Title: FETAL HEALTH CLASSIFICATION

Author: Alperen Unal

1. Introduction

The Fetal Health Classification project is an initiative directly aligned with the United Nations' Sustainable Development Goals, specifically focusing on the critical goal of reducing infant and maternal mortality rates by the year 2030. Utilizing data from Cardiotocograms (CTGs)—non-invasive tools that monitor fetal heart rates and uterine contractions—this project aims to harness advanced machine-learning techniques to predict fetal health conditions effectively. The overarching goal of the project is to empower healthcare professionals with accurate and actionable insights, facilitating timely medical interventions that could prevent potential adverse outcomes in pregnancy by analysing various indicators of fetal well-being, such as heart rate patterns and movements, which are encapsulated in the dataset, the project endeavours to classify fetal health into three distinct categories: Normal, Suspect, and Pathological. This classification is pivotal for early diagnosis and intervention, significantly contributing to the reduction of preventable deaths in infants and mothers, especially in under-resourced settings..

- **Objective**: Utilize Cardiotocograms (CTGs) to predict fetal health conditions effectively.

1.1. Dataset Overview

Source: Data obtained from a publicly available Kaggle dataset.
Dataset Link: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification
Features Explained: Detailed description of each feature, including heart rate, uterine contractions, and various types of decelerations.
Target Variable: Fetal health classified into three categories - Normal, Suspect, and Pathological.

1.2.Data Importing and Libraries

Libraries Used: Pandas for data manipulation, Matplotlib and Seaborn for visualization, Scikit-learn for modelling, TensorFlow and Keras for deep learning frameworks.
Data Loading: Step-by-step loading of the dataset into a DataFrame.

2. Data Cleaning and Pre-processing

2.1.Handling Missing Values

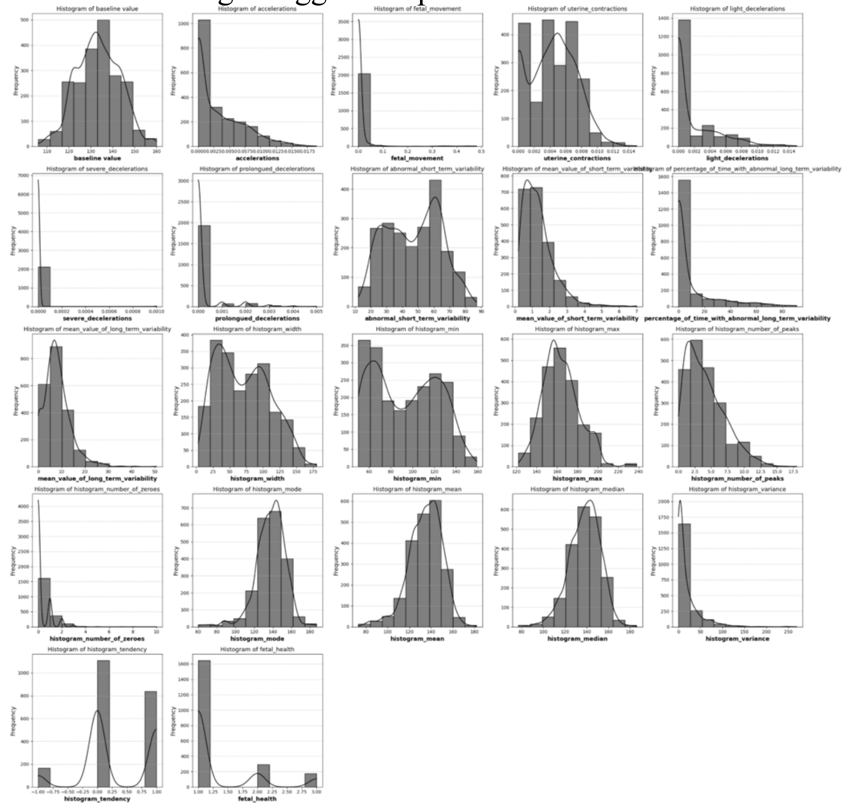
Assessment: The first step in data cleaning involved a thorough examination of the dataset for any missing values across all features..
Techniques Used: Utilized the isna() function in Pandas to check for null values across each column.
Outcome: The dataset was found to contain no missing entries.
Dealing with Duplicates
Identification: The duplicated() function of Pandas was employed to scan the dataset for any duplicate rows based on all columns.

Action Taken: Duplicate entries were removed using the drop_duplicates() function, ensuring that each data entry in the dataset was unique. This step helps in maintaining the integrity of the dataset and ensuring that the statistical analysis and model training are based on genuine data points.

2.2.Outlier Detection and Management

Statistical Analysis: Used statistical methods to identify outliers. The .describe() function provided a summary statistics table, highlighting key metrics like mean, standard deviation, and percentile distributions which help in spotting outliers.
Visual Inspection:

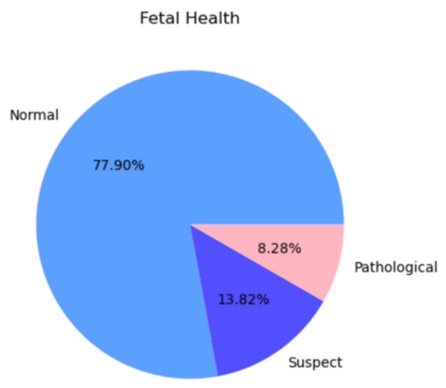
- **Histograms**: Created for each feature to understand the distribution of the data. This approach helps in recognizing any asymmetry in the distribution, such as skewness, which might suggest the presence of outliers.



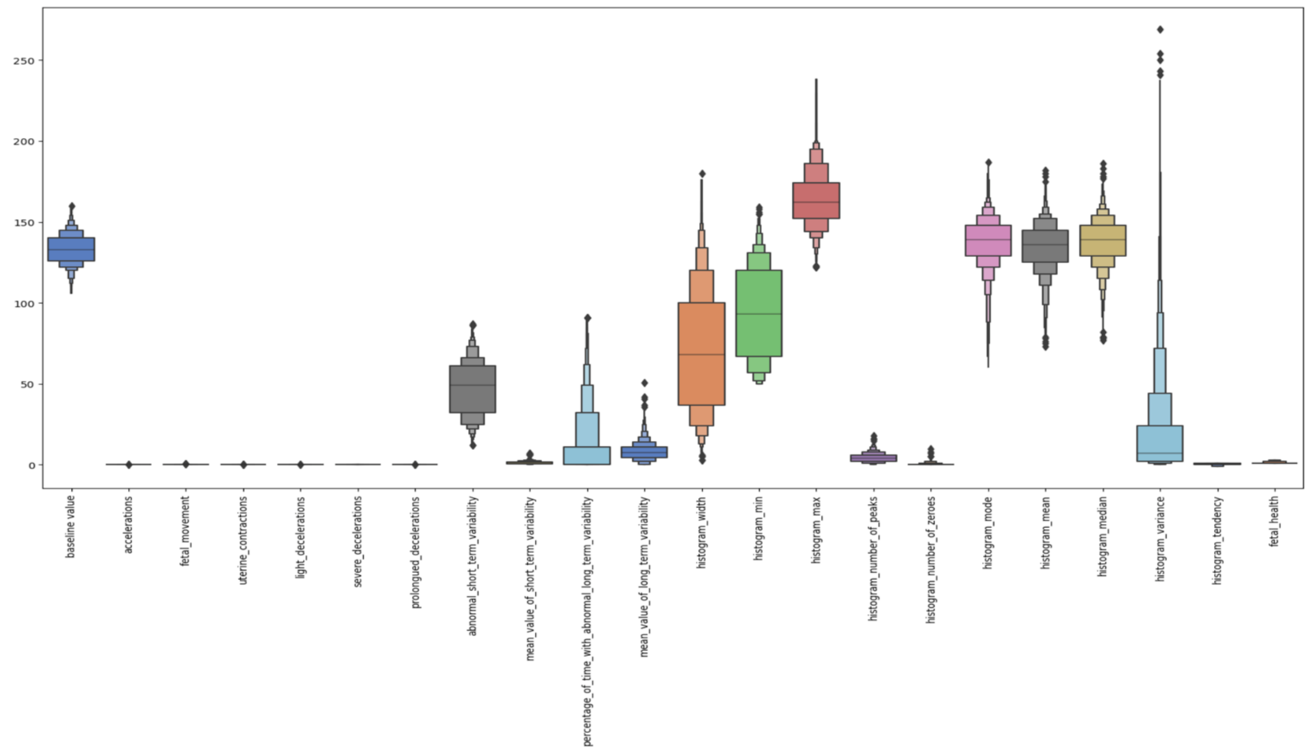
Handling Strategy: Considering the medical significance of the dataset, it was decided not to remove the outliers. In medical datasets, outliers may represent rare but important cases that could be crucial for accurate predictions. Instead of removal, these outliers were managed through robust scaling and transformation techniques in the pre-processing stage to minimize their impact on the predictive models.

3. Exploratory Data Analysis (EDA) and Pre-processing

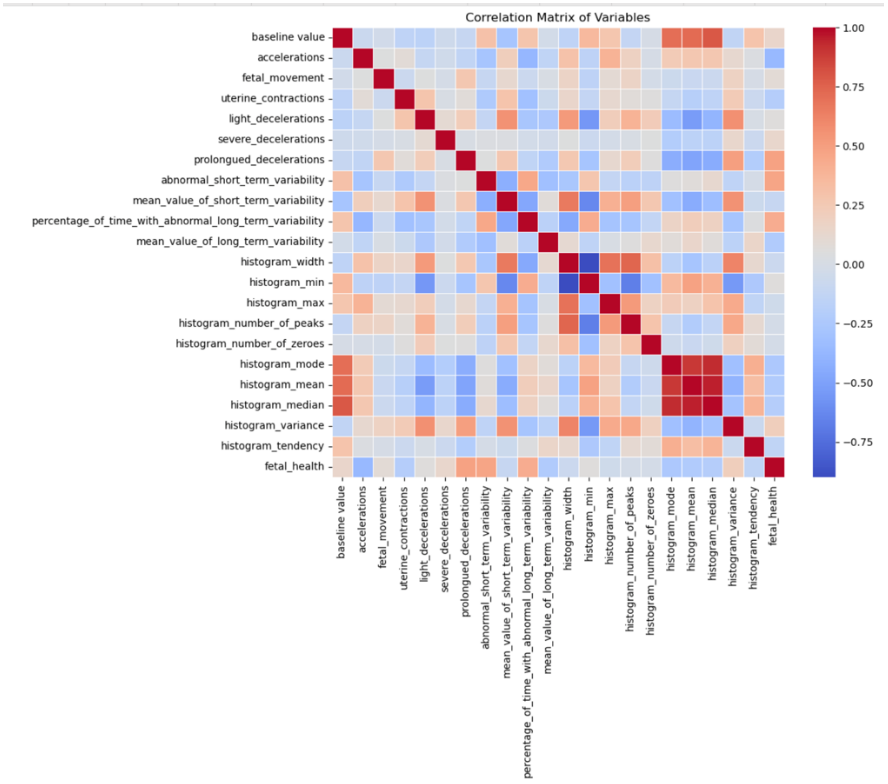
3.1.Distribution Analysis



Objective: To understand the distributions of various features within the dataset and identify any irregularities or patterns that could influence model performance.
Histograms: Employed to examine the frequency distribution of all numerical features. This helps in identifying the shape of the distribution (e.g., normal, skewed) and detecting any anomalies like unexpected spikes.
Box Plots: Used to assess the spread and central tendency of the data visually.



Correlation Analysis:
Heatmaps: Created to visualize the correlation matrix of the features. Heat maps help identify relationships between different features and the target variable, highlighting both positive and negative correlations.



Insights Gained: This analysis is crucial for recognizing features that might have a strong association with fetal health conditions, which can guide feature selection and model architecture decisions later on.

3.2.Feature Relationships

Scatter Plots: Generated to explore the relationships between pairs or groups of variables. This step is particularly useful for identifying patterns or trends that could indicate relationships not captured by correlation coefficients.

3.3.Pre-processing for Modelling

3.3.1. Handling Imbalanced Data

Challenge: The target variable 'fetal_health' shows class imbalance, which could lead to biased model predictions favoring the majority class.
Stratified Splitting: This approach involves dividing the dataset into training and test sets in such a way that the percentage of samples for each class label is preserved as compared to the original dataset. This is particularly important in a healthcare setting like fetal health classification, where maintaining a representative distribution of classes is crucial for model accuracy and reliability.
Tool Used: The train_test_split function from Scikit-learn was enhanced with the stratify parameter set to the target variable. This parameter ensures that both the training and testing subsets have the same proportion of class labels as the original dataset, thereby mitigating the effects of class imbalance on model training.

3.3.2. Feature Scaling

Standardization: Applied to normalize the data features to a mean of zero and a standard deviation of one. This step is crucial for models that assume normally distributed data, like logistic regression and SVM.

4. Model Creating

Features and Target Setup: Features (X) are derived from scaled_healthDf, a dataset where all feature values have been standardized. The target variable (y) is taken from healthDf['fetal_health'], which classifies fetal health into three categories: Normal, Suspect, and Pathological.
Dataset Splitting Methodology: The data is divided into training and testing sets using train_test_split, applying the stratify=y parameter to ensure the proportional distribution of the target classes is consistent across both sets. This split allocates 20% of the data to testing, maintaining reproducibility with a random state of 42.
Verification of Data Splits: After splitting, the dimensions of the training and testing sets are confirmed:

- Training set: 1690 samples (Dimensions: 1690, 19)
- Testing set: 423 samples (Dimensions: 423, 19)

4.1. Softmax Regression

Model Setup and Training: Initialized and trained a logistic regression model configured for multi-class classification (multinomial) using lbfgs solver with max_iter=1000.

4.1.1. Initial Evaluation: Evaluated using training data:

- High accuracy for 'Normal' class (precision and recall ~0.95).
- Lower performance for 'Suspect' (precision and recall ~0.67) and 'Pathological' classes (precision and recall ~0.81).
- Overall accuracy stood at 90%.

4.1.2. Cross-Validation: Applied Stratified K-Fold with 5 splits to assess model stability:

- Consistent accuracy around 89%-90% across folds.
- Slight variations in class-specific metrics suggested areas for improvement, particularly for the 'Suspect' class.
- Precision and Recall Analysis: After applying One-vs-Rest approach for multiclass classification:
- High precision and recall for 'Normal'.
- Lower metrics for 'Suspect'.
- Better, yet imperfect metrics for 'Pathological'.

4.2. Decision Tree

Model Training:

Trained a DecisionTreeClassifier with nearly perfect accuracy on training data, suggesting potential overfitting.

4.2.1. Initial Evaluation:

- Achieved high precision and recall across all classes, with overall accuracy near 99.88%.

4.2.2. Cross-Validation:

- Conducted with StratifiedKFold (10 splits), showing reduced accuracy (~90.95%) and variable performance across classes. 'Suspect' class performance notably lower.

4.2.3. Precision-Recall Analysis:

- High precision and recall for 'Normal' class. 'Suspect' class struggled with lower metrics, while 'Pathological' showed strong but imperfect performance.

4.3.Random Forest

4.3.1. Model Training and Evaluation:

Trained a RandomForestClassifier with near-perfect accuracy on training data, indicating potential overfitting.

- Achieved perfect precision, recall, and F1-score for all classes on the training data, confirming overfitting.

4.3.2. Cross-Validation:

- Utilized 10-fold stratified cross-validation to assess model generalization.
- Slightly decreased performance compared to training data, with lower precision and recall for "Suspect" class.
-

4.3.3. Precision-Recall Analysis:

- Precision vs. Recall curve highlighted high precision and recall for "Normal" class, challenges in "Suspect" class detection, and trade-off in "Pathological" class identification.

4.4. Stochastic Gradient Descent (SGD)

4.4.1. Model Training:

- An SGDClassifier was used with log loss, demonstrating strong performance on 'Normal' but weaker on 'Suspect' and 'Pathological' categories.

4.4.2. Cross-Validation:

- 10-fold stratified cross-validation showed an overall accuracy of about 89%, with the 'Suspect' category still underperforming.

4.4.3. Precision-Recall Analysis:

- Precision and recall curves indicated high accuracy for 'Normal', challenges for 'Suspect', and improved but not perfect performance for 'Pathological'.

4.5. XGBoost

4.5.1. Model Training and Evaluation:

- Configured XGBClassifier with specific parameters to optimize for multi-class classification. Trained on re-encoded target labels (shifted to start from 0).
- Achieved near-perfect performance on the training set with precision, recall, and F1-score all around 99.88%, indicating possible overfitting.

4.5.2. Cross-Validation:

- Conducted extensive 20-fold stratified cross-validation to test generalization.
- Post-validation metrics showed slight declines but remained high with an accuracy of about 94.6%.
- Performance for 'Normal' remained high, while 'Suspect' and 'Pathological' classes showed some variability but generally good results.

4.5.3. Precision-Recall Analysis:

- Generated precision-recall curves for each class, illustrating high precision and recall for 'Normal', moderate decline for 'Suspect', and robust performance for 'Pathological'.

4.6. Support Vector Machine (SVM)

4.6.1. Model Setup and Initial Performance:

- An SVC model with a linear kernel was used in a OneVsRest configuration, demonstrating strong performance on training data, particularly for the 'Normal' category.
- Classification metrics indicated good precision and recall for 'Normal', but lower performance for 'Suspect' and 'Pathological' classes.

4.6.2. Cross-Validation:

- Employed 20-fold StratifiedKFold to evaluate generalization capabilities.
- Cross-validation results showed a slight decrease in overall performance with accuracy around 90%. 'Suspect' class continued to show lower precision and recall.

4.6.3. Precision-Recall Analysis:

- Precision and recall curves were generated for each class, revealing high performance for 'Normal', moderate for 'Suspect', and relatively good for 'Pathological' with some trade-offs at higher recall levels.
- This summary describes the SVM model's training, validation, and performance across different classes, highlighting its effectiveness in detecting 'Normal' cases and areas where performance could be enhanced for other categories.

4.7. Multilayer Perceptron (MLP)

4.7.1. Model Configuration and Training:

- **Initial Setup:** The MLP model includes layers with 64 and 32 neurons respectively, using ReLU activation, and a softmax output layer for multi-class classification.
- **Training Process**: The model was trained using a categorical cross-entropy loss function and the Adam optimizer. The training involved a 50-epoch run with a batch

size of 10, and the training set was further split into training and validation sets to monitor and prevent overfitting.

4.7.2. Training and Validation Results:

- **Training Performance**: On the training set, the model achieved high accuracy and F1 scores, reflecting strong performance, particularly for the 'normal' class. However, performance metrics like precision and recall indicated potential overfitting as evidenced by perfect scores.
- **Validation Performance**: The validation results were slightly lower compared to the training, indicating some generalization issues but still maintained high accuracy and F1 scores.

4.7.3. Overfitting Concerns:

A clear discrepancy between training and validation performance suggested overfitting. The model performed exceptionally well on the training data but showed reduced effectiveness on the validation set, particularly for the 'suspect' and 'pathological' classes.

4.7.4. Cross-Validation and Precision-Recall Analysis:

- **Cross-Validation:** Employing StratifiedKFold with 20 splits, the model's generalized performance was confirmed with slightly varying but generally high precision and recall across the classes.
- **Precision-Recall Curves:** These curves indicated that while the model could identify the 'normal' class with high accuracy (near-perfect precision and recall), it struggled more with the 'suspect' class and achieved moderate success with the 'pathological' class.

4.7.5. Dropout and Learning Rate Adjustments:

Dropout: Introduced to combat overfitting by randomly omitting subset features and weights during training, it did not significantly alter the model's tendency to overfit, as indicated by the performance metrics which remained high.

Learning Rate Modifications: Adjusting the learning rate did not yield significant changes in performance, suggesting the model's architecture and training data characteristics might play a more substantial role in the observed training dynamics.

4.7.6. L2 Regularization (Ridge):

Parameter Setting: l2_lambda=0.01.
Performance: The training and validation accuracies stabilized around 35%, indicating minimal learning beyond the initial improvement. Despite regularization, the model was unable to effectively distinguish between the classes.

4.7.7. L1 Regularization (Lasso):

Parameter Setting: l1_lambda=0.001.
Performance: Slightly better than L2, with a marginal increase in training accuracy to around 37%. However, the majority of predictions were biased towards one class, indicating a significant imbalance and ineffective learning.

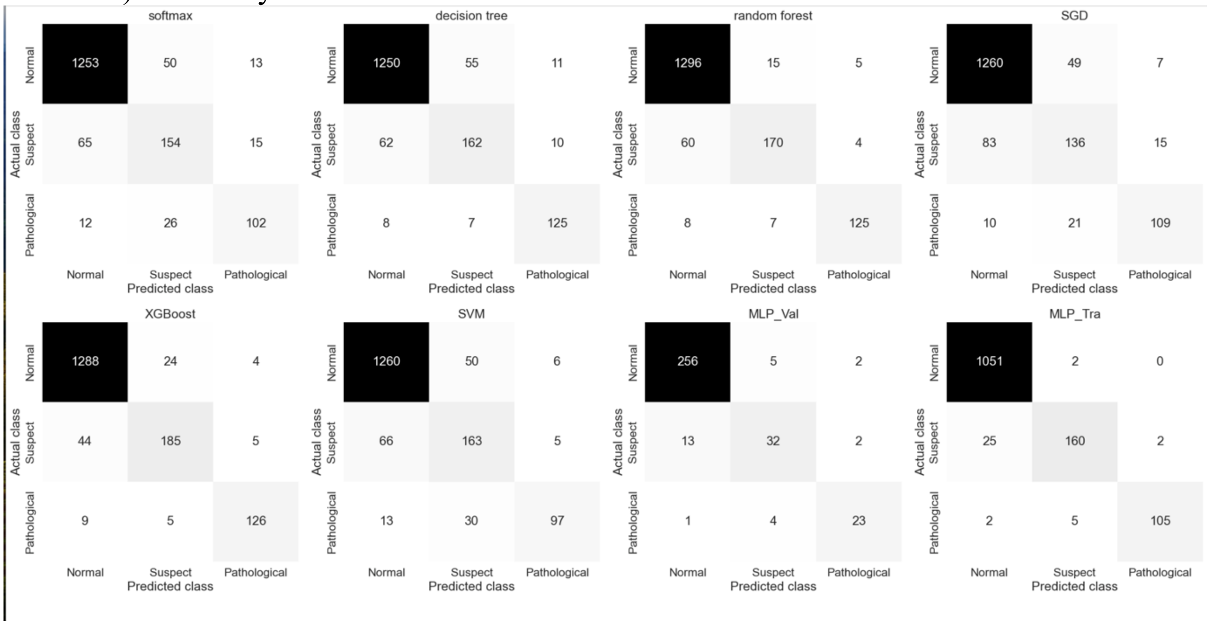
4.7.8. Early Stopping:

Configuration: Training stopped after 15 epochs with a restoration of the best weights to combat overfitting.
Performance: This model showed a slight improvement with a training accuracy of around 41%. However, the validation results were not significantly better, suggesting limited generalization to new data.

5. Evaluating Models - Summary Report

5.1. Overview:

This section of the project evaluates different machine learning models, focusing particularly on a comprehensive assessment of the MLP model in comparison to others such as Softmax, Decision Tree, Random Forest, SGD, XGBoost, and SVM. The evaluations leverage confusion matrices and a detailed examination of classifier performance metrics (Accuracy, F1, Recall, Precision) to identify the most effective model.



5.2. Comparative Performance:

The analysis began by visualizing confusion matrices for each model, which clearly demonstrated that the MLP model outperforms the others in terms of robustness and accuracy on the training set. This is complemented by the MLP's validation performance, confirming its efficacy.

Performance Metrics:

	Classifier	Accuracy	F1	Recall	Precision
0	Softmax	0.8929	0.8920	0.8929	0.8913
1	Decision Tree	0.9095	0.9089	0.9095	0.9085
2	Random Forest	0.9414	0.9392	0.9414	0.9397
3	SGD	0.8905	0.8875	0.8905	0.8855
4	XGBoost	0.9462	0.9452	0.9462	0.9449
5	SVM	0.8994	0.8985	0.8994	0.9000
6	MLP Validation	0.9201	0.9179	0.9201	0.9169
7	MLP Training	0.9734	0.9727	0.9734	0.9731

5.3.Advanced Optimization:

Further analysis with Grid and Random Search on the MLP model was conducted to optimize hyperparameters:

5.3.1. Grid Search

Showed a lower accuracy value (35.25%) with the best parameters being a configuration not surpassing the original MLP model.

5.3.2. Random Search

Yielded a better configuration with an accuracy of 91.77% but still did not exceed the baseline MLP model's performance.

After evaluating all models and attempting hyperparameter optimization, the initial MLP model remains the most effective. This MLP model not only showed superior accuracy on the training set but also demonstrated robustness and generalizability on the validation set, making it the standout model of the project.

5.4.Evaluation of the Best Model on the Test Set

The final test set evaluation of the MLP model shows an overall accuracy of 91.02%, with precision at 90.56%, recall at 91.02%, and an F1 score of 90.30%. The model performs exceptionally well for normal cases, with a precision of 92% and recall of 99%. However, it demonstrates lower efficacy in classifying suspect cases, with only 80% precision and 57% recall. Pathological cases are identified with good accuracy, though improvements are needed for optimal performance. This indicates that while the model is effective for most cases, it requires further tuning, especially for suspect conditions.

6. References

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly.

Ayres de Campos et al. (2000) SisPorto 2.0 A Program for Automated Analysis of Cardiotocograms. J Matern Fetal Med 5:311-318