FetalAI: USING MACHINE LEARNING TO PREDICT AND MONITOR FETAL HEALTH

Members:

Mohammad Anees A A

Christin Davis

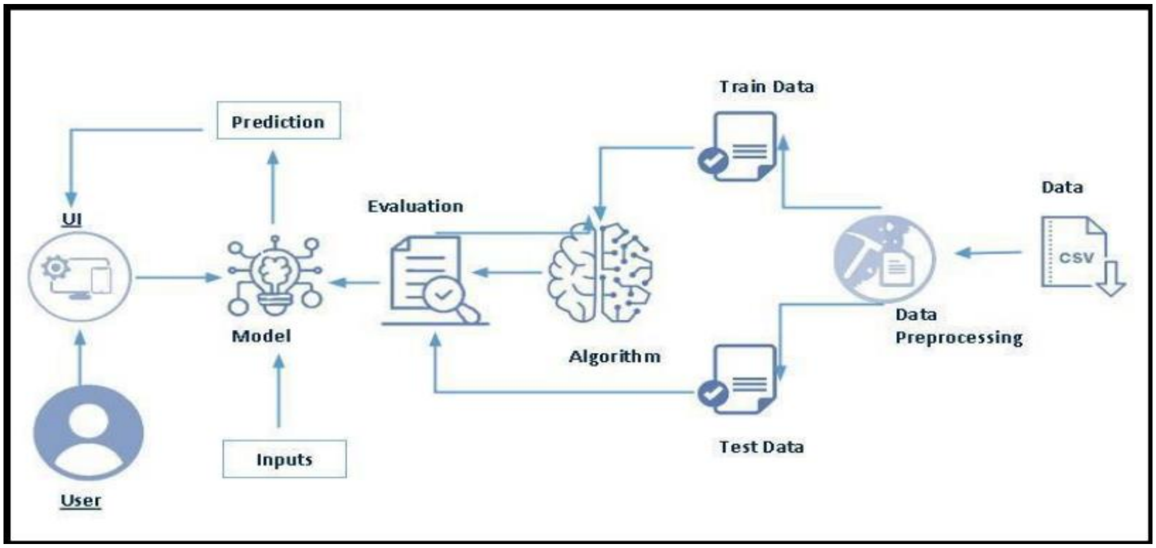
Abhishek Madhuraj

Vaishnav Vijayan

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, Cardiotocography (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

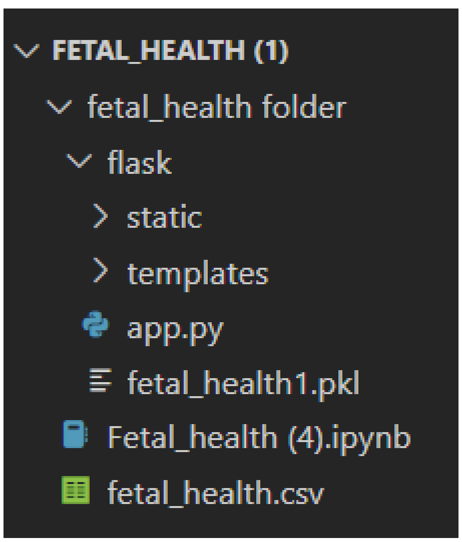
- Create dataframe of model performance
- Bar plot for model performance

6.)Model Deployment

- Save the best model
- Integrate with Web Framework

Project Structure:

Create the Project folder which contains files as shown below:



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

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

A literature survey for a Fetal Health classification project would involve researching and reviewing existing studies, articles, and other publications on the topic of drug classification. The survey would aim to gather information on current classification systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous classification projects, and any relevant data or findings that could inform the design and implementation of the current project.

Activity 4: Social or Business Impact

Social Impact:

- **Promoting Informed Decision-Making:-** By providing accurate and up-to-date information on Fetal Health, can help expectant parents make informed decisions about their pregnancy and childbirth. For example, if a serious health issue is detected in the fetus, parents can decide whether to continue with the pregnancy or consider other options.
- **Reducing Infant Mortality:-** Access to information about fetal health can help expectant parents identify and treat potential health issues before they become life-threatening to the unborn child. This can help reduce the infant mortality rate and ensure that more babies are born healthy.
- **Improving Prenatal Care:-** When expectant parents know about the health of their fetus, they can work with their healthcare provider to create a tailored prenatal care plan that addresses any issues that may be present. This can lead to better outcomes for both the mother and the child.

Business Model/Impact:

- **Improved Patient Outcomes:** By detecting potential health issues in fetuses early, healthcare providers can develop treatment plans that help ensure better outcomes for both the mother and the child. This can lead to improved patient satisfaction and retention rates.
- **Increased Revenue:** Healthcare providers who offer fetal health testing and monitoring services may be able to generate additional revenue streams from expectant parents who are willing to pay for these services. Additionally, if a health issue is detected in the fetus, additional tests, procedures, and treatments may be required, which can generate additional revenue for the healthcare provider.
- **Research and Development:** Knowing about fetal health can also drive research and development in the healthcare industry. For example, new diagnostic tests and treatments can be developed based on data from fetal health monitoring and testing.

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.

Activity 1: Collect the 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.

```
In [1]: import numpy as np
        import pandas as pd
        pd.set_option('max_columns', None)

        import matplotlib.pyplot as plt
        %matplotlib inline
        import seaborn as sns
        sns.set_style('darkgrid')

        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import StandardScaler
        from imblearn.over_sampling import SMOTE

        from sklearn.linear_model import LogisticRegression
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.svm import LinearSVC, SVC
        from sklearn.neural_network import MLPClassifier
        from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import plot_confusion_matrix
        from sklearn.metrics import ConfusionMatrixDisplay

        import warnings
        warnings.filterwarnings(action='ignore')
```

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]: data = pd.read_csv("C:/Users/hp/Downloads/fetal_health.csv")
```

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

Out[6]:

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

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

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 [8]:
```
data.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
```

In [66]:
```
data.describe().T
```

Out[66]:

	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

```
In [67]: data.nunique()

Out[67]: 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
```

```
In [68]: data.isnull().any()

         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
```

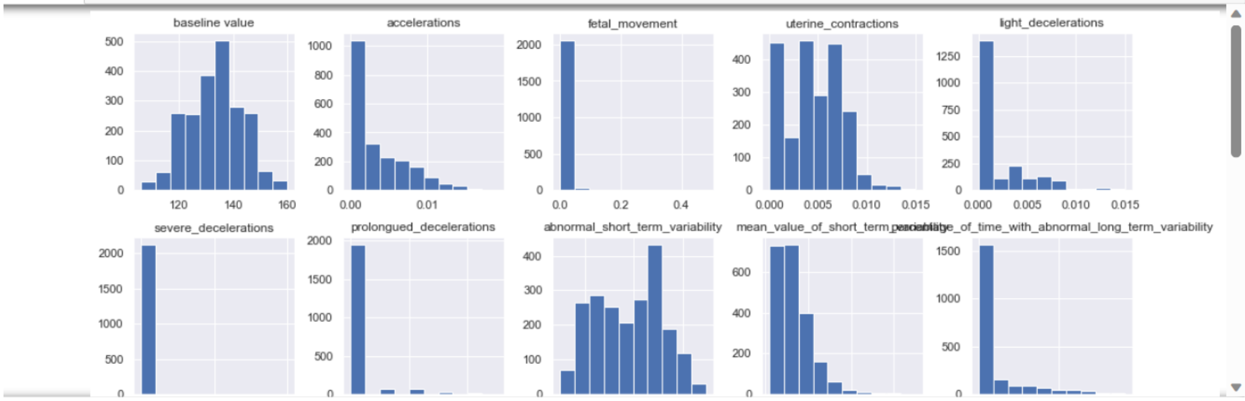
Activity 2: Visual analysis

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.

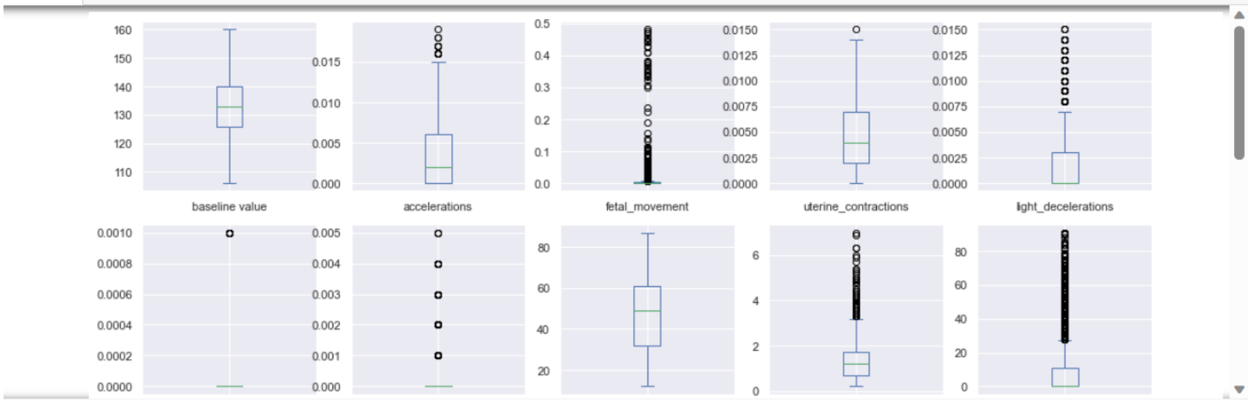
Activity 2.1: Univariate analysis

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 [70]: data.hist(figsize=(17,17), layout=(5,5), sharex=False);
```



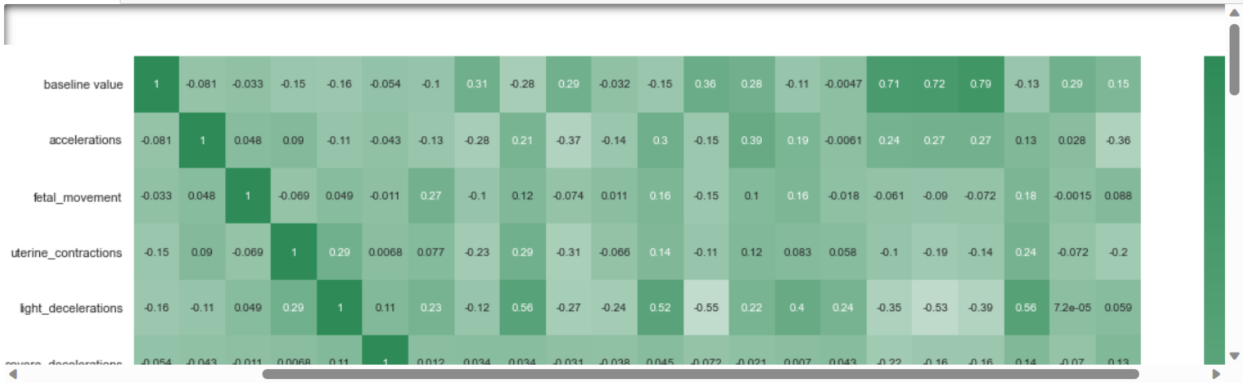

```
In [71]: data.plot(kind='box', figsize=(17,17), layout=(5,5), sharex=False, subplots= True);
```



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 [129]: #correlation matrix
corrmat= data.corr()
plt.figure(figsize=(20,20))

cmap = sns.light_palette("seagreen", as_cmap=True)

sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```



Activity 3: Feature Selection

```
In [130]: data.drop(columns=["histogram_mean"], axis=1, inplace=True)
```

```
In [131]: data.corr()["fetal_health"].sort_values(ascending=False)
```

```
Out[131]: 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
```

In [133]:
```
new_data=data.loc[:,["prolongued_decelerations", "abnormal_short_term_variability", "percentage_of_time_with_abnormal_long_term_v
```

In [134]:
```
new_data.head()
```

Out[134]:

	prolongued_decelerations	abnormal_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	histogram_variance	histogram_median	me
0	0.0	73.0	43.0	73.0	121.0	
1	0.0	17.0	0.0	12.0	140.0	
2	0.0	16.0	0.0	13.0	138.0	
3	0.0	16.0	0.0	13.0	137.0	
4	0.0	16.0	0.0	11.0	138.0	

Activity 4: Scaling the data

In [138]:
```
X = data.drop(columns=['fetal_health'])
y = data["fetal_health"]
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X_scaled = pd.DataFrame(scale.fit_transform(X), columns=X.columns)
X_scaled.head()
```

Out[138]:

	accelerations	prolongued_decelerations	abnormal_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	mean_value_of_long_term_va
0	0.000000	0.0	0.813333	0.472527	0.
1	0.315789	0.0	0.066667	0.000000	0.
2	0.157895	0.0	0.053333	0.000000	0.
3	0.157895	0.0	0.053333	0.000000	0.
4	0.368421	0.0	0.053333	0.000000	0.

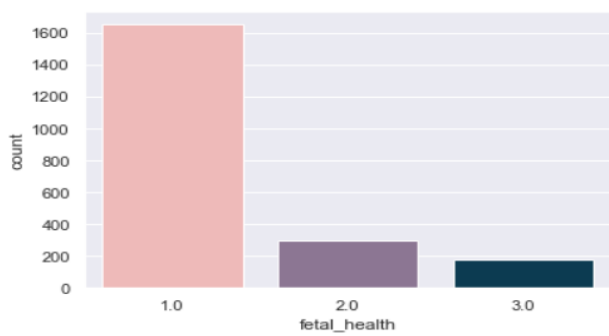
Activity 5: Checking if the dataset is balanced or not

In [139]:
```
#first of all let us evaluate the target and find out if our data is imbalanced or not
data['fetal_health'].value_counts()
```

Out[139]:
```
1.0    1655
2.0     295
3.0     176
Name: fetal_health, dtype: int64
```

In [140]:
```
colours=["#f7b2b0","#8f7198", "#003f5c"]
sns.countplot(data= data, x="fetal_health",palette=colours)
```

Out[140]: <AxesSubplot: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 4: Model Building

Activity 1: Splitting data into train and test

```
In [141]: from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

```
In [142]: from sklearn.model_selection import train_test_split
          X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 10)
          X_train.shape, X_test.shape

Out[142]: ((1488, 8), (638, 8))
```

Activity 2: Applying SMOTE for balancing the data

```
In [146]: pip install imblearn

          Requirement already satisfied: imblearn in c:\users\hp\anaconda3\lib\site-packages (
          Requirement already satisfied: imbalanced-learn in c:\users\hp\anaconda3\lib\site-pa
          Requirement already satisfied: joblib>=1.0.0 in c:\users\hp\anaconda3\lib\site-packa
          0)
          Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\hp\anaconda3\lib\site
          (1.1.2)
          Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\lib\sit
          n) (2.2.0)
          Requirement already satisfied: numpy>=1.17.3 in c:\users\hp\anaconda3\lib\site-packa
          2.4)
          Requirement already satisfied: scipy>=1.3.2 in c:\users\hp\anaconda3\lib\site-packag
          1)

          [notice] A new release of pip available: 22.2 -> 23.1
          [notice] To update, run: python.exe -m pip install --upgrade pip
          Note: you may need to restart the kernel to use updated packages.
```

```
In [147]: from imblearn.over_sampling import SMOTE
          smote = SMOTE()
```

```
In [148]: X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train)
```

```
In [149]: from collections import Counter
          print ("Before SMOTE :" , Counter(y_train))
          print ("After SMOTE :" , Counter(y_train_smote))

          Before SMOTE : Counter({1.0: 1158, 2.0: 201, 3.0: 129})
          After SMOTE : Counter({1.0: 1158, 2.0: 1158, 3.0: 1158})
```

After applying SMOTE, the dataset is balanced. And now we will again train the model after balancing the dataset to check the accuracy.

Activity 3: Training the model after applying SMOTE

```
In [151]: RF_model.fit(X_train_smote, y_train_smote)
          predictions=RF_model.predict(X_test)
          print(accuracy_score(y_test,predictions))
          pd.crosstab(y_test, predictions)
```

```
0.95141065830721
```

Out[151]:

col_0	1.0	2.0	3.0
fetal_health			
1.0	482	11	4
2.0	12	80	2
3.0	1	1	45

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

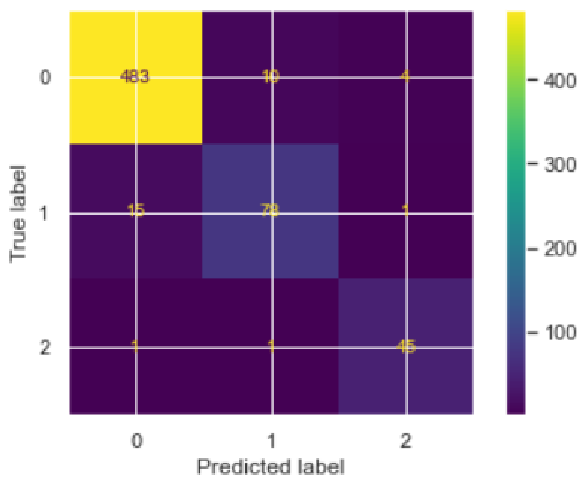
A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new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

In [153]:
```
RF_model = RandomForestClassifier()
RF_model.fit(X_train_smote, y_train_smote)
predictions=RF_model.predict(X_test)
print(accuracy_score(y_test,predictions))
```

```
0.9498432601880877
```

In [155]:
```
print("For the amounts of training data is: ",size)
print("Accuracy of RandomForestClassifier: ",RF_model.score(X_test,y_test))
cm = confusion_matrix(y_test, predictions)
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  3474
Accuracy of RandomForestClassifier:  0.9498432601880877
```

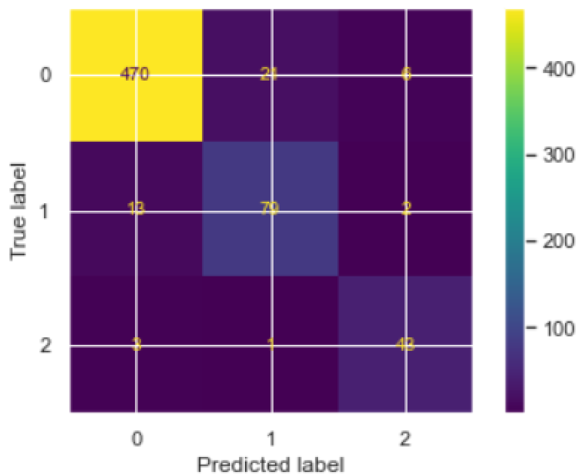


Activity 4.2: Decision Tree

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 [158]: DT_model = DecisionTreeClassifier()
          DT_model.fit(X_train_smote, y_train_smote)
          predictions = DT_model.predict(X_test)
          print(accuracy_score(y_test,predictions))

          0.9278996865203761
```

```
In [159]: print("For the amounts of training data is: ",size)
          print("Accuracy of DecisionTreeClassifier: ",DT_model.score(X_test,y_test))
          cm = confusion_matrix(y_test, predictions)
          cm_display = ConfusionMatrixDisplay(cm).plot()
          plt.show()

          For the amounts of training data is:  3474
          Accuracy of DecisionTreeClassifier:  0.9278996865203761
```



Activity 4.3: Logistic Regression

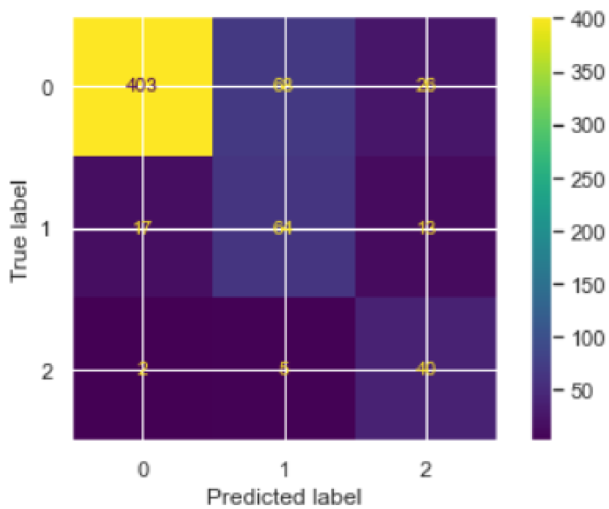
A function named LogisticRegression() is created and train and test data are passed as the parameters. Inside the function, LogisticRegression algorithm is initialized and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new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

In [156]:
```python
LR_model = LogisticRegression()
LR_model.fit(X_train_smote, y_train_smote)
predictions = LR_model.predict(X_test)
print(accuracy_score(y_test,predictions))
```

```
0.7946708463949843
```

In [157]:
```python
print("For the amounts of training data is: ",size)
print("Accuracy of LogisticRegression: ",LR_model.score(X_test,y_test))
cm = confusion_matrix(y_test, predictions)
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  3474
Accuracy of LogisticRegression:  0.7946708463949843
```

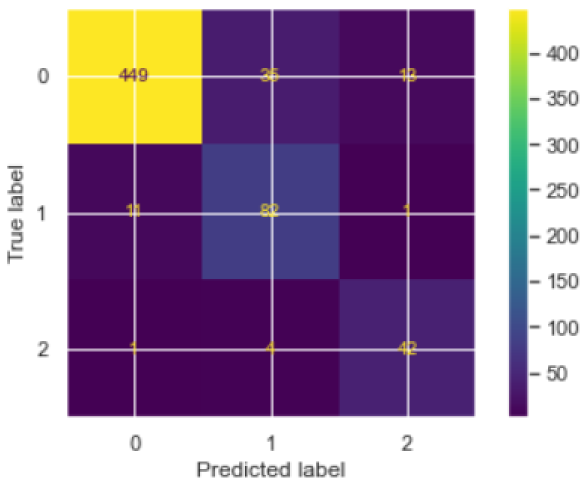


Activity 4.4: K-Nearest Neighbors

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 [160]: KNN_model = KNeighborsClassifier(n_neighbors=5)
          KNN_model.fit(X_train_smote, y_train_smote)
          predictions = KNN_model.predict(X_test)
          print(accuracy_score(y_test,predictions))

          0.8981191222570533
```

```
In [161]: print("For the amounts of training data is: ",size)
          print("Accuracy of KNeighborsClassifier: ",KNN_model.score(X_test,y_test))
          cm = confusion_matrix(y_test, predictions)
          cm_display = ConfusionMatrixDisplay(cm).plot()
          plt.show()

          For the amounts of training data is:  3474
          Accuracy of KNeighborsClassifier:  0.8981191222570533
```



Activity 5: Testing the model

```
In [169]: RF_model.predict([[0.345, 0.1225, 23346, 0.23456, 0.987, 2345, 123, 0]])
Out[169]: array([1.])
```

```
In [170]: RF_model.predict([[0.000, 0.0, 73.0, 43.0, 2.4, 73.0, 120.0, 121.0]])
Out[170]: array([2.])
```

Milestone 5: Performance Testing

Activity 1: Create dataframe of model performance

In [165]:
```
df = pd.DataFrame()
df['name'] = names
df['score'] = scores
df
```

Out[165]:

	name	score
0	Random Forest Classifier	0.948276
1	Logistic Regression	0.794671
2	Decision Tree Classifier	0.929467
3	K Neighbors Classifier	0.898119

Activity 1.1: Adding colors to the dataframe

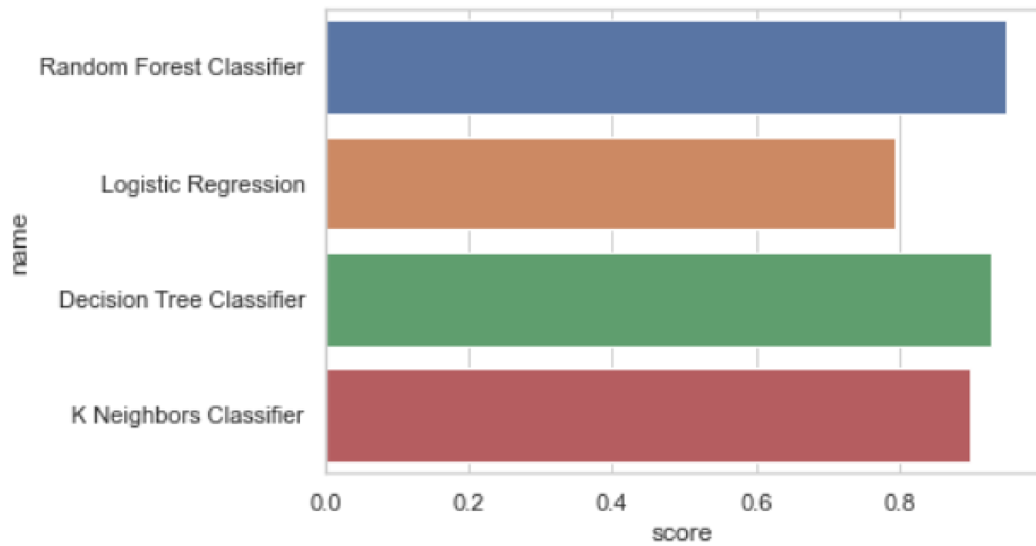
In [166]:
```
CM=sns.light_palette("red", as_cmap=True)
C = df.style.background_gradient(cmap=CM)
C
```

Out[166]:

	name	score
0	Random Forest Classifier	0.948276
1	Logistic Regression	0.794671
2	Decision Tree Classifier	0.929467
3	K Neighbors Classifier	0.898119

Activity 2: Bar plot for model performance

```
In [167]: sns.set(style="whitegrid")
          ax=sns.barplot(y="name", x="score", data=df)
```



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.

```
In [171]: # saving the model

          import pickle
          pickle.dump(RF_model,open('fetal_health1.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 three 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

```
from flask import Flask,request,render_template
import numpy as np
import pandas as pd
import pickle
```

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.

```
model=pickle.load(open(r'fetal_health1.pkl','rb'))
app=Flask(__name__)
```

Render HTML page:

```
@app.route("/")
def f():
    return render_template("index.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("/home", methods=["GET", "POST"])
def home():
    prolongued_decelerations = float(request.form['prolongued_decelerations'])
    abnormal_short_term_variability = float(request.form['abnormal_short_term_variability'])
    percentage_of_time_with_abnormal_long_term_variability = float(request.form['percentage_of_time_w
    histogram_variance = float(request.form['histogram_variance'])
    histogram_median = float(request.form['histogram_median'])
    mean_value_of_long_term_variability = float(request.form['mean_value_of_long_term_variability'])
    histogram_mode = float(request.form['histogram_mode'])
    accelerations = float(request.form['accelerations'])

    X = [[prolongued_decelerations,abnormal_short_term_variability,percentage_of_time_with_abnormal_l

    output = model.predict(X)
    out=['Normal','Pathological','Suspect']
    if int(output[0])==0:
        output='Normal'
    elif int(output[0])== 1:
        output='Pathological'
    else:
        output='Suspect'

    return render_template('output.html',output=output)
```

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:

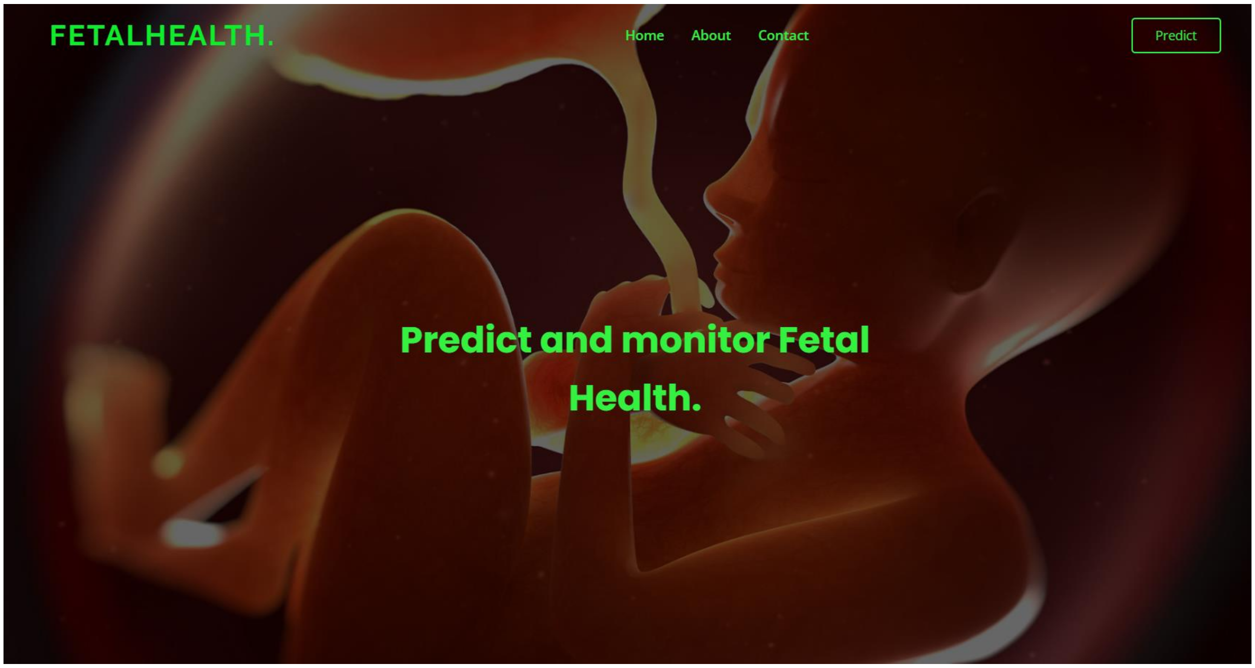
```
if __name__ == "__main__" :
        app.run(debug=True)
```

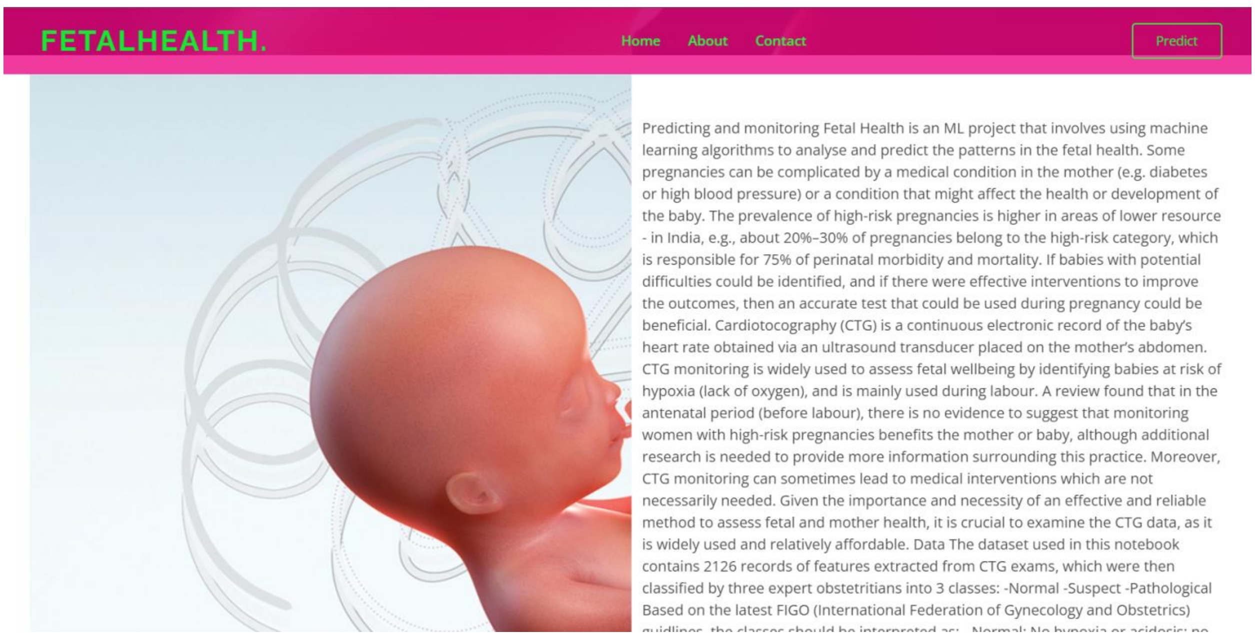
Activity 2.3: Run the web application

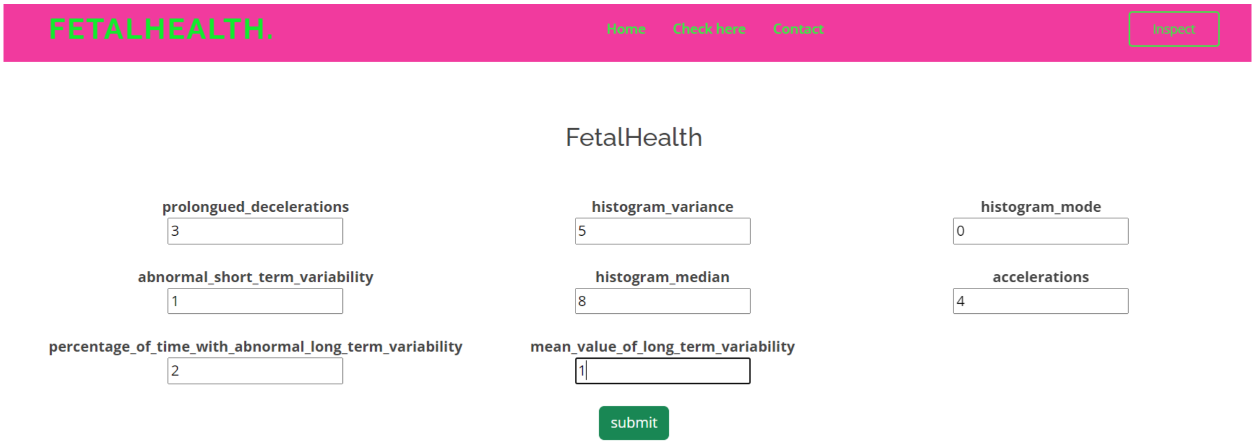
● Open anaconda prompt from the start menu
● Navigate to the folder where your python script is.
● Now type "python app.py" command '
● Navigate to the localhost where you can view your web page.
● Click on the predict button from the top left corner, enter the inputs, click on the submit button, and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

```
(env) C:\Users\hp\Downloads\Fetal_health (7)\fetal_health folder\flask>conda activate env

(env) C:\Users\hp\Downloads\Fetal_health (7)\fetal_health folder\flask>cd C:\Users\hp\Downl

(env) C:\Users\hp\Downloads\Fetal_health (7)\fetal_health folder\flask>python app.py
C:\Users\hp\anaconda3\envs\env\lib\site-packages\sklearn\base.py:318: UserWarning: Trying t
de or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\hp\anaconda3\envs\env\lib\site-packages\sklearn\base.py:318: UserWarning: Trying t
de or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a proc
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
C:\Users\hp\anaconda3\envs\env\lib\site-packages\sklearn\base.py:318: UserWarning: Trying t
de or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
C:\Users\hp\anaconda3\envs\env\lib\site-packages\sklearn\base.py:318: UserWarning: Trying t
de or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 480-081-585
```

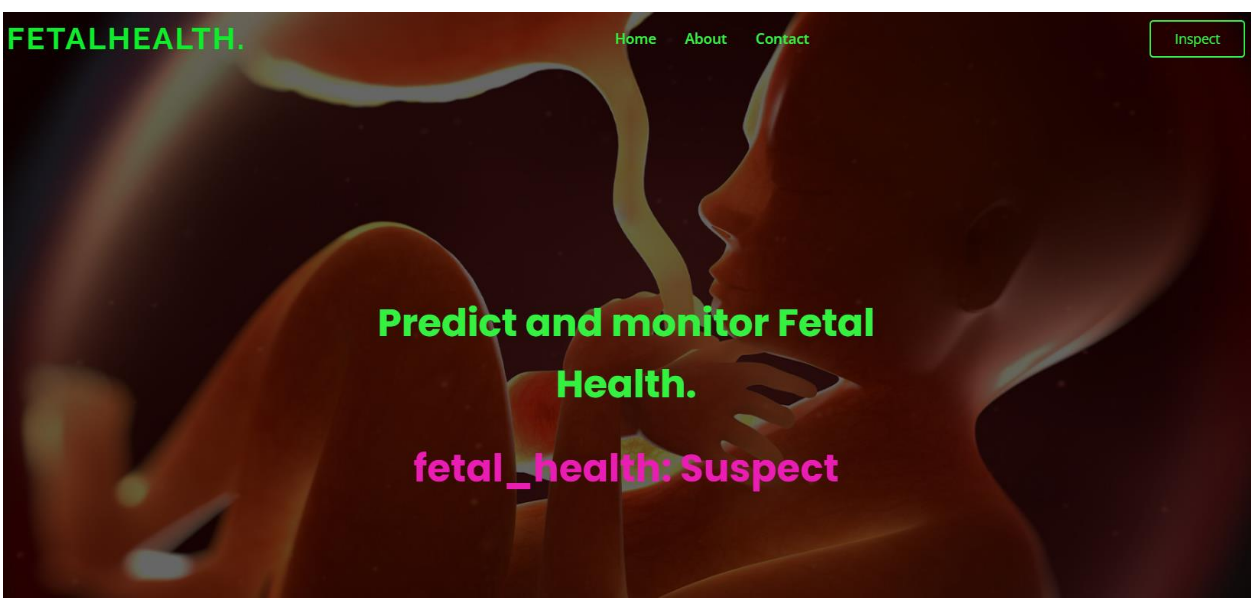
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result.


FETALHEALTH.
Home
About
Contact
Predict
Predict and monitor Fetal Health.


FETALHEALTH.
Home
About
Contact
Predict
Predicting and monitoring Fetal Health is an ML project that involves using machine learning algorithms to analyse and predict the patterns in the fetal health. Some pregnancies can be complicated by a medical condition in the mother (e.g. diabetes or high blood pressure) or a condition that might affect the health or development of the baby. The prevalence of high-risk pregnancies is higher in areas of lower resource - in India, e.g., about 20%–30% of pregnancies belong to the high-risk category, which is responsible for 75% of perinatal morbidity and mortality. If babies with potential difficulties could be identified, and if there were effective interventions to improve the outcomes, then an accurate test that could be used during pregnancy could be beneficial. Cardiotocography (CTG) is a continuous electronic record of the baby's heart rate obtained via an ultrasound transducer placed on the mother's abdomen. CTG monitoring is widely used to assess fetal wellbeing by identifying babies at risk of hypoxia (lack of oxygen), and is mainly used during labour. A review found that in the antenatal period (before labour), there is no evidence to suggest that monitoring women with high-risk pregnancies benefits the mother or baby, although additional research is needed to provide more information surrounding this practice. Moreover, CTG monitoring can sometimes lead to medical interventions which are not necessarily needed. Given the importance and necessity of an effective and reliable method to assess fetal and mother health, it is crucial to examine the CTG data, as it is widely used and relatively affordable. Data The dataset used in this notebook contains 2126 records of features extracted from CTG exams, which were then classified by three expert obstetritians into 3 classes: -Normal -Suspect -Pathological Based on the latest FIGO (International Federation of Gynecology and Obstetrics)


FETALHEALTH.
Home
Check here
Contact
Inspect
FetalHealth
prolongued_decelerations
3
histogram_variance
5
histogram_mode
0
abnormal_short_term_variability
1
histogram_median
8
accelerations
4
percentage_of_time_with_abnormal_long_term_variability
2
mean_value_of_long_term_variability
1
submit


FETALHEALTH.
Home
About
Contact
Inspect
Predict and monitor Fetal Health.
fetal_health: Pathological


FETALHEALTH.
Home
About
Contact
Inspect
Predict and monitor Fetal Health.
fetal_health: Suspect

CONTACT

CONTACT US

Location:

Survey no. 91, Sundarayya Vignana Kendram, Technical Block, 6th floor, Madhava Reddy Colony, Gachibowli, Hyderabad, Telangana 500032

Email:

info@thesmartbridge.com

Call:

+91 6304320044

Your Name

Your Email

Subject

Message

Send Message