BUDT704: DATA PROCESSING AND ANALYSIS IN PYTHON

**"Classification of Fetal Health"**


By:

ASHWIN BALAJI

DHWANI MUNI

LINDA TOM

MANASVI SURASANI

NAYANIKA BOBBILI

PRAKRUTHI KAGANTI SHIVAKUMAR

## INTRODUCTION

Maternal and fetal health remains a critical concern globally. In 2022, the U.S. reported 22 maternal deaths per 100,000 live births and a fetal mortality rate of 5.8 per 1,000 live births. Despite improvements, these figures underline the need for better diagnostic tools to reduce adverse outcomes. This project leverages machine learning techniques to analyze Cardiotocogram (CTG) data, which provides insights into fetal movements, uterine contractions, and other key health indicators. By classifying fetal health into three categories—Normal, Suspect, and Pathological—this study aims to enhance clinical decision-making, especially in resource-limited settings, to ensure timely interventions and improved health outcomes for mothers and children.

## BACKGROUND

Cardiotocograms (CTGs) are vital tools in obstetrics, offering real-time monitoring of fetal and maternal well-being during pregnancy. However, the interpretation of CTG data often requires expertise, and subjective evaluations may lead to inconsistent results. With advancements in machine learning, automated systems now have the potential to supplement healthcare providers by providing accurate, consistent, and scalable analyses. This project utilizes a dataset from Kaggle on fetal health classification, which includes features derived from CTG records. By employing algorithms such as K-Nearest Neighbors (KNN), Logistic Regression, Random Forest, and Support Vector Machines (SVM), this study identifies optimal models for classifying fetal health and guiding clinical interventions.

## RESEARCH METHOD SUMMARY

### Data Preprocessing

- The dataset was split into training (70%) and testing (30%) sets.
- Features were standardized to ensure uniform contributions to the models.

### Model Implementation

- KNN: Classified data points based on the majority vote of their nearest 5 neighbors.
- Logistic Regression: Adapted multinomial regression to classify three categories.
- SVM: Utilized kernel functions for handling non-linear relationships.
- Random Forest: Combined predictions from multiple decision trees for robust classification.

### Model Evaluation

- 5-fold cross-validation was employed for reliability.
- Metrics such as accuracy, precision, recall, and F1-score were analyzed.
- Hyperparameter tuning was conducted for SVM and Random Forest to enhance performance.

### Key Findings

- Random Forest emerged as the best-performing model, especially in identifying complex pathological cases, with high recall and F1 scores.
- Logistic Regression and KNN demonstrated reasonable accuracy but struggled with non-linear relationships and imbalanced classes.

**OBJECTIVE**

The primary objective of this project is to reduce maternal and fetal mortality rates by developing accurate and efficient predictive models for classifying fetal health. By integrating machine learning into clinical workflows, this study aims to provide healthcare providers with reliable tools to identify potential complications early, enabling timely interventions and improved health outcomes, particularly in resource-constrained environments.

**EXPLANATION FOR DATASET**

The following are explanations for key metrics in the data set:

- baseline value: Average heart rate of the fetus when it is resting/not actively moving
- accelerations: Count of times fetal heart rate increases per second
- fetal_movement: Count of times the fetus moves per second.
- uterine_contractions: Count of uterine contractions per second
- light_decelerations: Count of small temporary decreases in fetal heart rate/sec
- severe_decelerations: Count of severe decelerations per second
- prolongued_decelerations: Count of prolonged decelerations per second
- abnormal_short_term_variability: Percentage of time fetal heart rate changes drastically over a short time frame.
- mean_value_of_short_term_variability: Average value of short-term changes in fetal heart rate
- percentage_of_time_with_abnormal_long_term_variability: Percentage of time fetal heart rate changes drastically over a longer time frame
- mean_value_of_long_term_variability: Average value of long-term changes in fetal heart rate
- fetal_health: Class label for fetal health where 0:Normal, 1:Suspect, 2: Pathological

This dataset contains 2,126 records derived from features extracted from Cardiotocogram (CTG) exams, which are widely used in fetal health monitoring. CTGs operate by sending ultrasound pulses and analyzing their responses to measure various parameters, such as fetal heart rate (FHR), fetal movements, and uterine contractions. These features provide critical insights into fetal health, enabling timely interventions to reduce child and maternal mortality. The dataset has been categorized into three distinct classes—Normal, Suspect, and Pathological—based on classifications provided by three expert obstetricians. This classification aids healthcare professionals in identifying and addressing potential risks during pregnancy, contributing to the global efforts to prevent preventable deaths of newborns and mothers in line with the United Nations' Sustainable Development Goals.

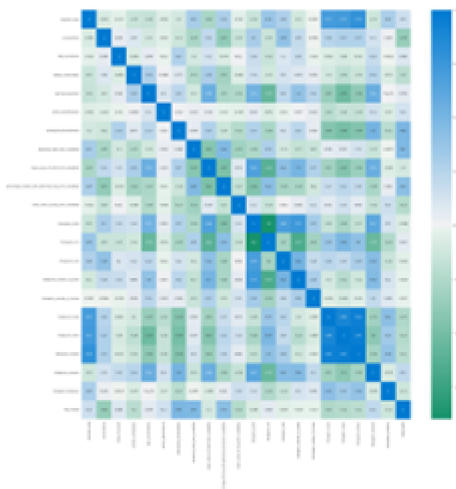
**DATA CLEANING**

The dataset was thoroughly evaluated for missing entries and null values. It was discovered to be complete with no missing data values. Therefore, no additional cleaning was done, and the dataset was adopted in its original state for the analysis

## EXPLORATORY DATA ANALYSIS

(A) CORRELATION

The correlation heatmap below visualizes the relationships between variables in the dataset. Blue boxes indicate strong positive correlations, while green boxes highlight strong negative correlations. For instance, a strong positive correlation is observed between **"histogram_mean"** and **"histogram_median,"** suggesting similar behavior. Conversely, minimal or no correlation between **"baseline value"** and **"field_movement"** implies these variables are largely independent.



(B) TRAIN, TEST, SPLIT DATA

1. Feature Selection

All features from the dataset except the target column (`fetal_health`) were selected for the model. These features represent the input variables (X) used to predict the target variable (y), which contains the labels for fetal health categories.

2. Feature Scaling

The features were scaled using StandardScaler, which standardizes the data by removing the mean and scaling it to unit variance. This step ensures that all features contribute equally to the model's performance, as KNN is sensitive to the scale of the input data.

3. Data Splitting

The dataset was split into training and test sets in a 70:30 ratio using train_test_split. The training set was used to train the model, while the test set was reserved for evaluating its performance on unseen data. A random seed (random state = 42) was set to ensure the reproducibility of the split.

This preprocessing ensures that the data is properly scaled and split, enabling the KNN model to effectively learn patterns from the training set and generalize its predictions on the test set. Scaling was a critical step to handle the algorithm's sensitivity to feature magnitudes.

## DATA ANALYSIS AND MODELING

(A) KNN

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric machine learning technique used for both classification and regression tasks. It predicts the output for a given input by finding the k nearest data points (neighbors) in the training set, based on a defined distance metric (e.g., Euclidean distance), and making predictions based on the majority class (for classification) or the average value (for regression).

**Key Steps and Results**

1. Cross-Validation
    - 5-fold cross-validation was performed to evaluate the model's performance. The dataset was divided into 5 subsets, and the model was trained on 4 subsets while being tested on the remaining one. This process was repeated for all subsets.
    - The cross-validation scores ranged from 0.86 to 0.92, with an average accuracy of approximately 0.89, indicating consistent model performance.

2. Training and Testing
    - The model was trained using the entire training dataset and then evaluated on the test dataset.
    - The final accuracy on the test dataset was 91%, suggesting that the model generalizes well to unseen data.

3. Model Parameters
    - The KNN model was configured with 5 neighbors (k=5) and used uniform weights, meaning all neighbors contributed equally to the prediction.

The KNN model demonstrated robust performance, as reflected in the high accuracy scores during cross-validation and on the test dataset. These results suggest that KNN is a suitable algorithm for this classification task, given the dataset and the parameters chosen.

(B) LOGISTIC REGRESSION

Logistic regression is typically used for binary classification (with two outcomes). By adopting the logistic function, it models the relationship between one or more independent variables and the probability of a specific outcome. It maps predicted values to probabilities, ensuring outputs range between 0 and 1. In particular, multinomial logistic regression is adapted as it can predict categorical outcomes with three or more classes. This is suitable for our exploration as we are looking to classify fetal health into three distinct categories: Normal, Suspect, and Pathological. By estimating the probability of each class via simultaneously solving many logistic regression equations, the model estimates the probability of each class, making it a suitable option for multi-class classification objectives.

**Key Steps and Results**

1. Training and Testing Logistic Regression

The logistic regression model was instantiated with default parameters. The model was then fit on the training data. The model accuracy was evaluated based on the training and testing data. The baseline logistic regression accuracy for training and testing data was 0.90 and 0.88 respectively.

2. Cross-Validation

- 5-fold cross-validation was performed (for similar reasons mentioned in KNN)
- The cross-validation scores ranged from 0.88 to 0.91, with an average accuracy of approximately 0.90, indicating consistent model performance.

3. Hyperparameter Tuning

By adopting GridSearchCV, we tested different values across selected hyperparameters to identify the optimal configuration for logistic regression. The parameters we selected for tuning and the reasons behind why we optimized them are as follows:

- C:  It controls the strength of regularization. Lower values enhance regularization, decreasing overfitting. The values, [0.01, 0.1, 1, 10, 100] were selected to balance performance and complexity.
- Tolerance (tol): It provides the precision for solver convergence. For more precise solutions, lower values like [0.0001, 0.0002, 0.0003, 0.0005] were used to ensure that the solver comes to a stop at an optimal point.
- Intercept-scaling: When features are normalized, it adjusts the intercept term enabling proper weighting. To account for variations in feature importance, the values [1, 2, 3, 4, 5] were incorporated.

The optimal values obtained were C=1, intercept_scaling=1, tol=0.0001 and the best cross-validation accuracy based on these parameters is 0.90. This is the same as the baseline model. This implies that tuning did not enhance general performance on cross-validation folds.

4. Update the model with tuned parameters for testing and training

The baseline logistic regression model is updated with the tuned parameters. The updated model's logistic regression accuracy is 0.89 which is greater than the baseline logistic regression accuracy of 0.88. This shows that although there was no improvement in cross-validation accuracy, a higher testing accuracy was still observed. The enhanced performance on unseen data can be attributed to the updated model being less prone to overfitting.

(C) SVM

Support Vector Machine (SVM) is a supervised machine learning algorithm often used for classification and regression tasks. It works by finding the optimal hyperplane that separates data into classes. SVM is particularly effective in high-dimensional spaces and with datasets where clear class boundaries exist.

**Key Steps and Results**

1. Training and Testing SVM
    - The SVM model was instantiated with default parameters. It was then trained on the training dataset and evaluated on the testing dataset.
    - The baseline SVM model achieved the following accuracies:
        - **Training Accuracy:** 94%
        - **Testing Accuracy:** 91%

2. Cross-Validation
    - A 5-fold cross-validation was performed to evaluate the model's consistency and generalization ability.
    - The cross-validation accuracy scores ranged between 0.89 and 0.93, with an average accuracy of 0.91, indicating a robust performance across different folds.

3. Hyperparameter Tuning
    - GridSearchCV was employed to optimize SVM hyperparameters, including:
        - **C:** Regularization parameter controlling the trade-off between maximizing margin and minimizing classification error. Values: [0.01, 0.1, 1, 10, 100].
        - **Kernel:** Determines the SVM decision boundary (linear, polynomial, RBF). Values: ['linear', 'poly', 'rbf'].
        - **Gamma:** Influences the curvature of the decision boundary. Values: ['scale'].
    - The best parameters obtained were:
        - **C:** 100
        - **Kernel:** RBF
        - **Gamma:** Scale
    - The tuned SVM model achieved the following:
        - **Best Cross-Validation Accuracy:** 92%

4. Updated Model Performance
    - Using the optimal hyperparameters, the SVM model was retrained and re-evaluated:
        - **Training Accuracy (Tuned):** 98%
        - **Testing Accuracy (Tuned):** 93%
    - The improved accuracy suggests the updated model is better suited for the dataset, with reduced bias and variance.

5. Prediction on New Data

- A sample data point was standardized and classified using the tuned SVM model.
- The predicted class for the new data was "Normal", demonstrating the model's ability to generalize to unseen inputs effectively.

(D) RANDOM FOREST

This project employs Random Forest, an ensemble method. It improves model performance by combining the outputs of multiple decision trees. Every decision tree is created by utilizing a random subset of the data set to measure a random subset of features in every partition. The randomness introduces variability amidst individual trees which in turn decreases the likelihood of overfitting and enhances generalization.

**Key Steps and Results**

1. Base Decision Tree Implementation

1. Model Selection
    - A DecisionTreeClassifier was used as a baseline model to assess initial performance.
2. Evaluation
    - The model's accuracy was evaluated on both the training and testing datasets to understand its performance and potential overfitting.

2. Hyperparameter Tuning

1. Tuning Process
    - GridSearchCV was used to perform an exhaustive search for optimal hyperparameter combinations to maximize model performance.
2. Parameters Tuned
    - Key parameters included:
        - **n_estimators:** Number of trees in the forest.
        - **max_features:** Number of features considered for splitting at each node.
        - **max_depth:** Maximum depth of each decision tree.
3. Application
    - The best parameter combination was identified and applied to create a new, optimized Random Forest model.

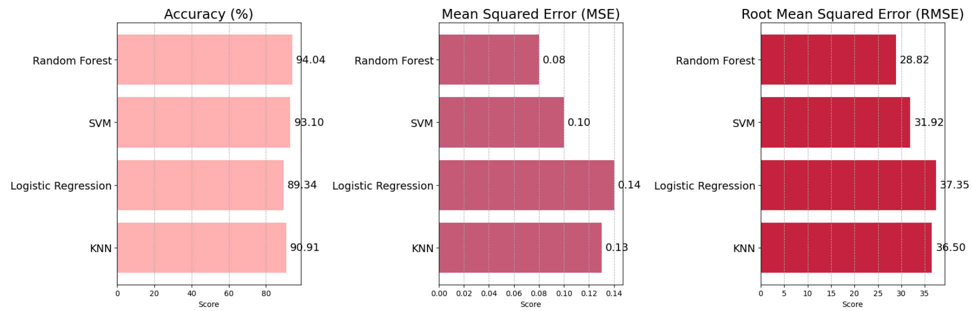
3. Performance Metrics

1. Accuracy Measurement
    - Training and testing accuracy scores were recorded for all models: the Decision Tree model, the base Random Forest model, and the tuned Random Forest model.
    - The optimized Random Forest model attained the largest accuracy for testing with 94.67% while having a near-perfect training accuracy of 99.79%. This signifies enhanced generalization in comparison to the baseline models.

## COMPARISON BETWEEN MODELS

### 1. Accuracy and Error

Random Forest performs best, it likely benefits from its ensemble nature, which reduces variance. While SVM has high accuracy it has slightly higher errors compared to Random Forest. Logistic Regression has lower accuracy and higher error metrics, which could be due to non-linear relationships that Logistic Regression cannot model well. KNN has decent accuracy, but its RMSE indicates it may not generalize well (performance could depend heavily on the choice of parameters)



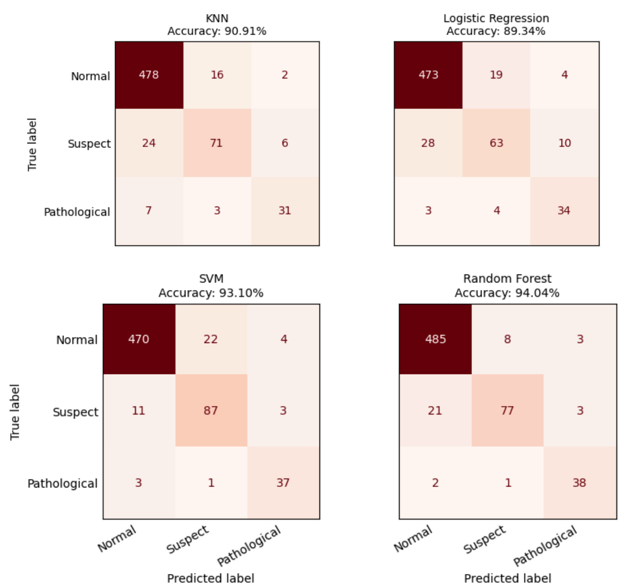

### 2. Confusion Matrix

Since this is a classification model we need to look at the confusion Matrices to see how the predicted labels fare against the actual labels.

**Normal**: All models perform well in predicting the Normal class

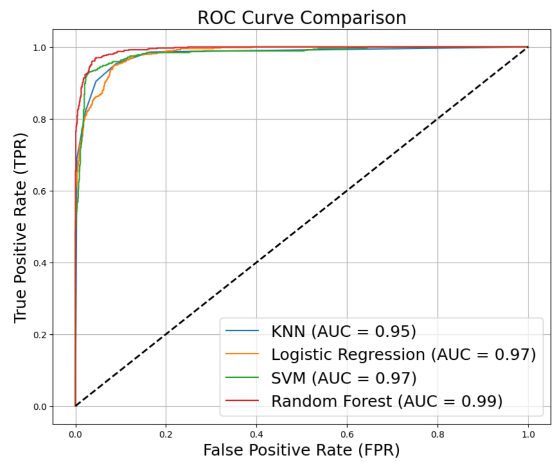
**Suspect**: This class is more challenging for models, with higher false negatives. Random Forest and SVM perform better here, with fewer misclassifications.

**Pathological**: Misclassifications are common due to imbalanced data or overlap between classes. Random Forest performs best, with only 3 errors.



### 3. Precision, Recall, and F1 Score

For our use case detection of Suspect and Pathological cases is crucial and we have to prioritise recall. Random Forest and SVM have the highest values for precision.
We would look at Precision if the cost or effort of investigating a false positive is costly. Here again, Random Forest and SVM perform the best.


### 4. ROC Curve and ROC-AUC Score

Since the goal is to maximize recall, we are looking for the ROC curve to be close to the upper-left corner. The ROC curve for Random Forest is doing the best in this respect which is also reflected in the AUC score, followed by SVM, KNN, and Logistic Regression.

## <u>CONCLUSION</u>

In conclusion, the evaluation of the four comparison metrics indicates that **Random Forest** is the most suitable classification model for accurately categorizing fetal health into the **'Normal,' 'Suspect,' and 'Pathological'** classes.

**APPENDIX**

**Citation**

Ayres de Campos et al. (2000). *SisPorto 2.0: A Program for Automated Analysis of Cardiotocograms*. J Matern Fetal Med, 5:311-318. [Link to article]

**License**



**Visual Credits**

- Splash Banner: Photo by Aditya Romansa on Unsplash.
- Splash Icon: Icon by Freepik, available on Flaticon.

**Data Dictionary**

| Variable | Description |
|---|---|
| baseline value | Average heart rate of the fetus when it is resting/not actively moving |
| accelerations | Count of times fetal heart rate increases per second |
| fetal_movement | Count of times the fetus moves per second. |
| uterine_contractions | Count of uterine contractions per second |
| light_decelerations | Count of small temporary decreases in fetal heart rate/sec |
| severe_decelerations | Count of severe decelerations per second |
| prolongued_decelerations | Count of prolonged decelerations per second |
| abnormal_short_term_variability | Percentage of time fetal heart rate changes drastically over a short time frame. |
| mean_value_of_short_term_variability | The average value of short-term changes in fetal heart rate |
| percentage_of_time_with_abnormal_long_term_variability | Percentage of time fetal heart rate changes drastically over a longer time frame |
| mean_value_of_long_term_variability | The average value of long-term changes in fetal heart rate |
| histogram_width | Width of the histogram for fetal heart rate |
| histogram_min | The minimum value of the histogram for fetal heart rate |
| histogram_max | The minimum value of the histogram for fetal heart rate |
| histogram_number_of_peaks | Number of peaks in histogram for fetal heart rate |
| histogram_number_of_zeroes | Number of zeros in the histogram for fetal heart rate |
| histogram_mode | Mode of the histogram for fetal heart rate |
| histogram_mean | Mean of the histogram for fetal heart rate |
| histogram_median | The median of the histogram for fetal heart rate |
| histogram_variance | Variance of the histogram for fetal heart rate |
| histogram_tendency | Tendency of the histogram for fetal heart rate |
| fetal_health | Class label for fetal health: 1:Normal, 2:Suspect, 3: Pathological |