793
3	Gradient Boosting Classifier	0.907524
4	Support Vector Classifier	0.800940
5	GridSearchCV(Hyper Parameter Tuning)	0.786834
6	CNN	0.833856

Activity 2: Bar plot for model performance

```
In [155]: plt.figure(figsize=(8,8))
          sns.set(style="whitegrid")
          sns.barplot(x='accuracy_score_of_model',y='model_name',data=performance)
          plt.title('Performance comparison of models')
          plt.show()
```



After comparing the model with the help of bar plot. We came to a conclusion that Random Forest is showing the highest accuracy and is performing well.

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle
pickle.dump(RF_model ,open("fetal_health12.pkl"  , 'wb'))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI. This section has the following tasks:

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Pages

For this project create two HTML files namely

- index.html
- inspect.html
- outputt.html

and save them in the templates folder.

Activity 2.2: Build Python code

Import the libraries

```
import os
import pickle
import joblib
from flask import Flask, render_template, request, jsonify
import sklearn
print(sklearn.__version__)
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
app = Flask(__name__)



model_path = os.path.abspath("fetal_health12.pkl")
```

Render HTML page:

```
@app.route('/')
def index5():
    return render_template('8 form.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method. Retrieves the value from UI:

```
@app.route('/login', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    features = [data['accelerations'], data['prolongued_decelerations'], data['abnormal_short_term_variability'], data['percentage_of_time_with_abnormal_long_term_vari
                data['mean_value_of_long_term_variability'], data['histogram_mode'], data['histogram_median'], data['histogram_variance']]
    prediction = model_path.predict([features])[0]
    return jsonify(prediction=str(prediction))
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

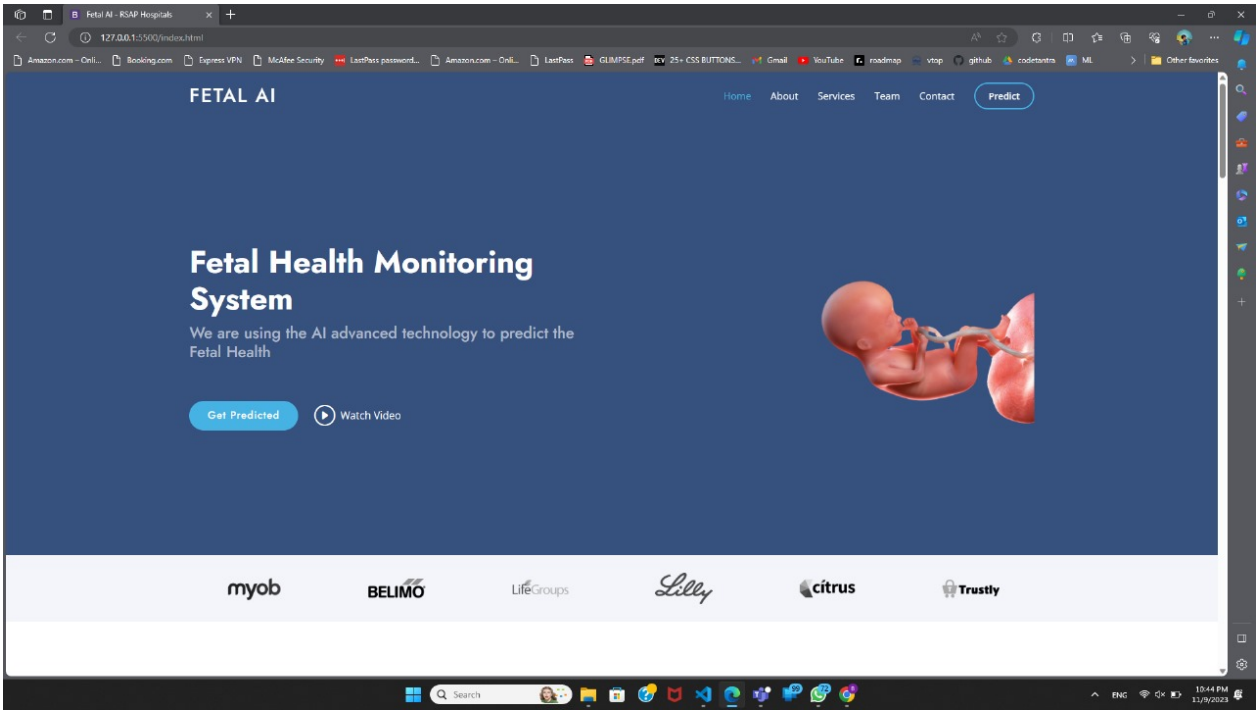
```
if __name__ == '__main__':
    app.run(debug=True)
```

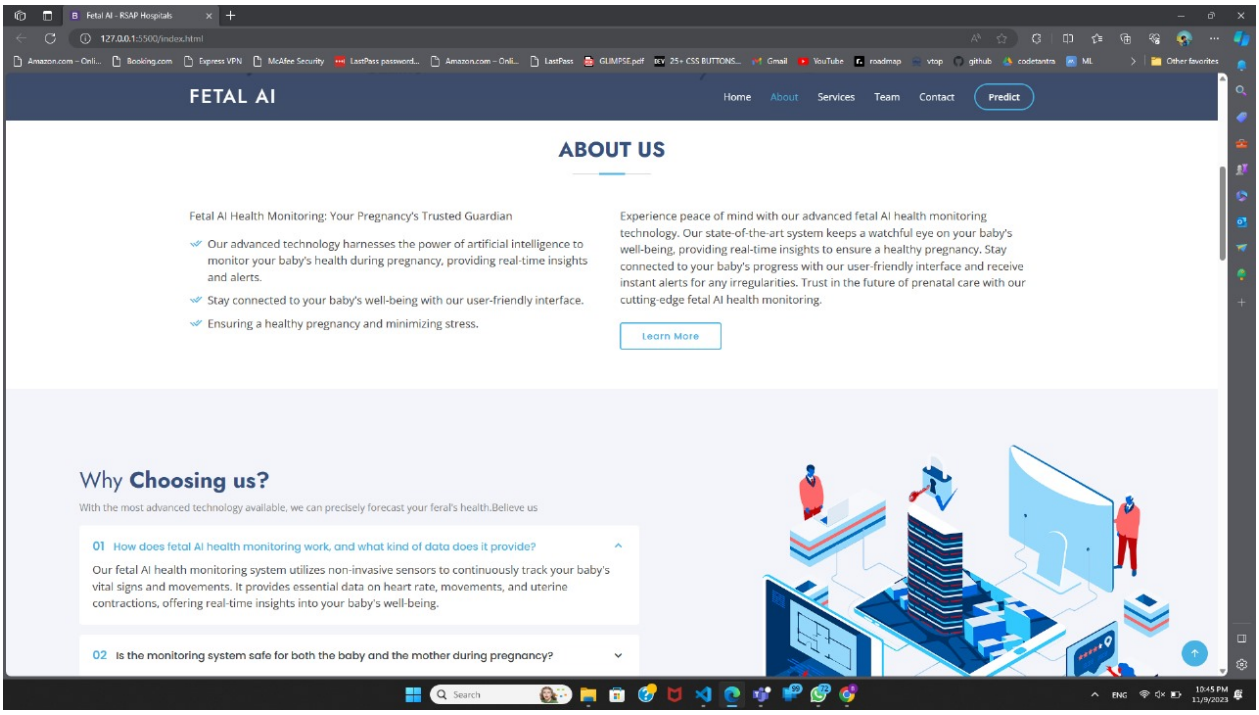
Activity 2.3: Run the web application

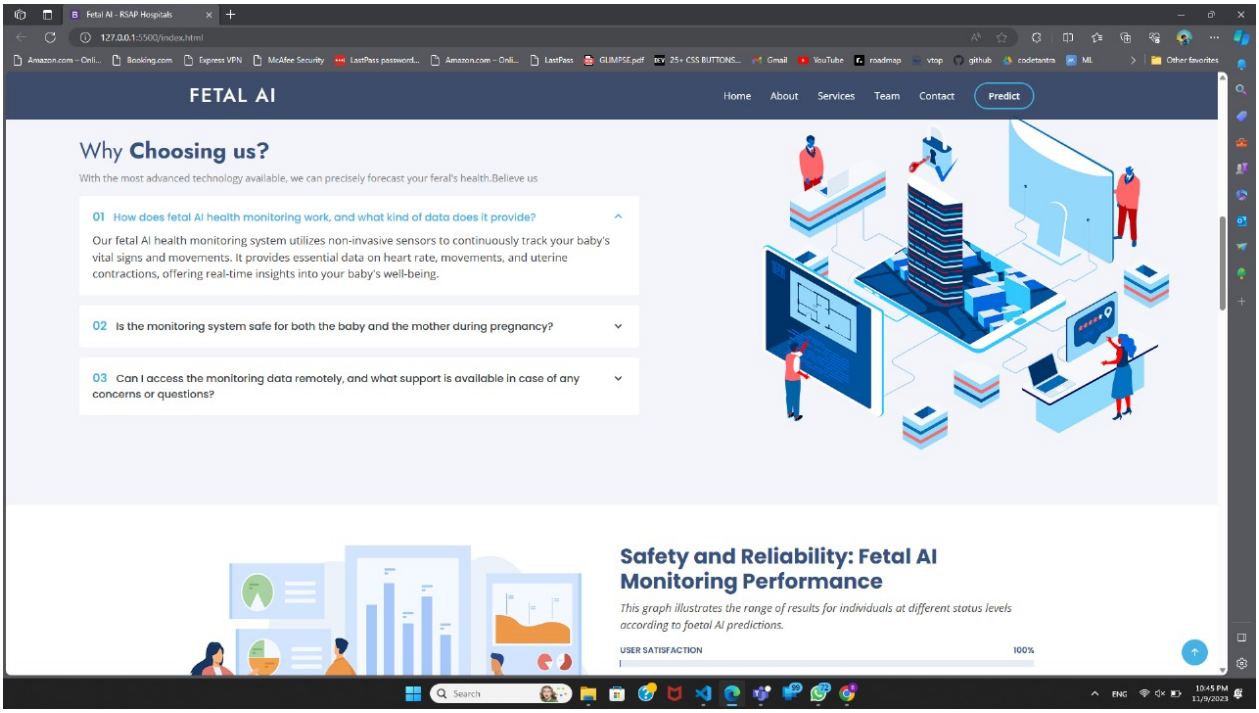
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command ‘
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

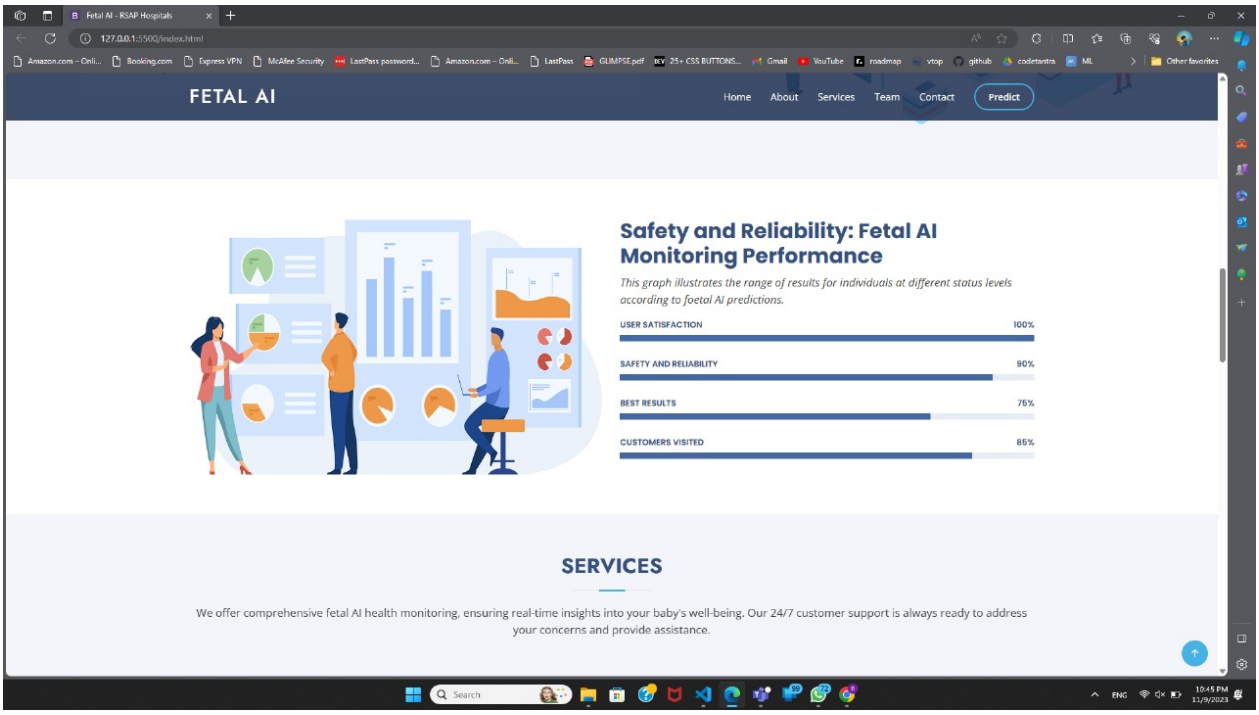
```
PS D:\THANUSH\web development\css\ai ml project> python -u "d:\THANUSH\web development\css\ai ml project\app.py"
1.3.2
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
1.3.2
 * Debugger is active!
 * Debugger PIN: 794-146-930
```

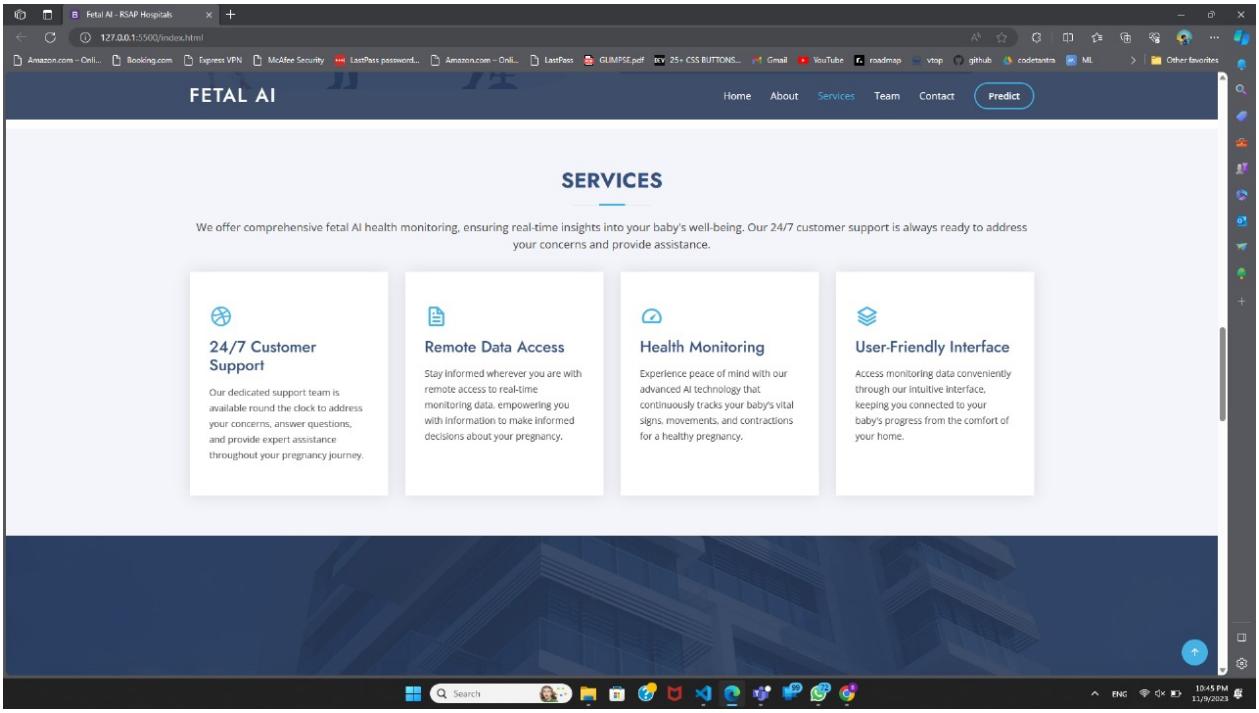
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.

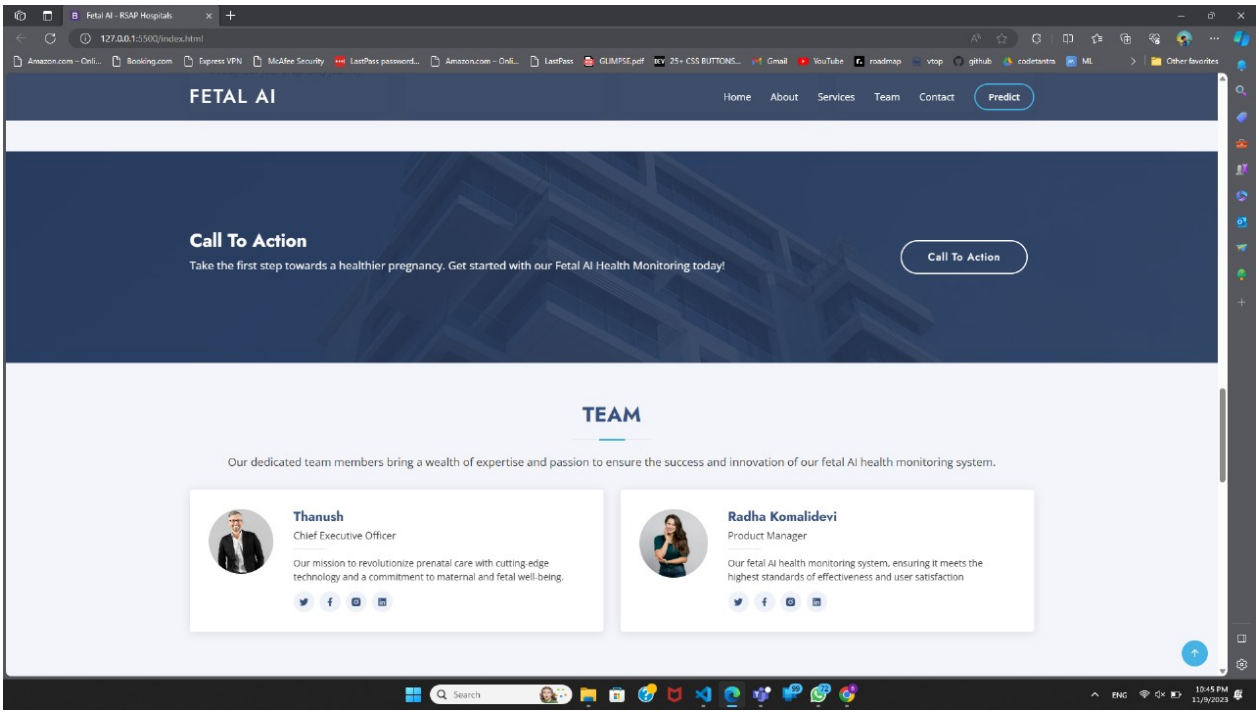

FETAL AI
Home
About
Services
Team
Contact
Predict
Fetal Health Monitoring System
We are using the AI advanced technology to predict the Fetal Health
Get Predicted
Watch Video
myob
BELIMO
LifeGroups
Lilly
citrus
Trustly

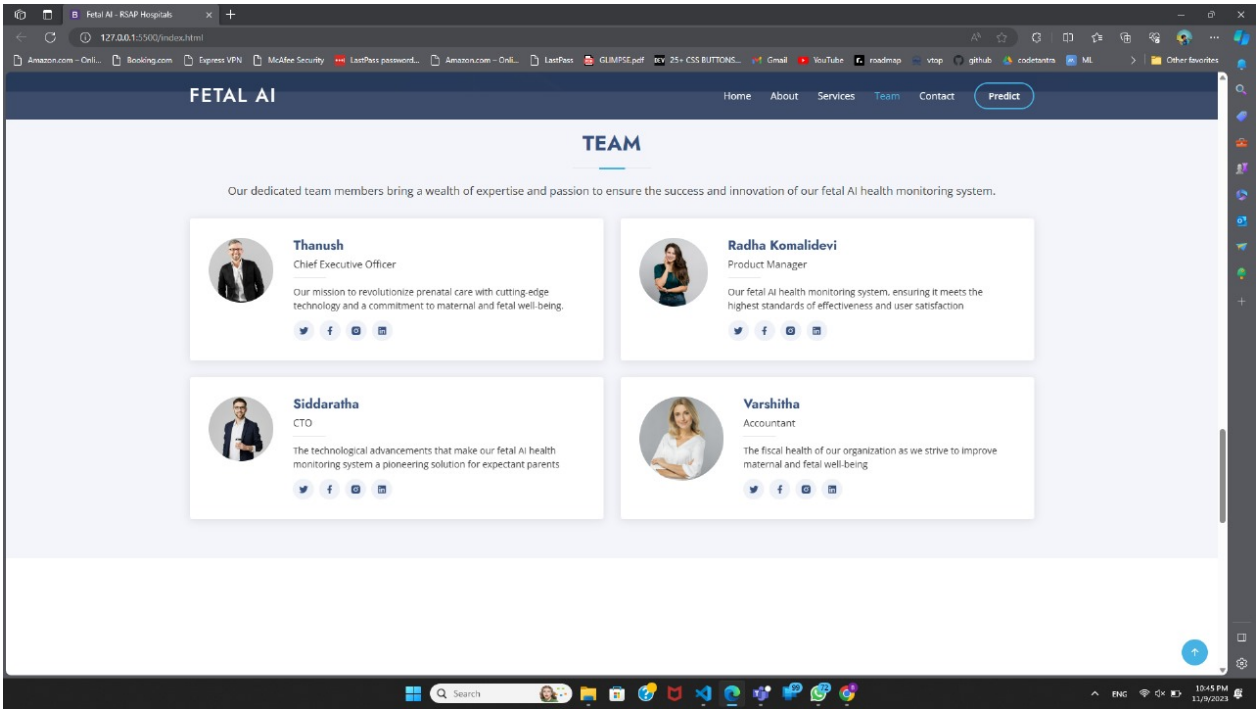

FETAL AI
Home
About
Services
Team
Contact
Predict
ABOUT US
Fetal AI Health Monitoring: Your Pregnancy's Trusted Guardian
Our advanced technology harnesses the power of artificial intelligence to monitor your baby's health during pregnancy, providing real-time insights and alerts.
Stay connected to your baby's well-being with our user-friendly interface.
Ensuring a healthy pregnancy and minimizing stress.
Experience peace of mind with our advanced fetal AI health monitoring technology. Our state-of-the-art system keeps a watchful eye on your baby's well-being, providing real-time insights to ensure a healthy pregnancy. Stay connected to your baby's progress with our user-friendly interface and receive instant alerts for any irregularities. Trust in the future of prenatal care with our cutting-edge fetal AI health monitoring.
Learn More
Why Choosing us?
With the most advanced technology available, we can precisely forecast your feral's health.Believe us
01 How does fetal AI health monitoring work, and what kind of data does it provide?
Our fetal AI health monitoring system utilizes non-invasive sensors to continuously track your baby's vital signs and movements. It provides essential data on heart rate, movements, and uterine contractions, offering real-time insights into your baby's well-being.
02 Is the monitoring system safe for both the baby and the mother during pregnancy?

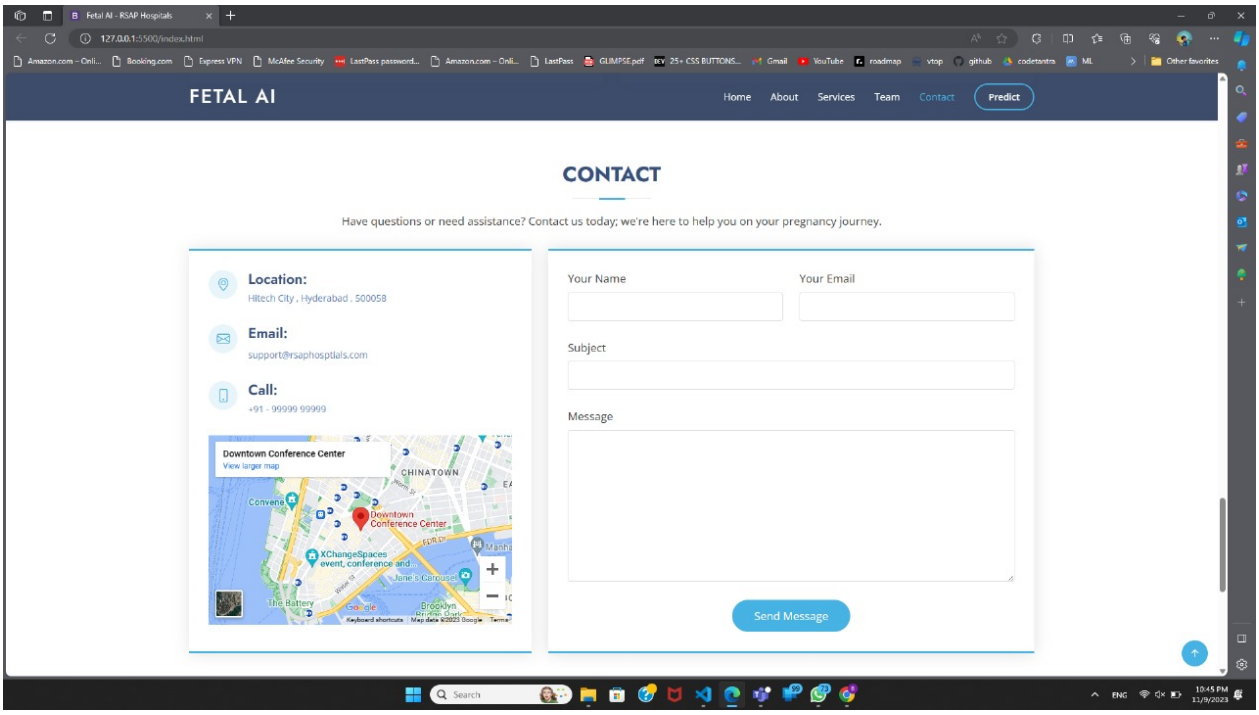

FETAL AI
Home
About
Services
Team
Contact
Predict
Why Choosing us?
With the most advanced technology available, we can precisely forecast your feral's health.Believe us
01 How does fetal AI health monitoring work, and what kind of data does it provide?
Our fetal AI health monitoring system utilizes non-invasive sensors to continuously track your baby's vital signs and movements. It provides essential data on heart rate, movements, and uterine contractions, offering real-time insights into your baby's well-being.
02 Is the monitoring system safe for both the baby and the mother during pregnancy?
03 Can I access the monitoring data remotely, and what support is available in case of any concerns or questions?
Safety and Reliability: Fetal AI Monitoring Performance
This graph illustrates the range of results for individuals at different status levels according to foetal AI predictions.
USER SATISFACTION
100%

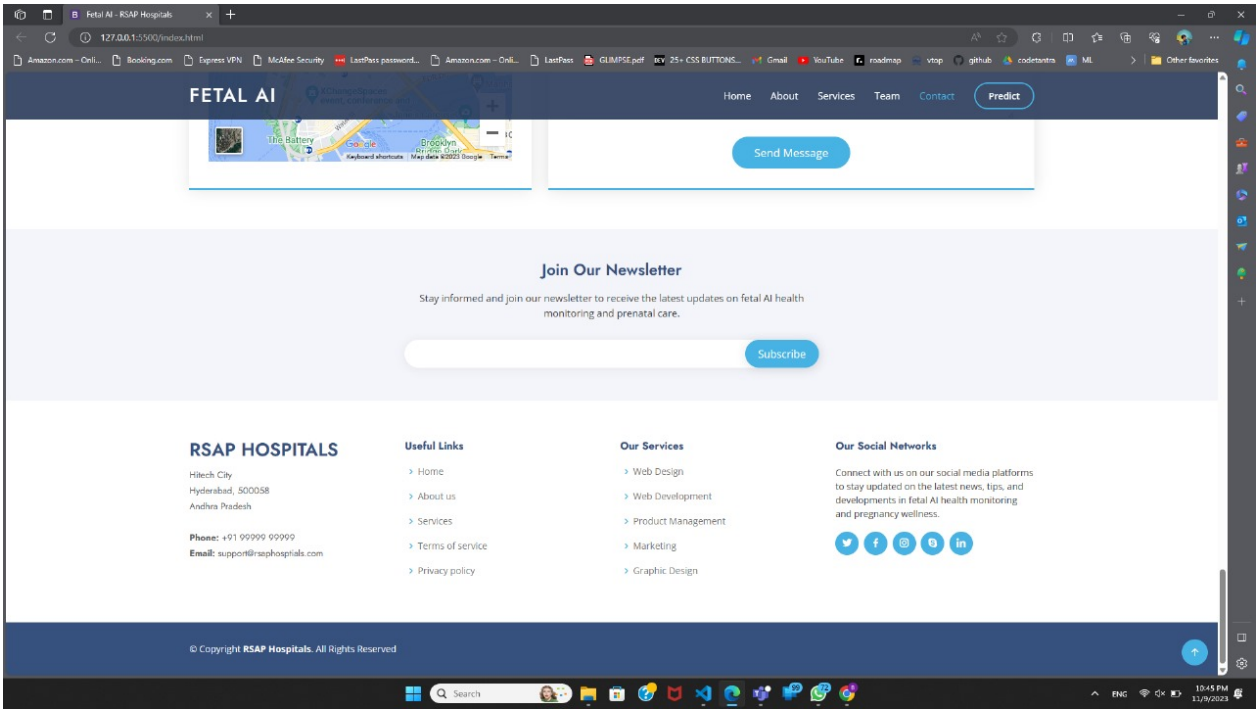

FETAL AI
Home
About
Services
Team
Contact
Predict
Safety and Reliability: Fetal AI Monitoring Performance
This graph illustrates the range of results for individuals at different status levels according to foetal AI predictions.
USER SATISFACTION
100%
SAFETY AND RELIABILITY
90%
BEST RESULTS
75%
CUSTOMERS VISITED
85%
SERVICES
We offer comprehensive fetal AI health monitoring, ensuring real-time insights into your baby's well-being. Our 24/7 customer support is always ready to address your concerns and provide assistance.

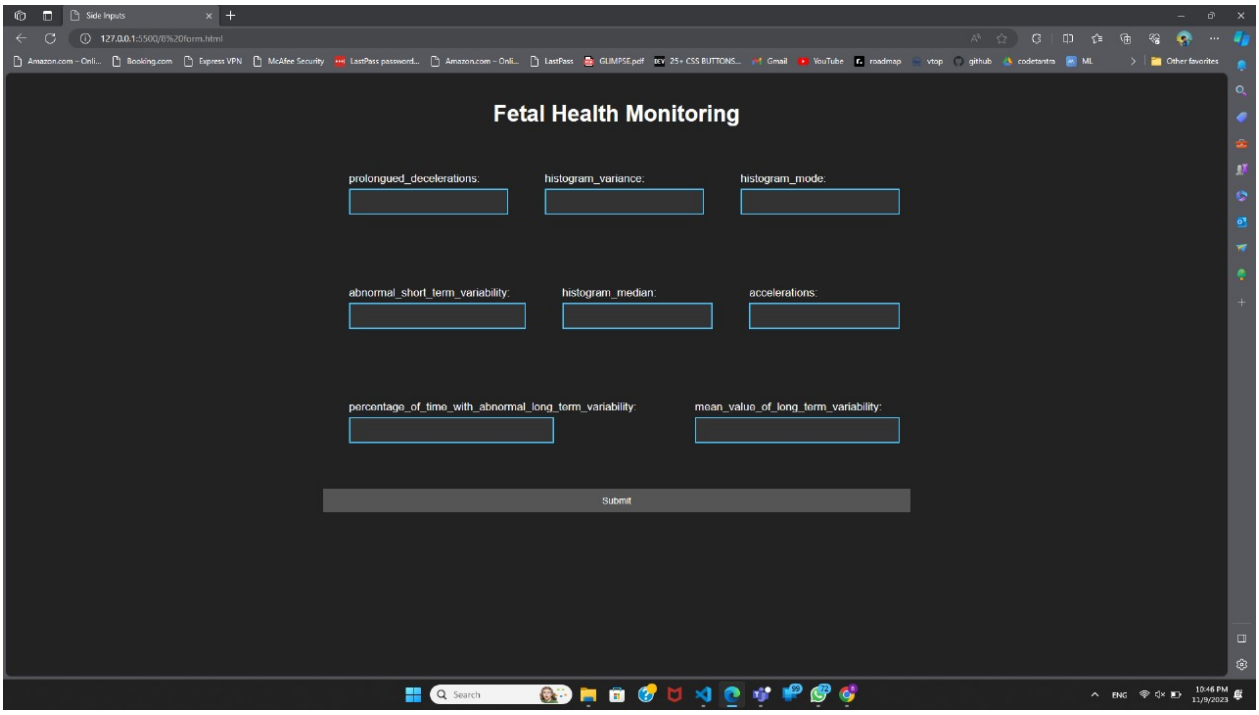

FETAL AI
Home
About
Services
Team
Contact
Predict
SERVICES
We offer comprehensive fetal AI health monitoring, ensuring real-time insights into your baby's well-being. Our 24/7 customer support is always ready to address your concerns and provide assistance.
24/7 Customer Support
Our dedicated support team is available round the clock to address your concerns, answer questions, and provide expert assistance throughout your pregnancy journey.
Remote Data Access
Stay informed wherever you are with remote access to real-time monitoring data, empowering you with information to make informed decisions about your pregnancy.
Health Monitoring
Experience peace of mind with our advanced AI technology that continuously tracks your baby's vital signs, movements, and contractions for a healthy pregnancy.
User-Friendly Interface
Access monitoring data conveniently through our intuitive interface, keeping you connected to your baby's progress from the comfort of your home.


FETAL AI
Home
About
Services
Team
Contact
Predict
Call To Action
Take the first step towards a healthier pregnancy. Get started with our Fetal AI Health Monitoring today!
Call To Action
TEAM
Our dedicated team members bring a wealth of expertise and passion to ensure the success and innovation of our fetal AI health monitoring system.
Thanush
Chief Executive Officer
Our mission to revolutionize prenatal care with cutting-edge technology and a commitment to maternal and fetal well-being.
Radha Komalidevi
Product Manager
Our fetal AI health monitoring system, ensuring it meets the highest standards of effectiveness and user satisfaction


FETAL AI
Home
About
Services
Team
Contact
Predict
TEAM
Our dedicated team members bring a wealth of expertise and passion to ensure the success and innovation of our fetal AI health monitoring system.
Thanush
Chief Executive Officer
Our mission to revolutionize prenatal care with cutting-edge technology and a commitment to maternal and fetal well-being.
Radha Komalidevi
Product Manager
Our fetal AI health monitoring system, ensuring it meets the highest standards of effectiveness and user satisfaction
Siddaratha
CTO
The technological advancements that make our fetal AI health monitoring system a pioneering solution for expectant parents
Varshitha
Accountant
The fiscal health of our organization as we strive to improve maternal and fetal well-being


FETAL AI
Home
About
Services
Team
Contact
Predict
CONTACT
Have questions or need assistance? Contact us today; we're here to help you on your pregnancy journey.
Location:
Hitech City , Hyderabad , 500058
Email:
support@rsaphosptials.com
Call:
+91 - 99999 99999
Downtown Conference Center
View larger map
CHINATOWN
Your Name
Your Email
Subject
Message
Send Message





Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution
Activity 2:- Project Documentation-Step by step project development procedure