Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

FETAL HEALTH CLASSIFIER

Supervised By:

Mr. Kirandeep Singh

Submitted By:

Bhavik Sharma (2210990217)-L

Avleen Kaur (2210990203)

Anushka Malik(2210991323)

Deepanshu (2210990253)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Introduction

Leveraging Machine Learning to Predict Fetal Health from Cardiotocogram Data.

Background:

Child and maternal mortality rates remain significant global health challenges, particularly in low-resource settings. The United Nations has set ambitious goals to reduce preventable deaths among newborns and children under 5 by 2030, emphasizing the importance of accessible and effective healthcare interventions. One crucial aspect of prenatal care is monitoring fetal health, which can be assessed through Cardiotocogram (CTG) exams. These exams provide valuable insights into fetal well-being by analyzing parameters such as fetal heart rate, movements, and uterine contractions.

Objectives:

The primary objective of this project is to develop machine learning models capable of predicting fetal health outcomes based on features extracted from CTG exams. By accurately classifying CTG records into categories such as "Normal," "Suspect," or "Pathological," healthcare providers can intervene proactively to prevent adverse outcomes for both the foetus and the mother. Specifically, the project aims to:

1. Utilize machine learning algorithms, including Support Vector Machines (SVM), Decision Trees, and Logistic Regression, to analyze CTG data and classify fetal health.

2. Evaluate the performance of each algorithm in terms of accuracy, sensitivity, and specificity to identify the most effective model for clinical application.

3. Explore the impact of various features extracted from CTG exams on the predictive capability of the models.

4. Provide a user-friendly interface for healthcare professionals to input CTG data and obtain real-time predictions of fetal health status.

Significance:

- Improve prenatal care and reduce fetal/maternal mortality using AI/ML to analyze CTG data.
- Early detection of fetal health issues.
- Timely interventions to prevent adverse outcomes.
- Enhanced healthcare efficiency in resource-constrained settings.
- Supports UN Sustainable Development Goals related to child and maternal health.
- Potential to save lives and improve family well-being globally.
- Significant concern in resource-constrained regions regarding child and maternal mortality.
- Essential prenatal care interventions for improving health outcomes.
- CTG exams analyze fetal heart rate, movements, and uterine contractions; require advanced analysis techniques.
- Use Support Vector Machines (SVM), Decision Trees, and Logistic Regression.
- Classify CTG data into "Normal," "Suspect," and "Pathological" categories to facilitate early detection of complications.
- Dataset includes 2126 CTG records classified by obstetricians.
- Preprocessing data for model training.
- Choosing appropriate algorithms.
- Assessing model performance with evaluation metrics.
- Enhancing model interpretability through feature importance analysis.
- Aligns with UN Sustainable Development Goals.
- Improves healthcare delivery and outcomes worldwide.

Problem Definition and Requirements

Problem Statement:

The problem at hand revolves around predicting fetal health outcomes from Cardiotocogram (CTG) data, aiming to mitigate the risks associated with adverse fetal and maternal outcomes. The primary challenge lies in accurately classifying CTG records into categories such as "Normal," "Suspect," or "Pathological," based on features extracted from the exams. The ultimate goal is to develop machine learning models capable of providing timely insights into fetal well-being, thereby enabling healthcare providers to intervene proactively and reduce the incidence of fetal and maternal mortality.

Software Requirements:

- Programming Language: Python will be the primary programming language due to its extensive libraries for machine learning and data analysis, including scikit-learn, pandas, and NumPy.
- Development Environment: Anaconda or a similar Python distribution will be used to manage dependencies and create virtual environments.
- Machine Learning Libraries: Scikit-learn will be utilized for building and evaluating machine learning models, while additional libraries such as TensorFlow or PyTorch may be considered for advanced modeling techniques.
- Data Visualization Tools: Matplotlib and Seaborn will be employed for data visualization to gain insights into the dataset and model performance.
- Text Editor or Integrated Development Environment (IDE): Jupyter Notebooks or IDEs like PyCharm will be used for coding and experimentation.
- Version Control: Git and platforms like GitHub will facilitate collaboration, version control, and project management.

Hardware Requirements:

• Processor: A multi-core processor to handle data preprocessing and model training efficiently.

• Memory (RAM): At least 8 GB of RAM to accommodate large datasets and machine learning algorithms.

• Storage: Sufficient storage space for storing datasets, code files, and model artifacts. SSD storage is preferable for faster data access and model training.

• Graphics Processing Unit (GPU) (Optional): While not mandatory, a GPU (NVIDIA GeForce or AMD Radeon) can significantly accelerate model training, especially for deep learning algorithms.

• Operating System: The project can be executed on Windows, macOS, or Linux-based systems, ensuring compatibility across different platforms.

Datasets:

The primary dataset consists of 2126 records of features extracted from CTG exams, expertly classified by obstetricians into three categories: "Normal," "Suspect," or "Pathological." Each record includes features such as baseline fetal heart rate, accelerations, fetal movement, uterine contractions, and variability measures. The dataset is sufficiently large and diverse to train and evaluate machine learning models effectively, ensuring robust performance in real-world scenarios.

Features:

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

- baseline value: Baseline Fetal Heart Rate (FHR) (beats per minute)
- accelerations: Number of accelerations per second
- fetal_movement: Number of fetal movements per second
- uterine_contractions: Number of uterine contractions per second
- light_decelerations: Number of light decelerations (LDs) per second
- severe_decelerations: Number of severe decelerations (SDs) per second
- prolongued_decelerations: Number of prolonged decelerations (PDs) per second

- abnormal_short_term_variability: Percentage of time with abnormal short term variability
- mean_value_of_short_term_variability: Mean value of short term variability
- percentage_of_time_with_abnormal_long_term_variability: Percentage of time with abnormal long term variability
- mean_value_of_long_term_variability: Mean value of long term variability
- fetal_health: Encoded as 1-Normal; 2-Suspect; 3-Pathological. It is our very target column in the dataset.

Proposed Design / Methodology

- **Data Preprocessing:**

Handling Missing Values: Missing values will be imputed using appropriate techniques such as mean, median, or mode imputation.Feature Scaling: Features will be scaled to a standard range (e.g., 0 to 1) to ensure that no feature dominates due to its scale.

Encoding Categorical Variables: Categorical variables will be encoded using techniques like one-hot encoding or label encoding for compatibility with machine learning algorithms.

- **Feature Selection:**

Correlation Analysis: Pearson correlation coefficient will be computed to identify highly correlated features and remove redundant ones.

- **Model Development:**

a. Logistic Regression:

Logistic Regression will be implemented to predict the probability of each class (Normal, Suspect, Pathological) based on the input features. Regularization techniques like L1 (Lasso) or L2 (Ridge) regularization may be applied to prevent overfitting.

b. Decision Trees:

Decision Trees will be constructed to partition the feature space into regions that best discriminate between the classes.Techniques like pruning and setting minimum samples per leaf node will be applied to avoid overfitting.

c. SVM (Support **Vector Machine):**

SVM will be utilized to find the hyperplane that best separates the classes while maximizing the margin.Kernel tricks such as linear, polynomial, or radial basis function (RBF) kernels will be explored to handle non-linear decision boundaries.

- **Model Evaluation:**

The models will be evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix to assess their performance across different classes.Cross-validation techniques like k-fold cross-validation will be employed to ensure robustness and avoid overfitting

- **Hyperparameter Tuning**:

Grid search or random search will be conducted to optimize hyperparameters for each algorithm, such as regularization strength in logistic regression and kernel parameters in SVM.

- **Model Interpretation:**

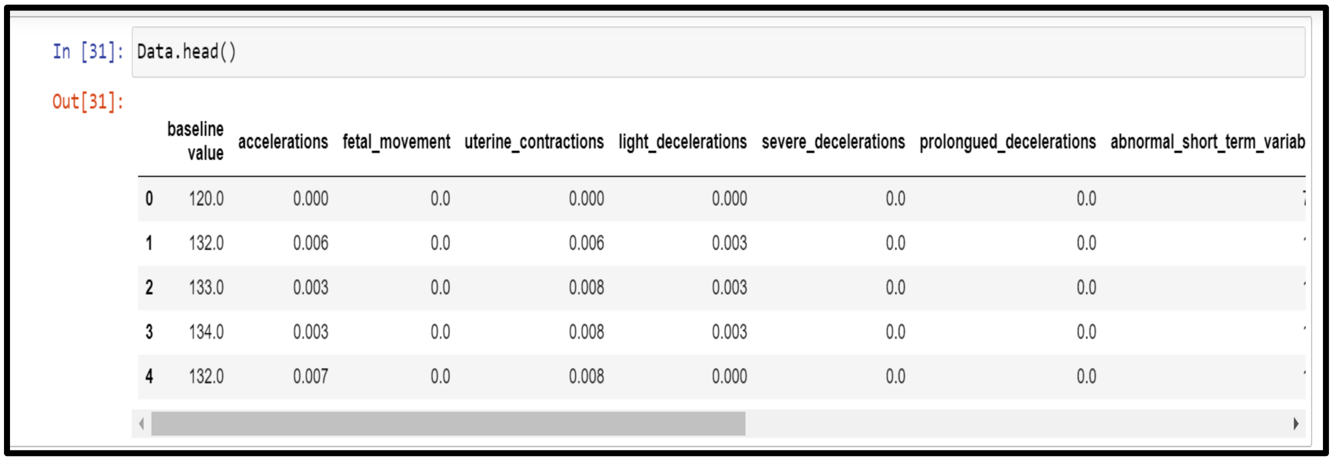
The trained models' decision boundaries and feature importance will be visualized to provide insights into their behavior and aid in model interpretation.

- **Implementation:**

The proposed design and methodology will be implemented using Python programming language and relevant libraries such as scikit-learn, pandas, and matplotlib.Jupyter Notebooks will facilitate code development, experimentation, and documentation.

Results

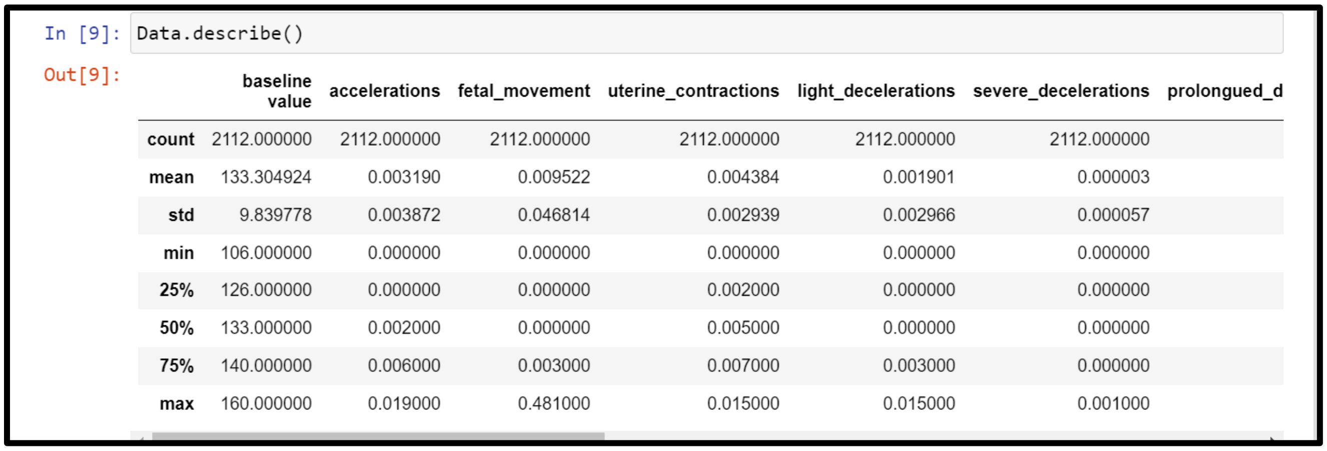
Dataset:



In [31]: `Data.head()`

Out[31]:

	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	
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	



In [9]: `Data.describe()`

Out[9]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_d
count	2112.000000	2112.000000	2112.000000	2112.000000	2112.000000	2112.000000	
mean	133.304924	0.003190	0.009522	0.004384	0.001901	0.000003	
std	9.839778	0.003872	0.046814	0.002939	0.002966	0.000057	
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	
50%	133.000000	0.002000	0.000000	0.005000	0.000000	0.000000	
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	

Scatter Plot between features and label:

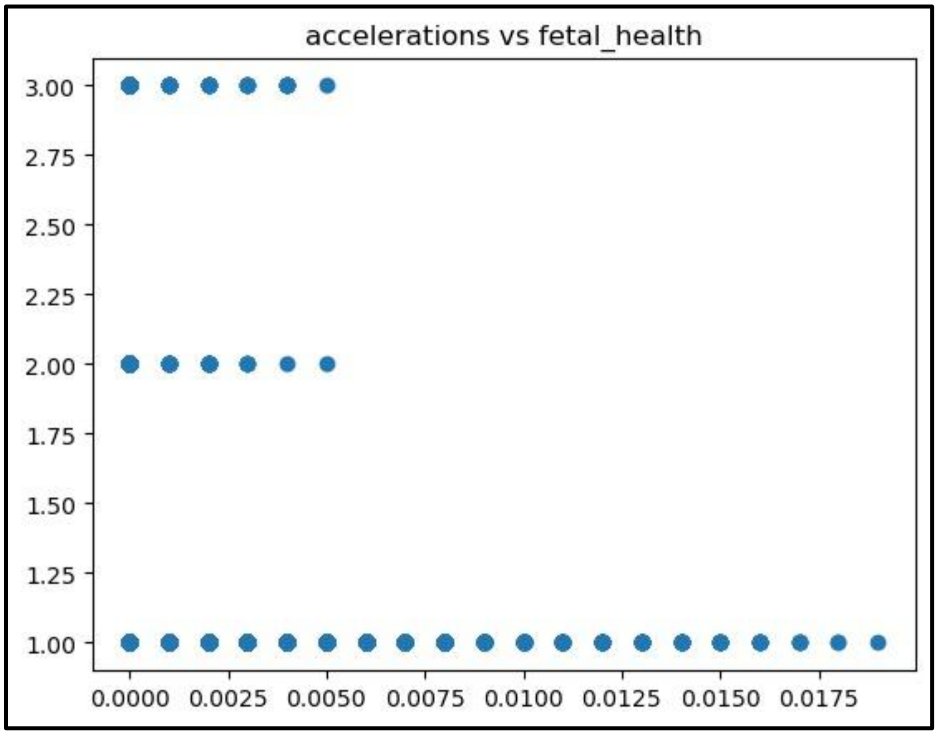


Fig1:scatter plot between accelerations and fetal health

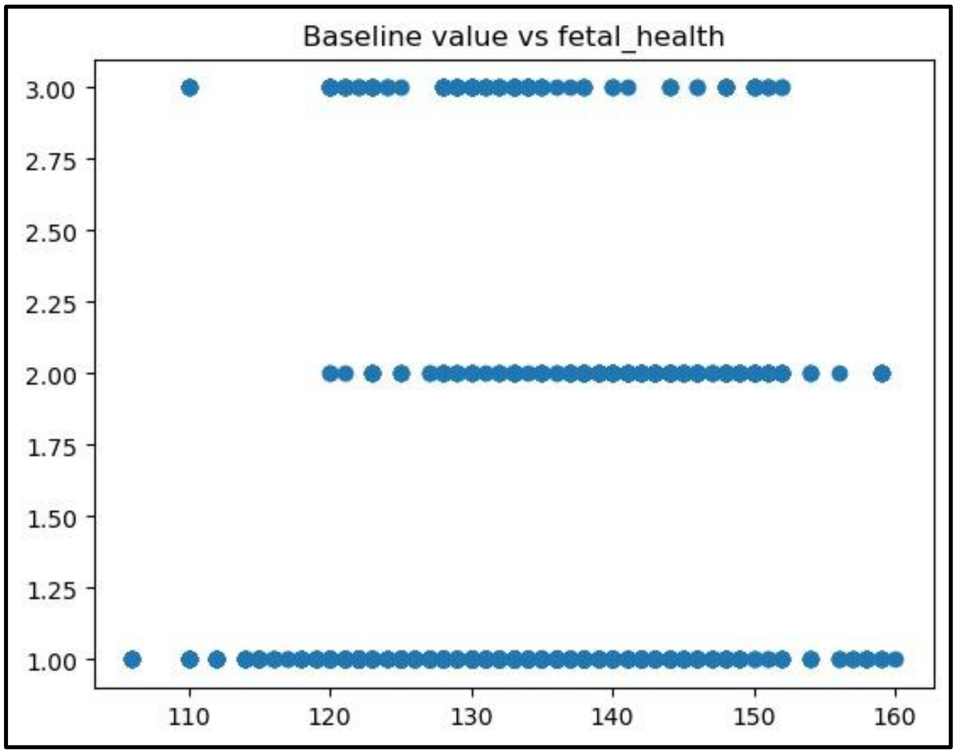


Fig2:scatter plot between baseline value and fetal health

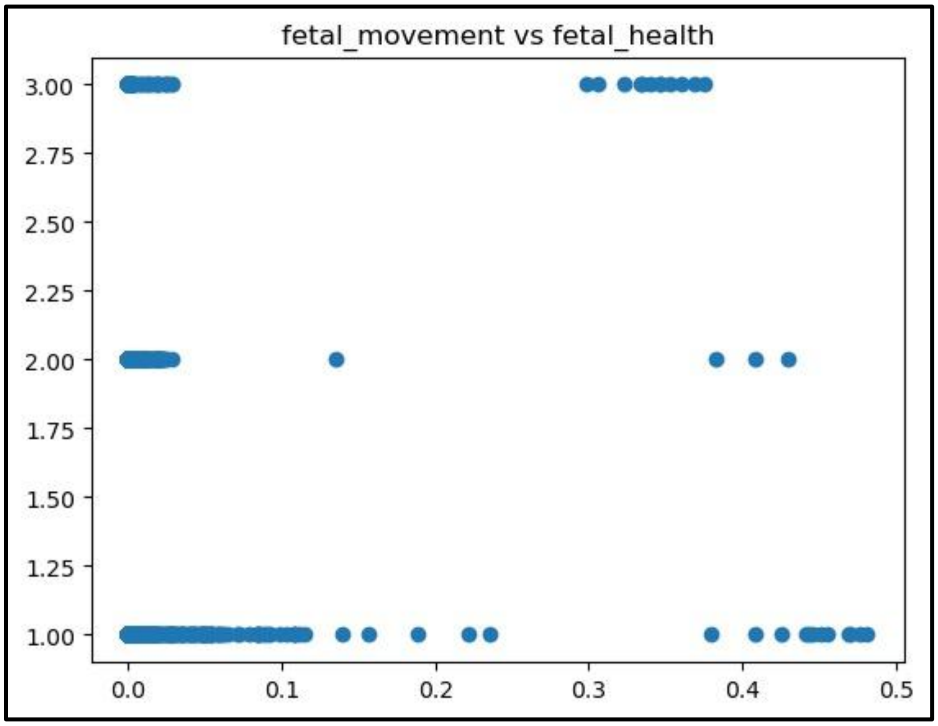


Fig3:scatter plot between fetal movement and fetal health

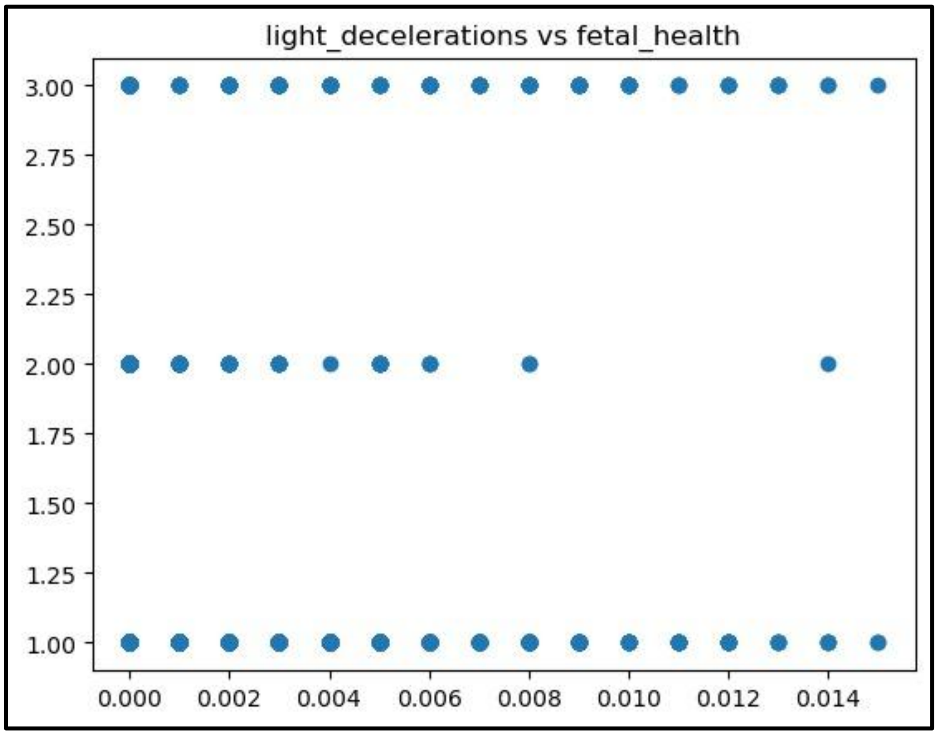


Fig4:scatter plot between light decelerations and fetal health

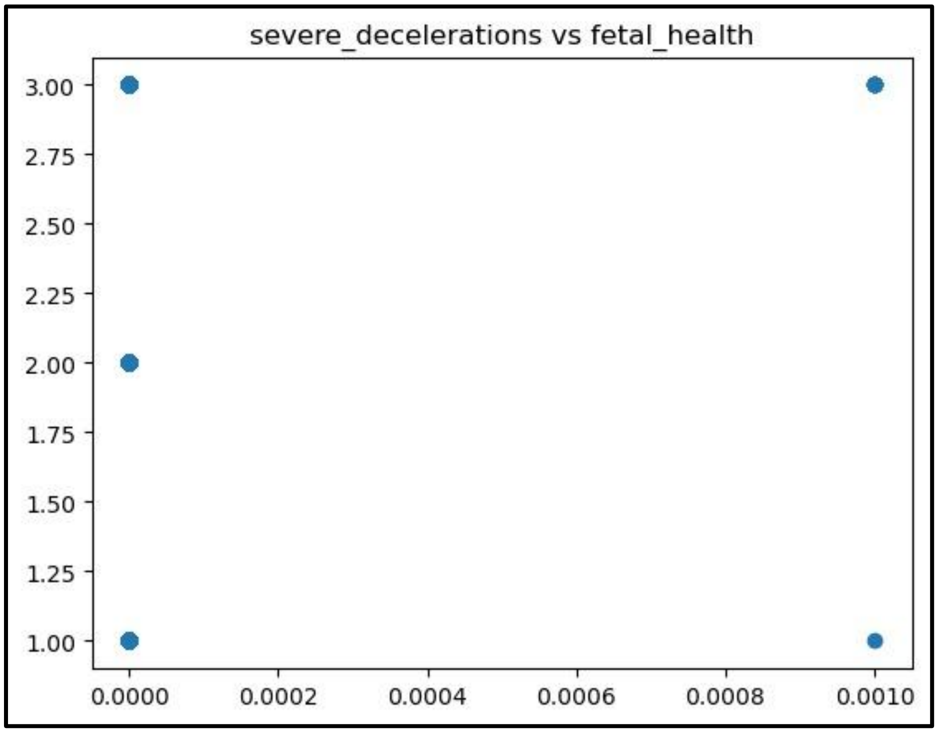


Fig5:scatter plot between severe decelerations and fetal health

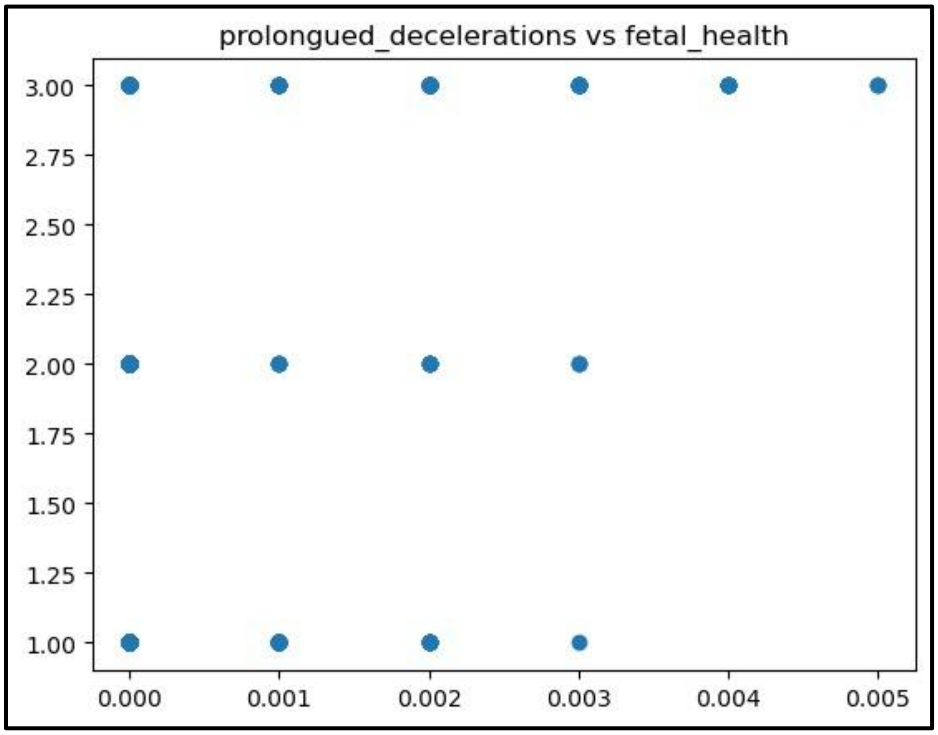


Fig6:scatter plot between prolongued decelerations and fetal health

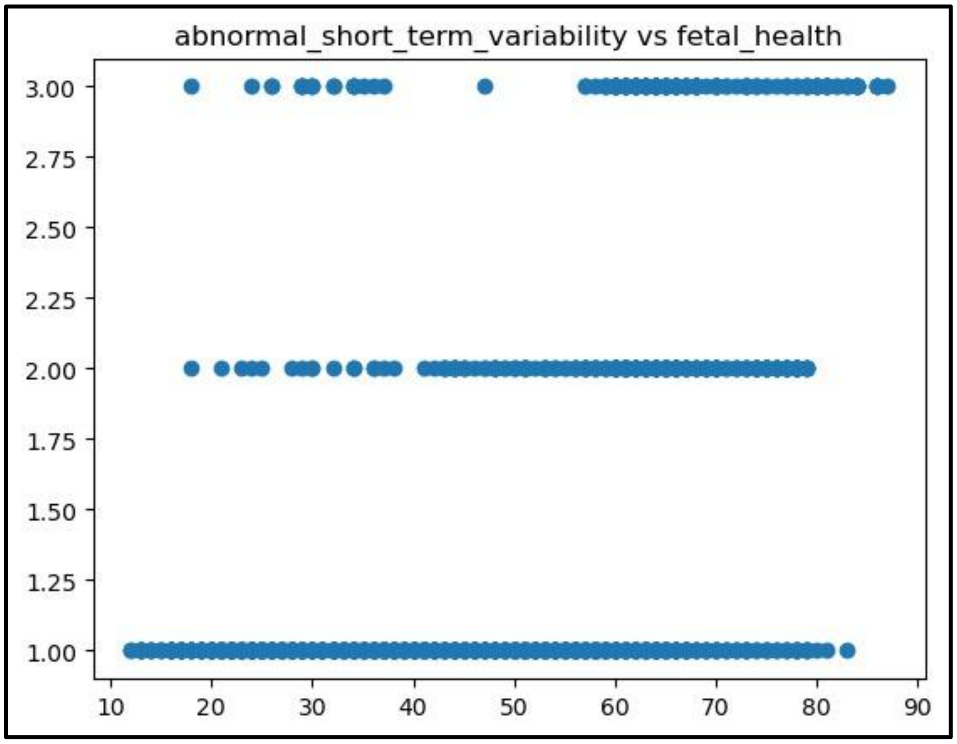


Fig7:scatter plot between abnormal short term variability and fetal health

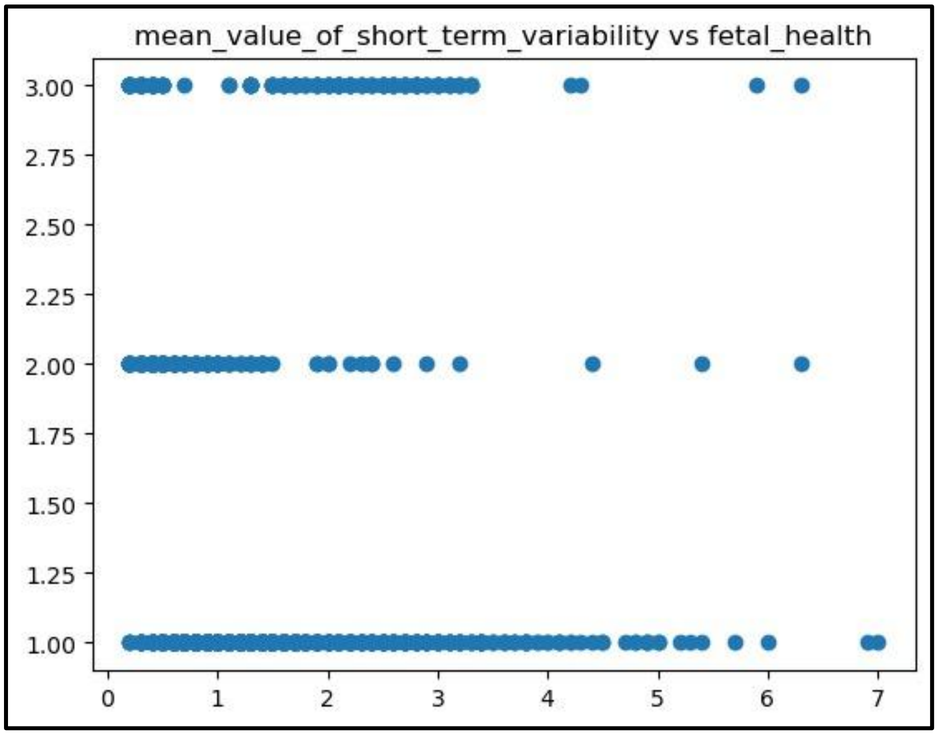


Fig8:scatter plot between mean value of short term variability and fetal health

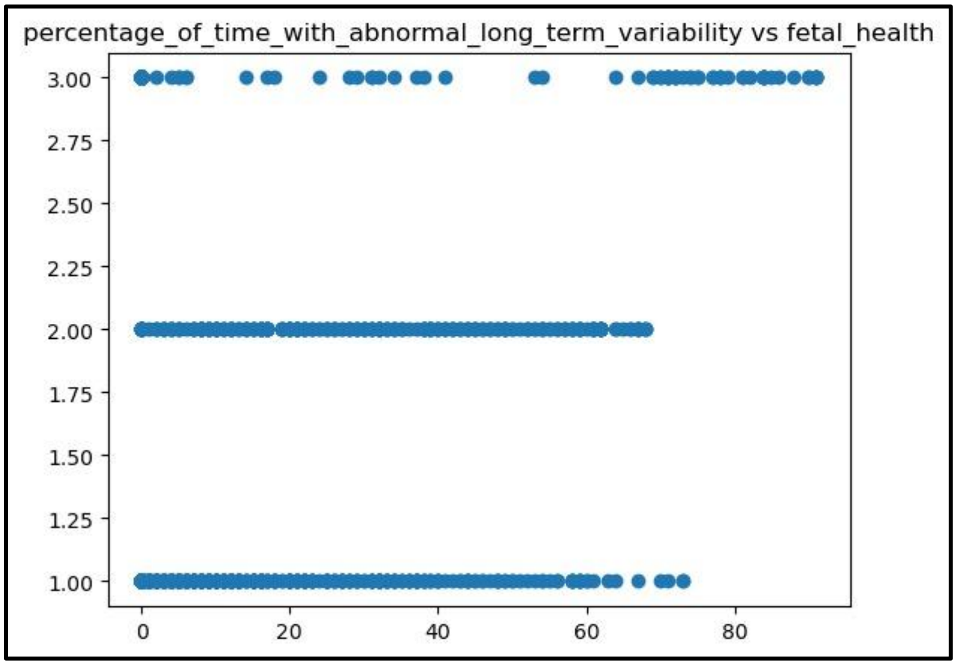


Fig9:scatter plot between percentage of time with abnormal long term variability and fetal health

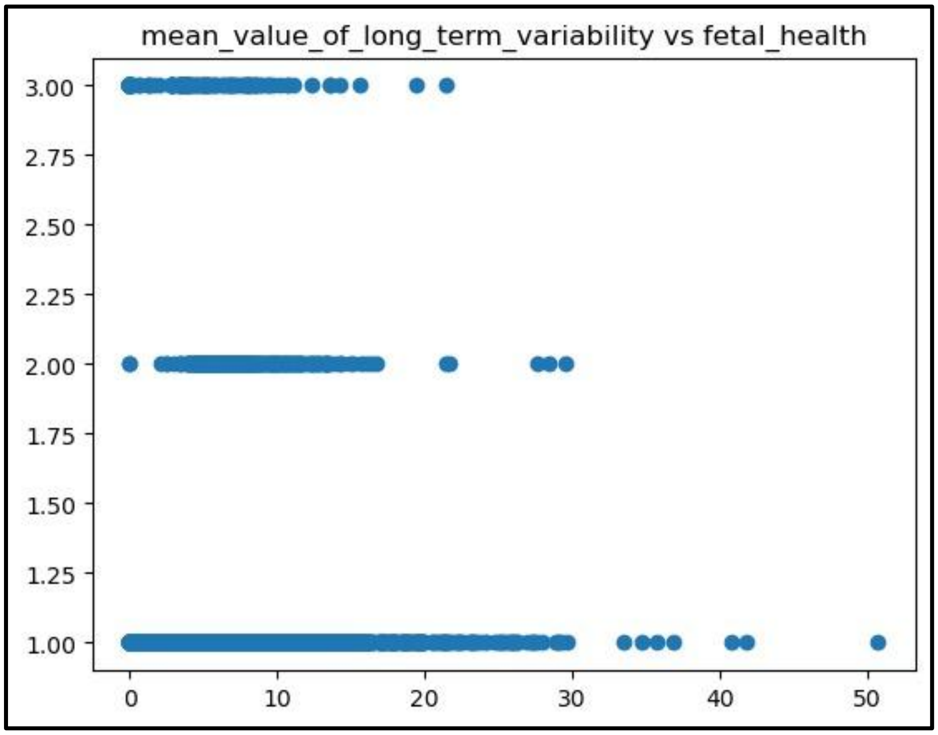
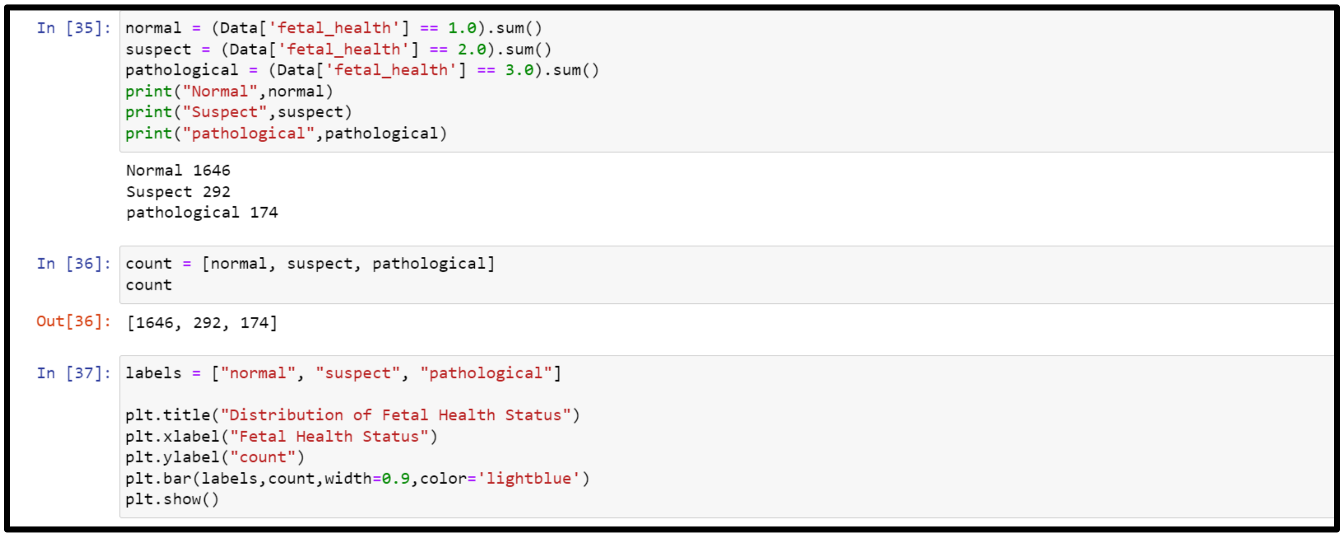


Fig10:scatter plot between mean value long term variability and fetal health



```
In [35]: normal = (Data['fetal_health'] == 1.0).sum()
         suspect = (Data['fetal_health'] == 2.0).sum()
         pathological = (Data['fetal_health'] == 3.0).sum()
         print("Normal",normal)
         print("Suspect",suspect)
         print("pathological",pathological)

         Normal 1646
         Suspect 292
         pathological 174

In [36]: count = [normal, suspect, pathological]
         count

Out[36]: [1646, 292, 174]

In [37]: labels = ["normal", "suspect", "pathological"]

         plt.title("Distribution of Fetal Health Status")
         plt.xlabel("Fetal Health Status")
         plt.ylabel("count")
         plt.bar(labels,count,width=0.9,color='lightblue')
         plt.show()
```

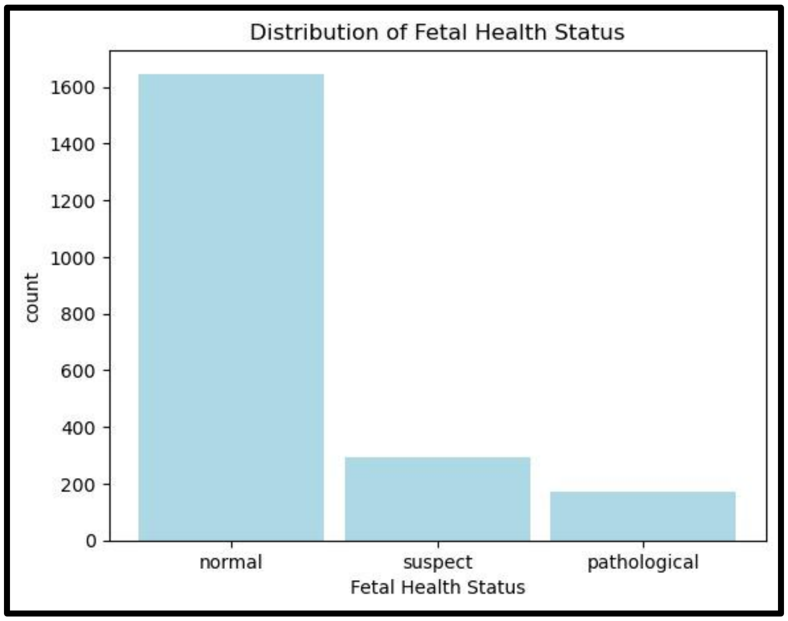


Fig11: Distribution of fetal health status bar graph

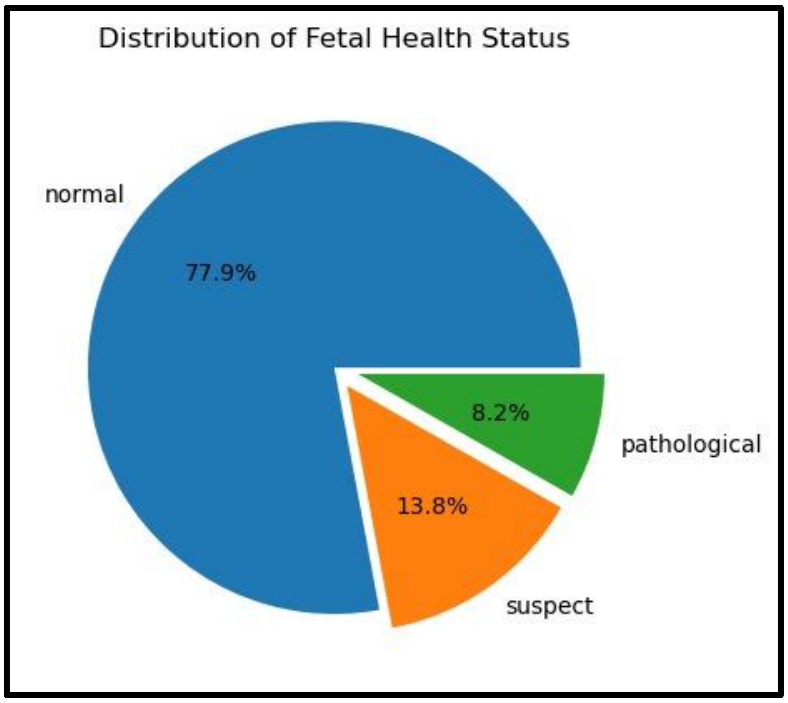
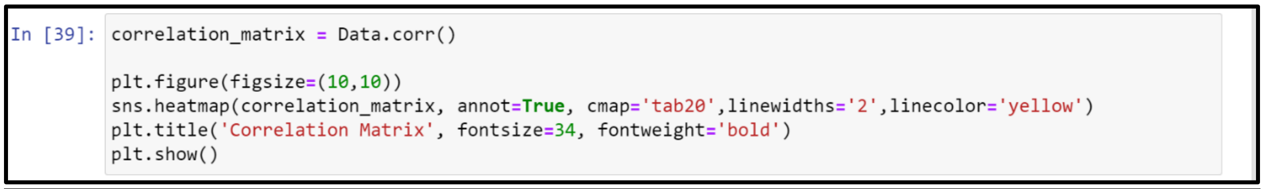
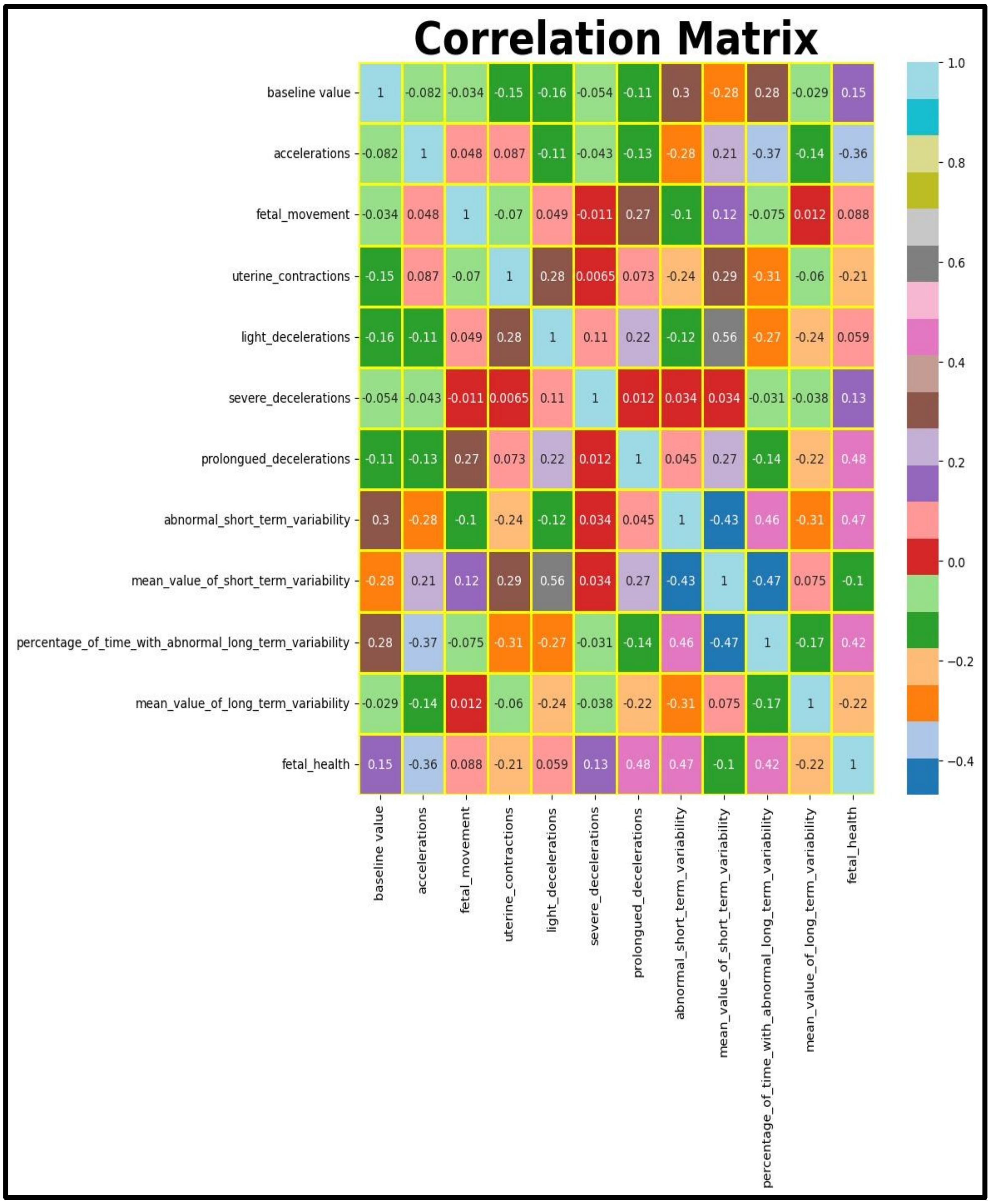


Fig12: Distribution of fetal health status pie chart



```
In [39]: correlation_matrix = Data.corr()

         plt.figure(figsize=(10,10))
         sns.heatmap(correlation_matrix, annot=True, cmap='tab20',linewidths='2',linecolor='yellow')
         plt.title('Correlation Matrix', fontsize=34, fontweight='bold')
         plt.show()
```



	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_variability	mean_value_of_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	mean_value_of_long_term_variability	fetal_health
baseline value	1	-0.082	-0.034	-0.15	-0.16	-0.054	-0.11	0.3	-0.28	0.28	-0.029	0.15
accelerations	-0.082	1	0.048	0.087	-0.11	-0.043	-0.13	-0.28	0.21	-0.37	-0.14	-0.36
fetal_movement	-0.034	0.048	1	-0.07	0.049	-0.011	0.27	-0.1	0.12	-0.075	0.012	0.088
uterine_contractions	-0.15	0.087	-0.07	1	0.28	0.0065	0.073	-0.24	0.29	-0.31	-0.06	-0.21
light_decelerations	-0.16	-0.11	0.049	0.28	1	0.11	0.22	-0.12	0.56	-0.27	-0.24	0.059
severe_decelerations	-0.054	-0.043	-0.011	0.0065	0.11	1	0.012	0.034	0.034	-0.031	-0.038	0.13
prolongued_decelerations	-0.11	-0.13	0.27	0.073	0.22	0.012	1	0.045	0.27	-0.14	-0.22	0.48
abnormal_short_term_variability	0.3	-0.28	-0.1	-0.24	-0.12	0.034	0.045	1	-0.43	0.46	-0.31	0.47
mean_value_of_short_term_variability	-0.28	0.21	0.12	0.29	0.56	0.034	0.27	-0.43	1	-0.47	0.075	-0.1
percentage_of_time_with_abnormal_long_term_variability	0.28	-0.37	-0.075	-0.31	-0.27	-0.031	-0.14	0.46	-0.47	1	-0.17	0.42
mean_value_of_long_term_variability	-0.029	-0.14	0.012	-0.06	-0.24	-0.038	-0.22	-0.31	0.075	-0.17	1	-0.22
fetal_health	0.15	-0.36	0.088	-0.21	0.059	0.13	0.48	0.47	-0.1	0.42	-0.22	1

Fig13: Heat Map representing the dependency between all classification

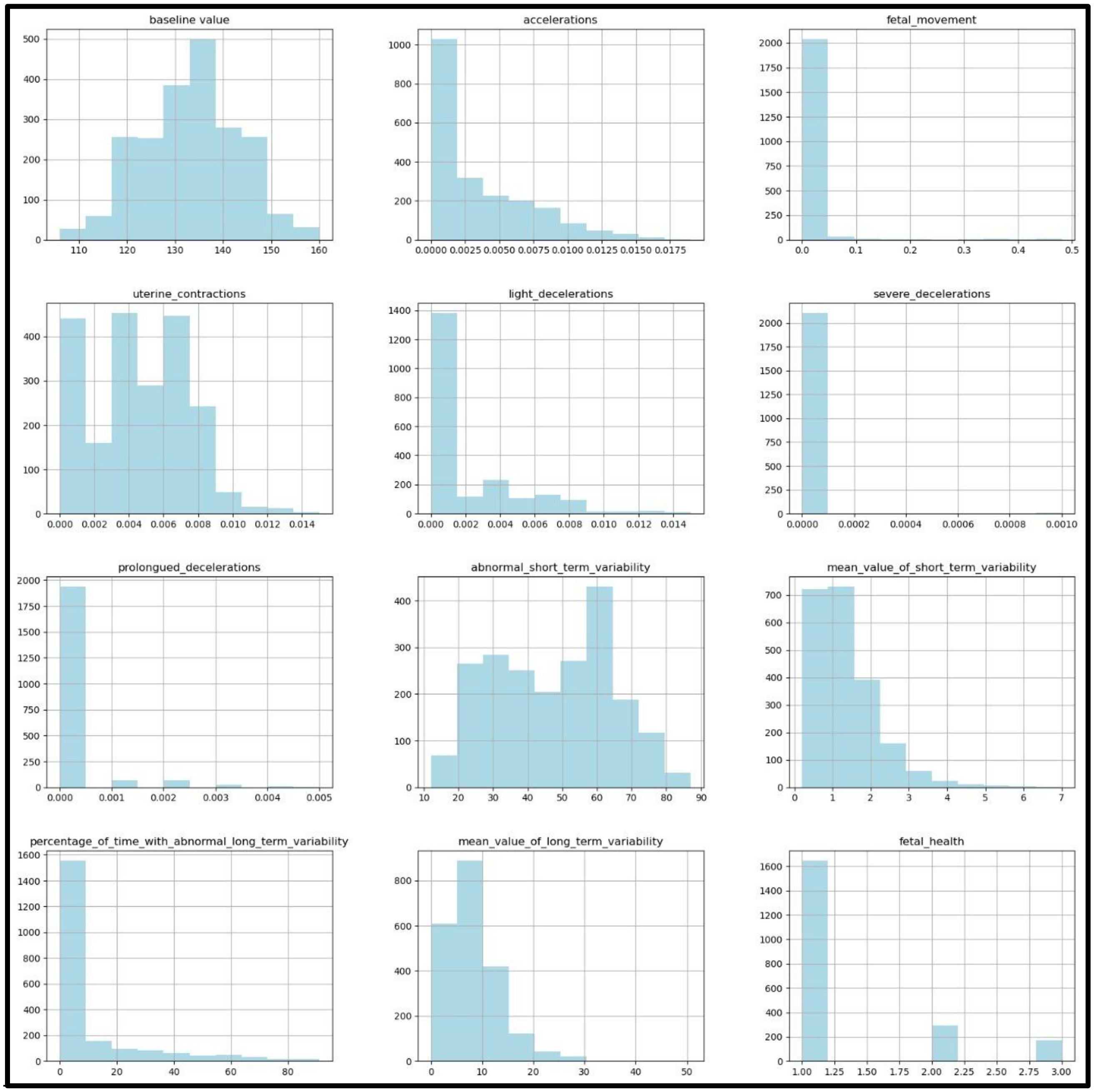


Fig14: Histogram of all the feature columns

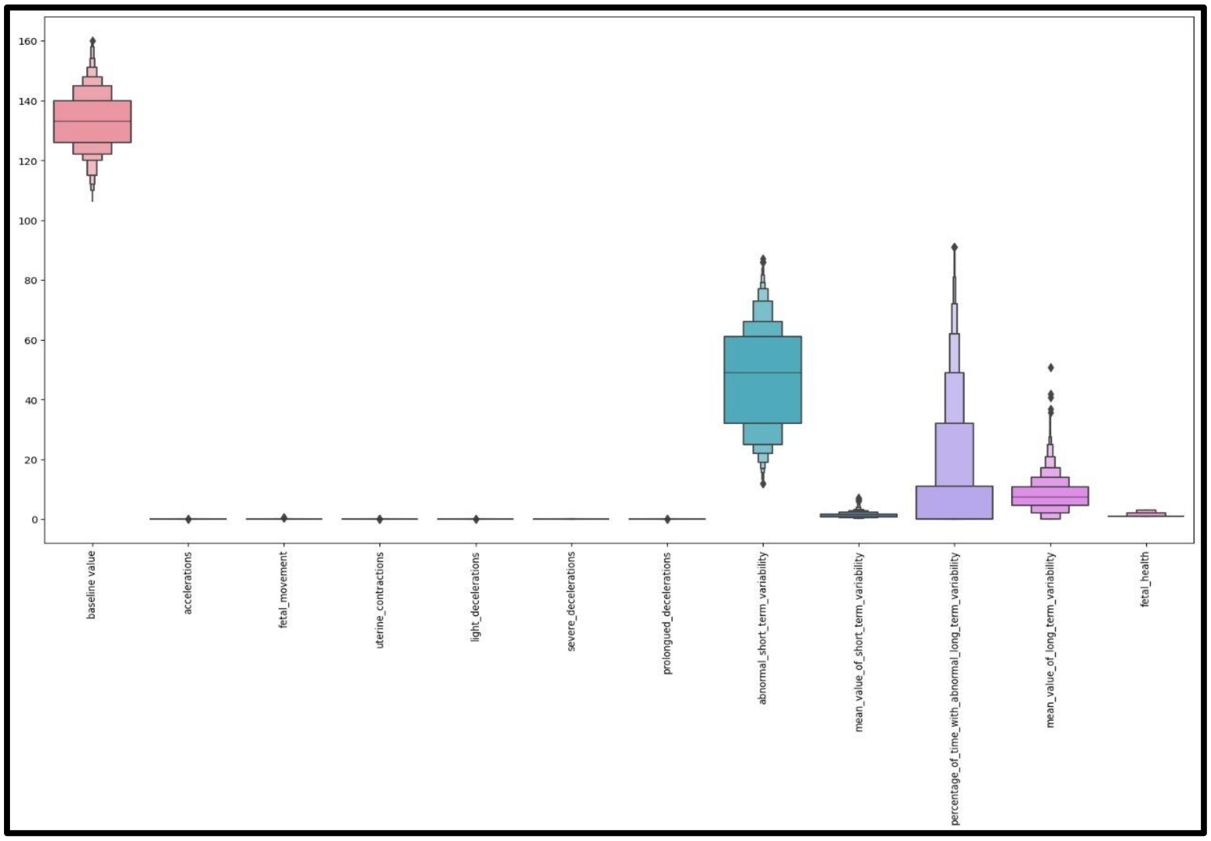


Fig 15: Box plot denoting lower and upper quartile range of feature

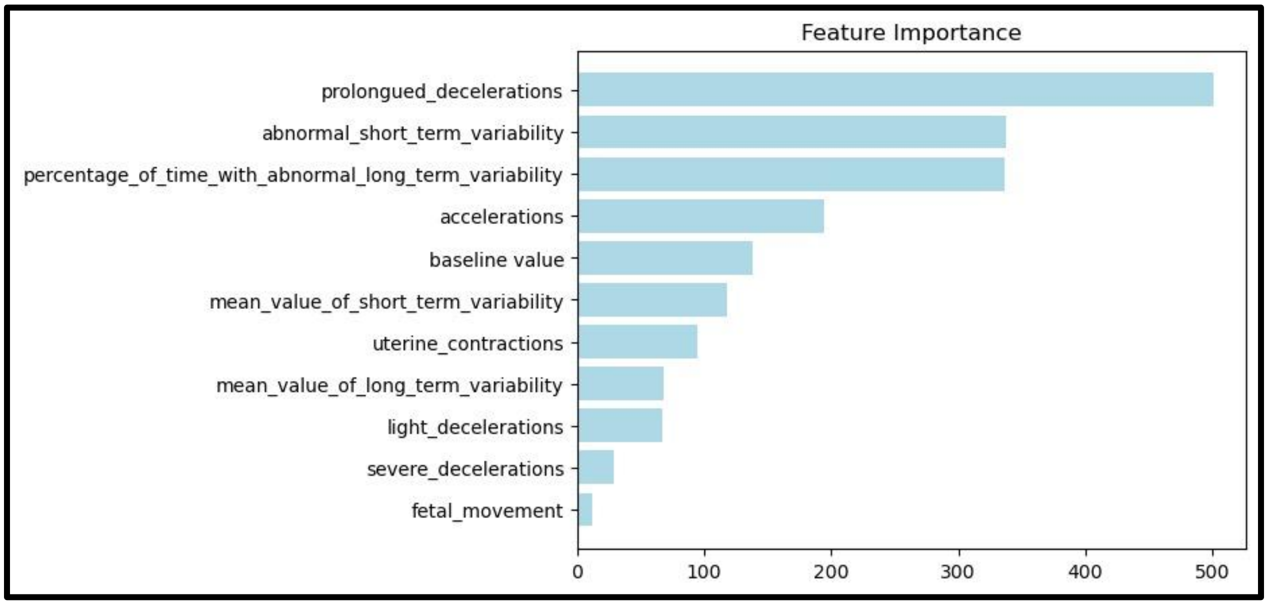
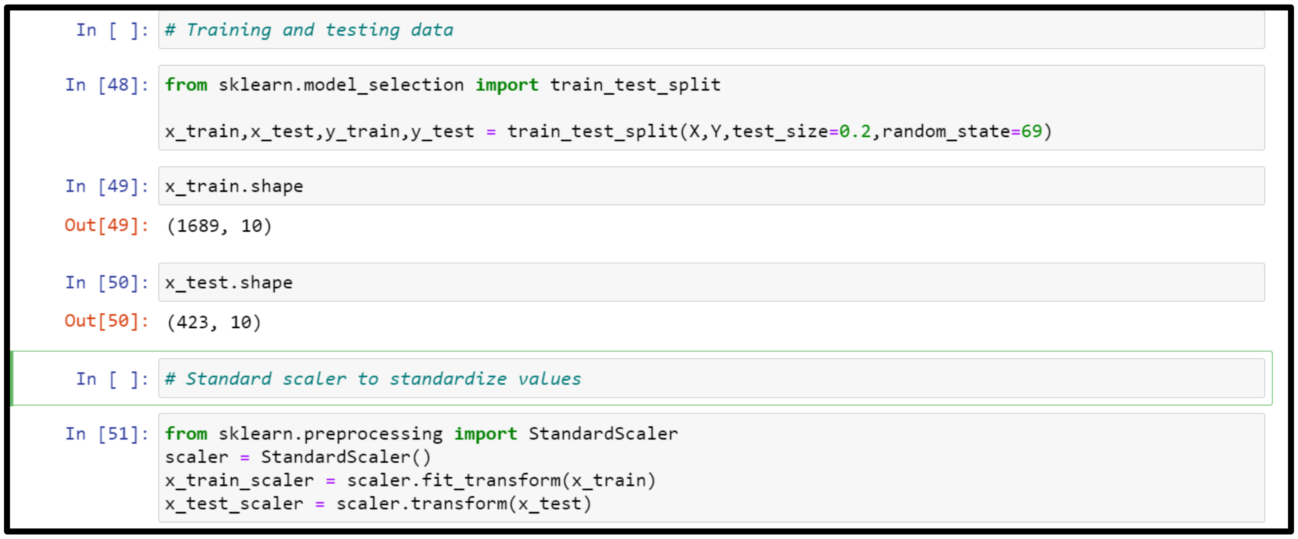
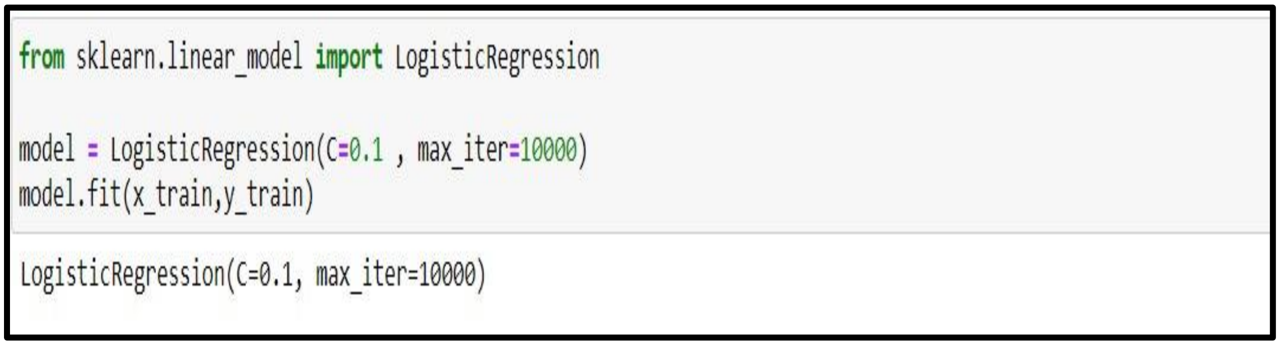


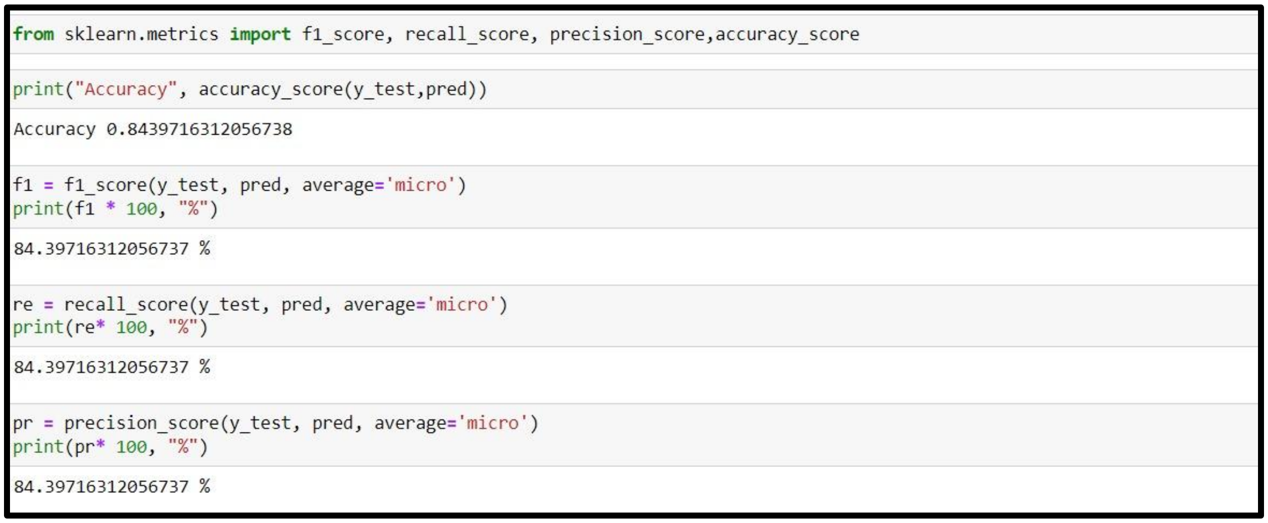
Fig 16: Horizontal bar graph of features vs score



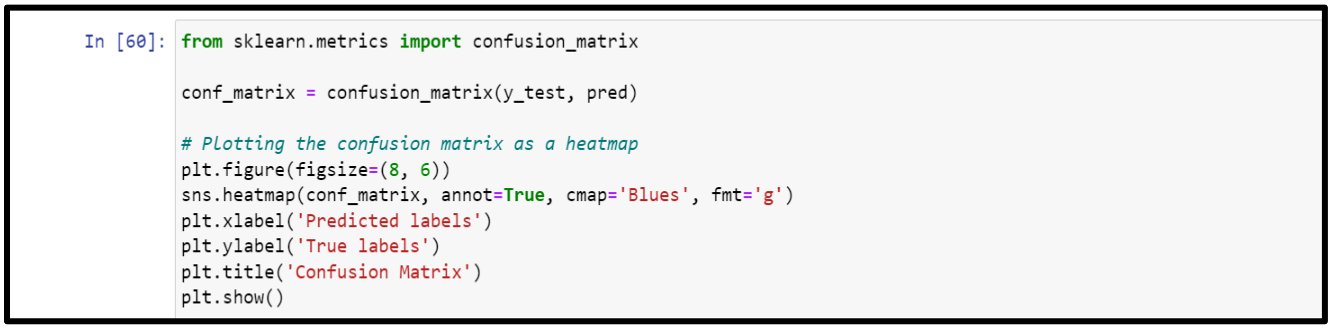
```
In [ ]: # Training and testing data

In [48]: from sklearn.model_selection import train_test_split

         x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=69)

In [49]: x_train.shape

Out[49]: (1689, 10)

In [50]: x_test.shape

Out[50]: (423, 10)

In [ ]: # Standard scaler to standardize values

In [51]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         x_train_scaler = scaler.fit_transform(x_train)
         x_test_scaler = scaler.transform(x_test)
```



```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(C=0.1 , max_iter=10000)
model.fit(x_train,y_train)

LogisticRegression(C=0.1, max_iter=10000)
```



```
from sklearn.metrics import f1_score, recall_score, precision_score,accuracy_score

print("Accuracy", accuracy_score(y_test,pred))

Accuracy 0.8439716312056738

f1 = f1_score(y_test, pred, average='micro')
print(f1 * 100, "%")

84.39716312056737 %

re = recall_score(y_test, pred, average='micro')
print(re* 100, "%")

84.39716312056737 %

pr = precision_score(y_test, pred, average='micro')
print(pr* 100, "%")

84.39716312056737 %
```



```
In [60]: from sklearn.metrics import confusion_matrix

         conf_matrix = confusion_matrix(y_test, pred)

         # Plotting the confusion matrix as a heatmap
         plt.figure(figsize=(8, 6))
         sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')
         plt.xlabel('Predicted labels')
         plt.ylabel('True labels')
         plt.title('Confusion Matrix')
         plt.show()
```

Logistic Regression:

Logistic regression achieved an accuracy of 84.9% in predicting fetal health status, demonstrating its predictive capability, albeit slightly lower than both decision tree (91%) and SVM (84.3%) models.

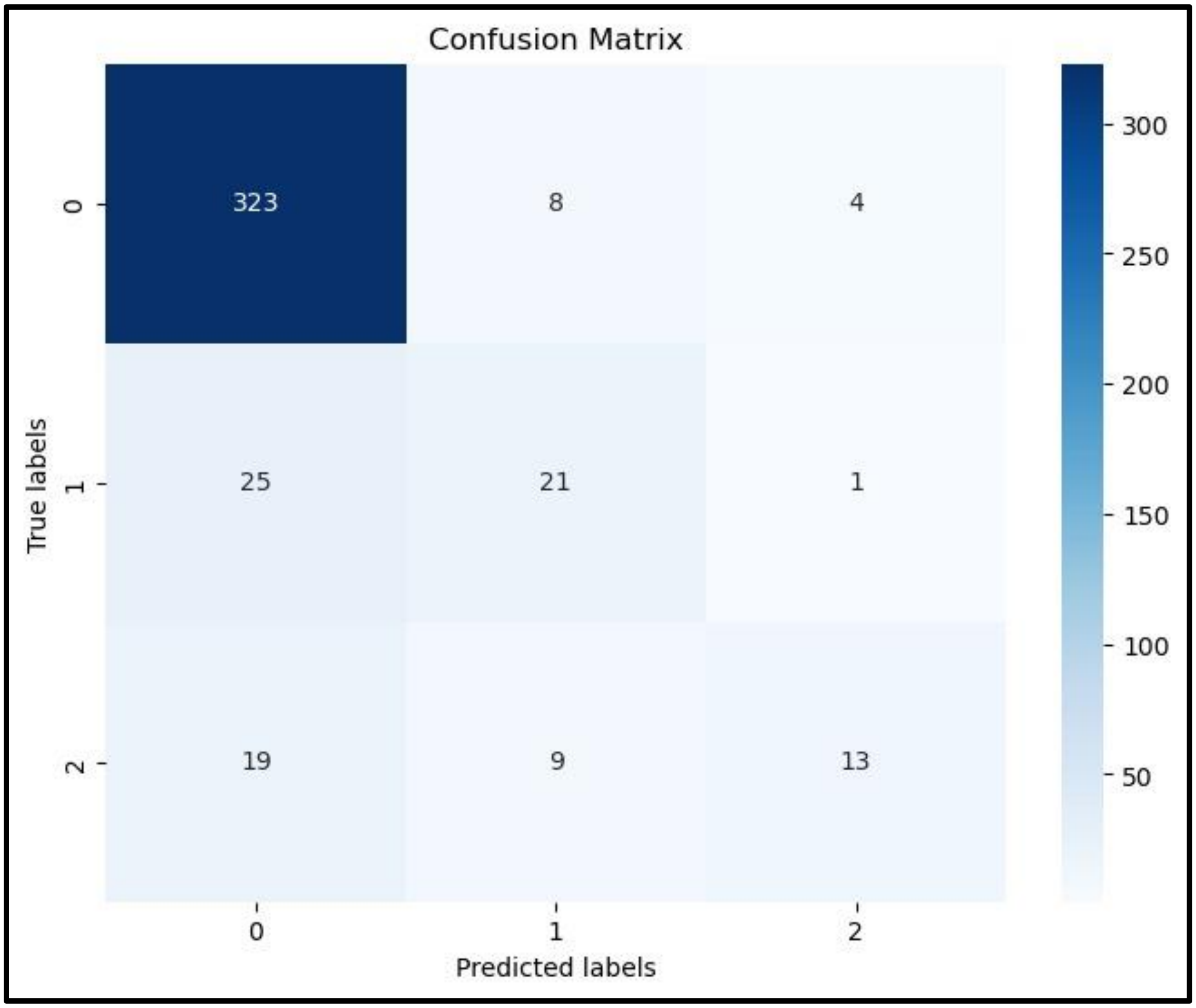
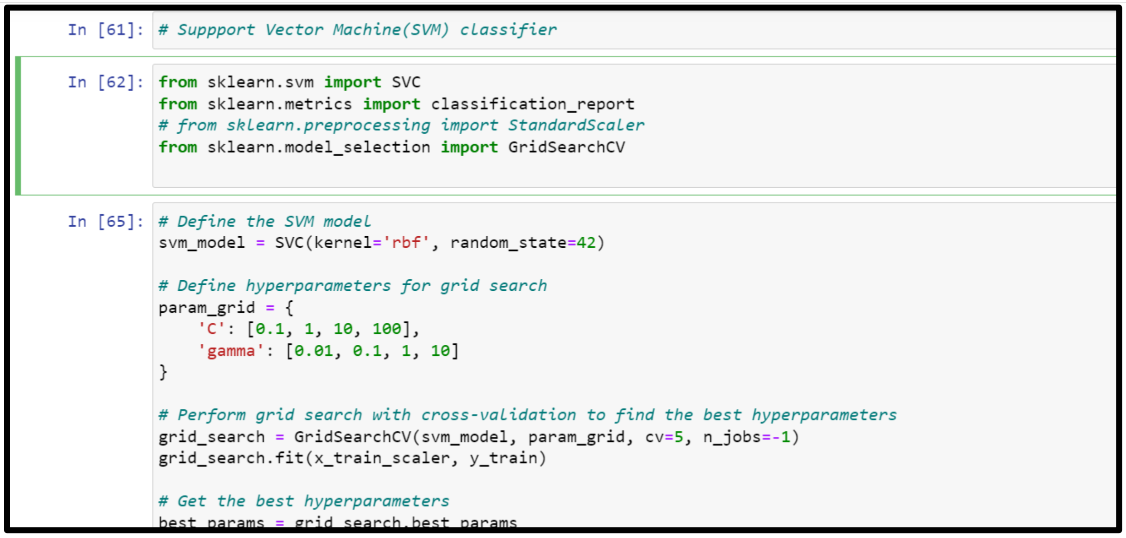


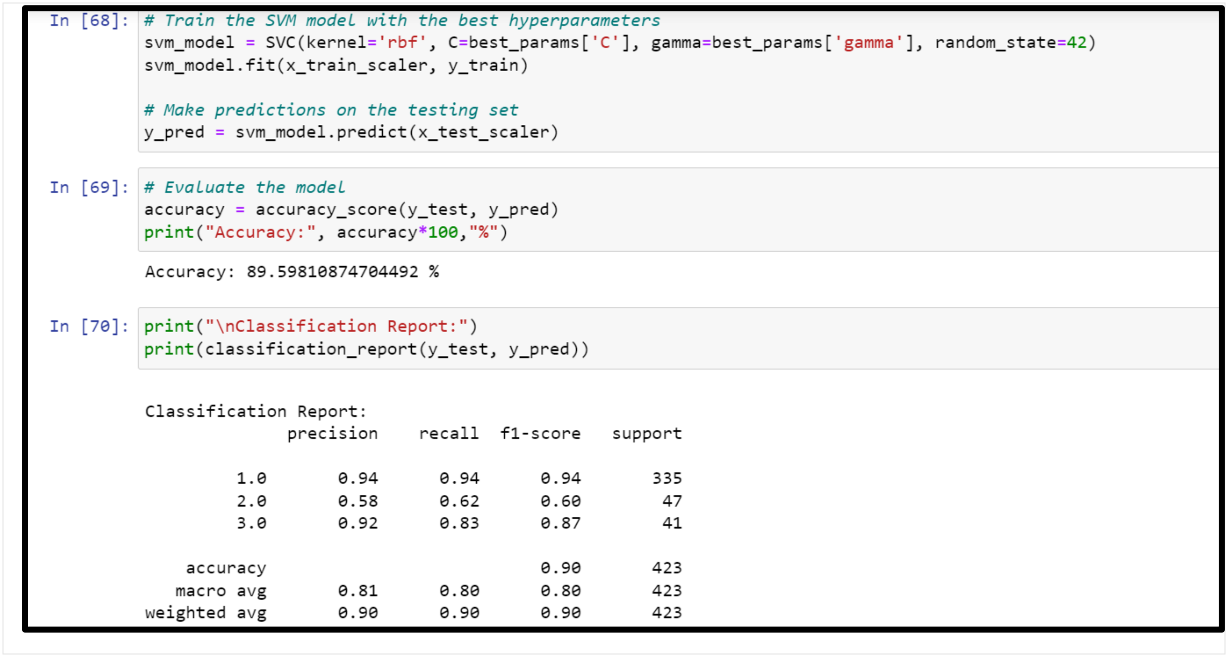
Fig17: Confusion matrix of actual and predicted output values(Logistic)

SVM(support vector machine):

SVM yielded an accuracy of 89% in predicting fetal health status, slightly trailing behind the decision tree model's accuracy of 91%, and outperforming logistic regression (84.3%).



```
In [61]: # Suppport Vector Machine(SVM) classifier

In [62]: from sklearn.svm import SVC
         from sklearn.metrics import classification_report
         # from sklearn.preprocessing import StandardScaler
         from sklearn.model_selection import GridSearchCV

In [65]: # Define the SVM model
         svm_model = SVC(kernel='rbf', random_state=42)

         # Define hyperparameters for grid search
         param_grid = {
             'C': [0.1, 1, 10, 100],
             'gamma': [0.01, 0.1, 1, 10]
         }

         # Perform grid search with cross-validation to find the best hyperparameters
         grid_search = GridSearchCV(svm_model, param_grid, cv=5, n_jobs=-1)
         grid_search.fit(x_train_scaler, y_train)

         # Get the best hyperparameters
         best_params = grid_search.best_params_
```



```
In [68]: # Train the SVM model with the best hyperparameters
         svm_model = SVC(kernel='rbf', C=best_params['C'], gamma=best_params['gamma'], random_state=42)
         svm_model.fit(x_train_scaler, y_train)

         # Make predictions on the testing set
         y_pred = svm_model.predict(x_test_scaler)

In [69]: # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy*100,"%")

         Accuracy: 89.59810874704492 %

In [70]: print("\nClassification Report:")
         print(classification_report(y_test, y_pred))


         Classification Report:
                       precision    recall  f1-score   support

                  1.0       0.94      0.94      0.94       335
                  2.0       0.58      0.62      0.60        47
                  3.0       0.92      0.83      0.87        41

             accuracy                           0.90       423
            macro avg       0.81      0.80      0.80       423
         weighted avg       0.90      0.90      0.90       423
```

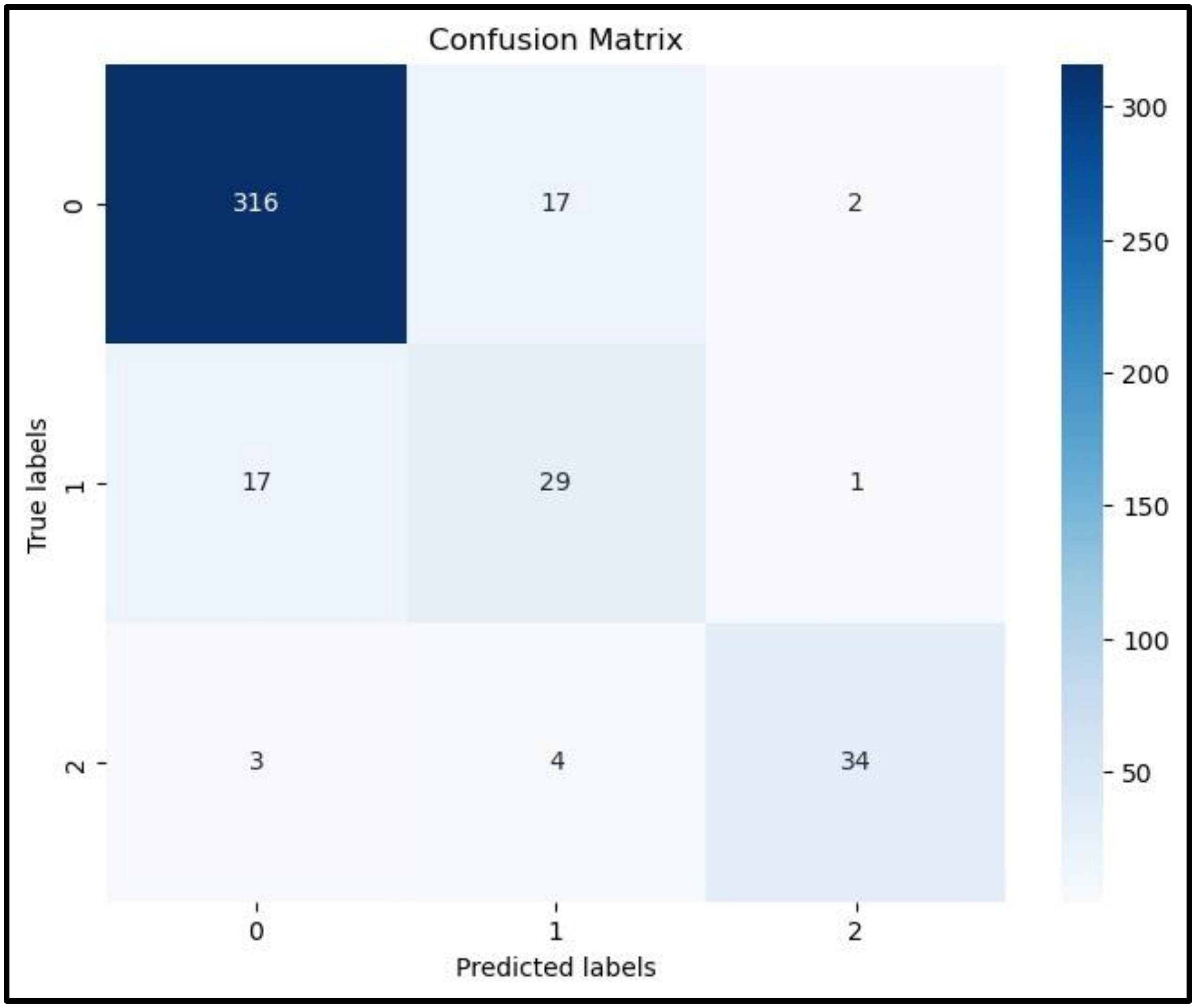
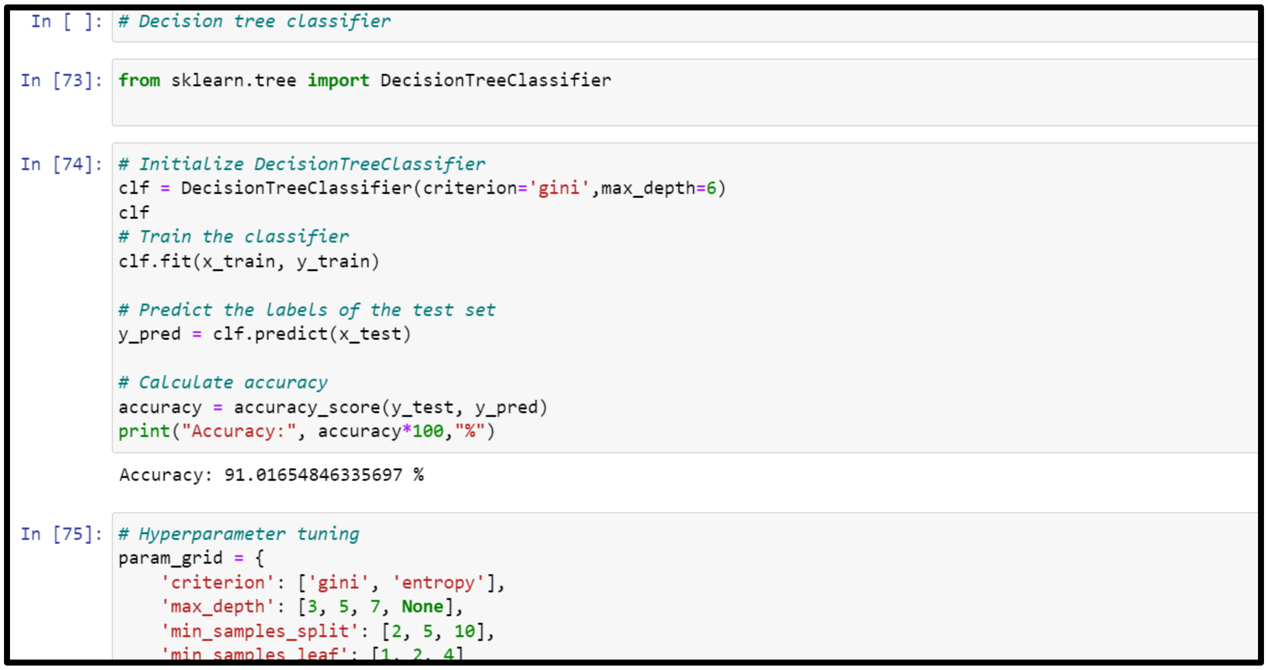


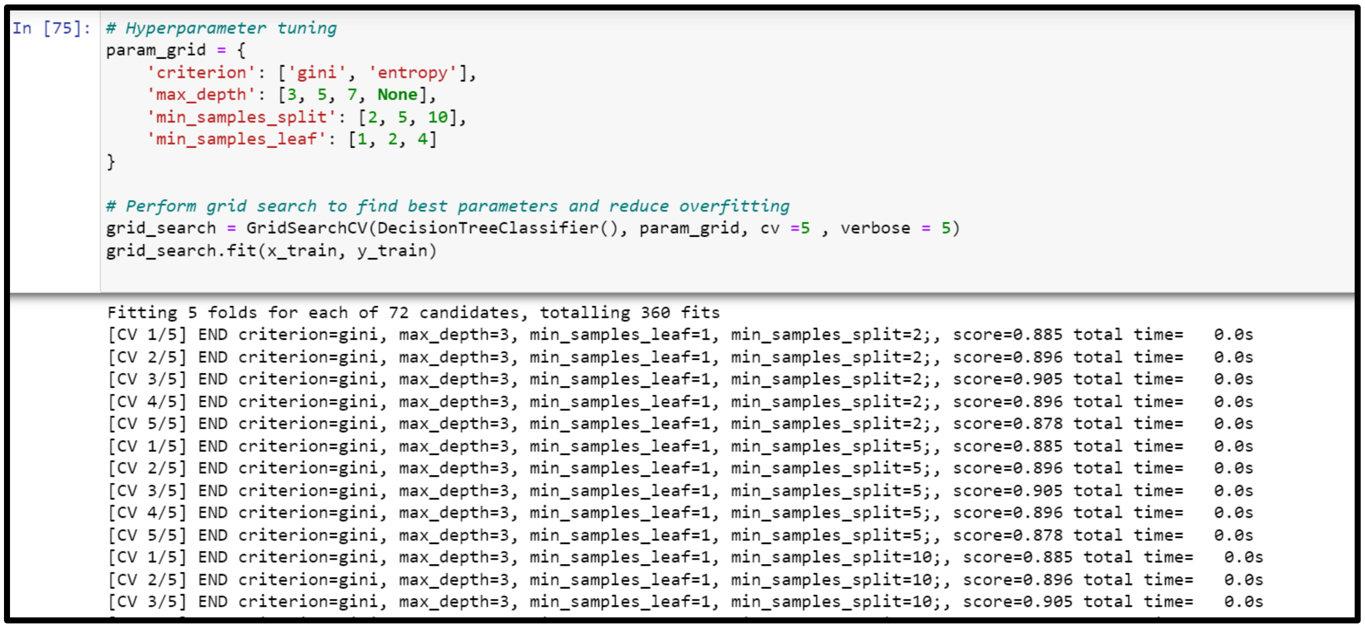
Fig 18: Confusion matrix of actual and predicted output values(SVM)

Decision Tree:

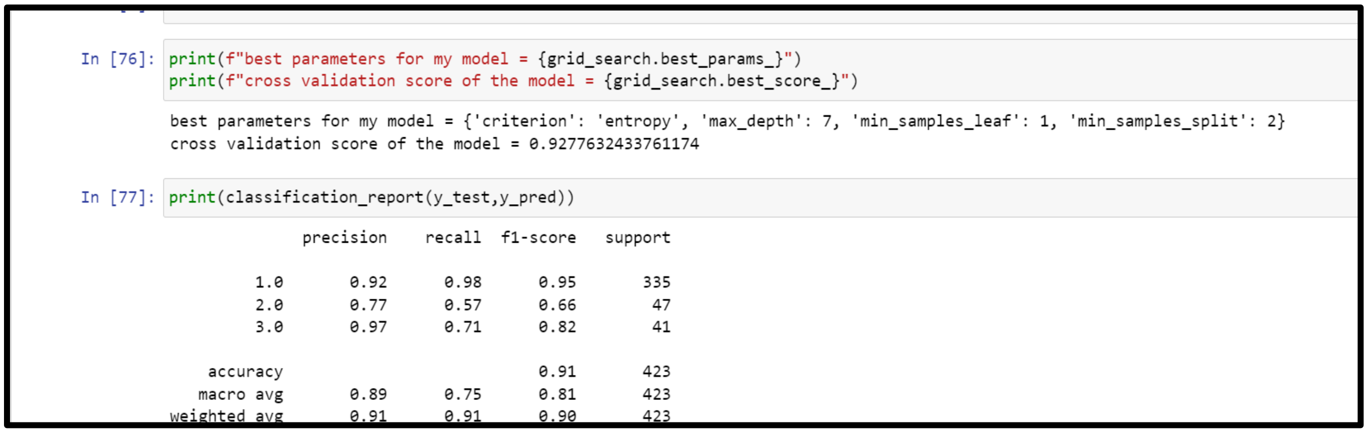
Decision tree analysis achieved the highest accuracy of 91% in predicting fetal health status, showcasing its superior performance compared to SVM (89%) and logistic regression (84.3%).



```
In [ ]: # Decision tree classifier

In [73]: from sklearn.tree import DecisionTreeClassifier

In [74]: # Initialize DecisionTreeClassifier
         clf = DecisionTreeClassifier(criterion='gini',max_depth=6)
         clf
         # Train the classifier
         clf.fit(x_train, y_train)

         # Predict the labels of the test set
         y_pred = clf.predict(x_test)

         # Calculate accuracy
         accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy*100,"%")

         Accuracy: 91.01654846335697 %

In [75]: # Hyperparameter tuning
         param_grid = {
             'criterion': ['gini', 'entropy'],
             'max_depth': [3, 5, 7, None],
             'min_samples_split': [2, 5, 10],
```



```
In [75]: # Hyperparameter tuning
         param_grid = {
             'criterion': ['gini', 'entropy'],
             'max_depth': [3, 5, 7, None],
             'min_samples_split': [2, 5, 10],
             'min_samples_leaf': [1, 2, 4]
         }

         # Perform grid search to find best parameters and reduce overfitting
         grid_search = GridSearchCV(DecisionTreeClassifier(), param_grid, cv =5 , verbose = 5)
         grid_search.fit(x_train, y_train)

         Fitting 5 folds for each of 72 candidates, totalling 360 fits
         [CV 1/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=2;, score=0.885 total time=   0.0s
         [CV 2/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=2;, score=0.896 total time=   0.0s
         [CV 3/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=2;, score=0.905 total time=   0.0s
         [CV 4/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=2;, score=0.896 total time=   0.0s
         [CV 5/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=2;, score=0.878 total time=   0.0s
         [CV 1/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=5;, score=0.885 total time=   0.0s
         [CV 2/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=5;, score=0.896 total time=   0.0s
         [CV 3/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=5;, score=0.905 total time=   0.0s
         [CV 4/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=5;, score=0.896 total time=   0.0s
         [CV 5/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=5;, score=0.878 total time=   0.0s
         [CV 1/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=10;, score=0.885 total time=   0.0s
         [CV 2/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=10;, score=0.896 total time=   0.0s
         [CV 3/5] END criterion=gini, max_depth=3, min_samples_leaf=1, min_samples_split=10;, score=0.905 total time=   0.0s
```



```
In [76]: print(f"best parameters for my model = {grid_search.best_params_}")
         print(f"cross validation score of the model = {grid_search.best_score_}")

         best parameters for my model = {'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 1, 'min_samples_split': 2}
         cross validation score of the model = 0.9277632433761174

In [77]: print(classification_report(y_test,y_pred))

                      precision    recall  f1-score   support

                  1.0       0.92      0.98      0.95       335
                  2.0       0.77      0.57      0.66        47
                  3.0       0.97      0.71      0.82        41

             accuracy                           0.91       423
            macro avg       0.89      0.75      0.81       423
         weighted avg       0.91      0.91      0.90       423
```

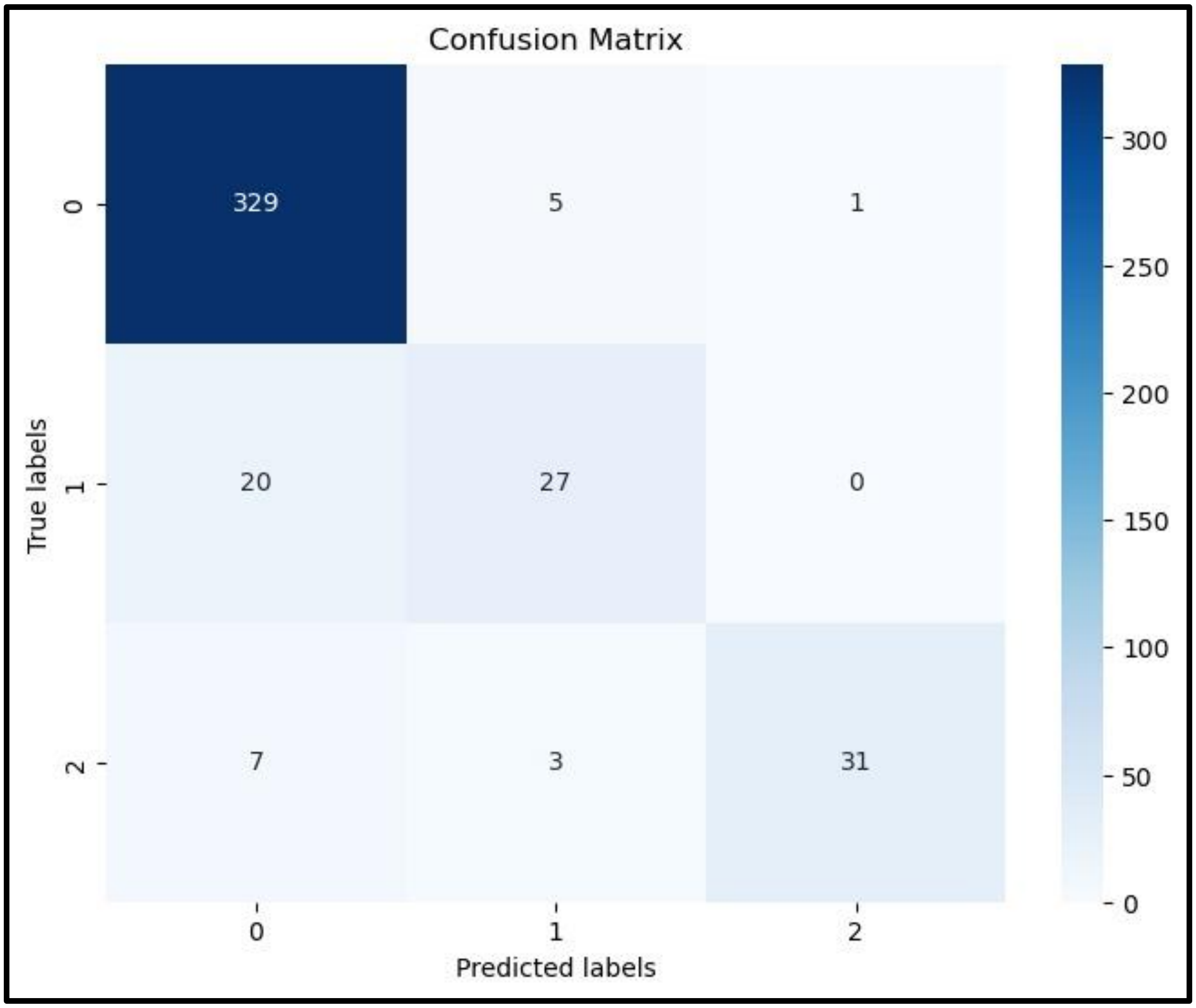


Fig 19: Confusion matrix of actual and predicted output values(Decision tree)

References

- https://www.kaggle.com
- https://www.geeksforgeeks.org/machine-learning-with-python