**Fetal Health Classification**

CPSC 381 01 (SP24): Introduction to Machine Learning

Final Project Report

Bai Xiaoxue, Jiang Bei, Liu Yuchen

1. **Introduction**

The United Nations has identified the reduction of child and maternal mortality as a critical goal, aiming to decrease under-5 deaths to fewer than 25 per 1,000 live births by 2030. In areas with limited resources, where a significant proportion of maternal deaths occur, the use of accessible diagnostic tools such as Cardiotocograms (CTGs) can be transformative. CTGs assess fetal health by monitoring heart rates, movements, and uterine contractions, providing a cost-effective means for early detection of potential complications. This project focuses on using CTG data to classify fetal health into three categories—Normal, Suspect, and Pathological—to enhance prenatal care and prevent mortality, thereby addressing a significant gap in healthcare in underserved areas.

Our approach to tackling this issue involves a multi-step process starting with data preprocessing to address missing values and normalize the data, ensuring the scale of different features does not skew the model's accuracy. The dataset is then divided into training and test sets to evaluate the performance of various machine learning models including Logistic Regression, Decision Trees, Gradient Boosting, Random Forest, KNN, and SVM. By employing pipelines for each model, we streamline the process of fitting and making predictions, assessing their performance through cross-validation to ensure reliability and independence from any particular data split.

One of the key challenges is the imbalanced nature of the dataset, which has a higher proportion of normal cases compared to suspect or pathological ones. To counteract potential model bias towards the majority class, we might need to implement techniques like oversampling the minority class or adjusting class weights. Another significant challenge involves selecting appropriate features and model hyperparameters, which we address through exploratory data analysis and the use of GridSearchCV for optimizing parameters based on cross-validation scores.

The final phase of the project involves evaluating the chosen model on the test set using metrics such as accuracy, precision, recall, and F1 score. The interpretation of confusion matrices will further illuminate the model’s strengths and weaknesses in classifying fetal health. With a dataset sourced from Kaggle comprising 2126 records from CTG exams and classified by expert obstetricians, we have a robust basis for our models. This dataset is not only comprehensive but also highly relevant to our goal of enhancing prenatal care, especially in resource-limited settings.

Success in this project will be quantified by several performance indicators such as model accuracy, F1 score, and area under the curve (AUC). A high accuracy rate and an AUC close to 1.0 would signify an excellent performance, crucial for the effective and reliable classification of fetal health necessary for reducing maternal and child mortality in accordance with global health objectives.

1. **Data**

The dataset, sourced from Kaggle, comprising 2126 records from Cardiotocogram (CTG) exams forms the foundation of our project to classify fetal health into three categories: Normal, Suspect, and Pathological. Each record in this dataset encapsulates various features extracted from CTG exams, which have been meticulously classified by three expert obstetricians into the aforementioned categories. The inclusion of expert classifications provides a robust ground truth for training and validating our classification models.

The dataset is ideal for our project due to its comprehensive range of fetal health indicators, such as heart rate, uterine contractions, and movements. This allows for a detailed analysis of fetal well-being. Its direct relevance to reducing child and maternal mortality through early intervention aligns with our goal of improving prenatal care, particularly in resource-limited settings. Additionally, the dataset's size and variety offer a robust foundation for training diverse and effective machine learning models, enhancing the project's applicability and potential impact.

1. **Methodology**

To tackle the problem of fetal health classification using CTG data, we adopted a systematic approach employing various machine learning models. Each model was selected based on its ability to handle classification tasks effectively. The models included Logistic Regression, Decision Tree, Gradient Boosting, Random Forest, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). Here, we delve into the specifics of each model's functionality, advantages, and challenges.

* **Logistic Regression**

**Function Class & Objective**:

Logistic Regression is a linear model for binary and multi-class classification problems. It estimates probabilities using a logistic function, with the objective to find the best parameters β that minimize the cost function, which is typically the log-likelihood function for logistic regression

*Logistic Function：*

*Objective Function：*

**Features & Parameters**: Key parameters include regularization strength (**C**) and the type of solver used for optimization (e.g., **liblinear**, **sag**, **lbfgs**). Features are selected based on their correlation with the target variable, considering multicollinearity.

**Advantages**: Simplicity and efficiency for smaller datasets with fewer features.

**Disadvantages**: Prone to underfitting and may not perform well on non-linear problems or complex relationships.

* **Decision Tree**

**Function Class & Objective**: Decision Trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. The nodes are split based on certain criteria like Gini impurity or entropy in information gain, aiming to create homogenous subsets.

*Gini Impurity*:

where is the probability of an object being classified to a particular class.

The goal in each split is to decrease the Gini impurity. A perfect classification (all elements are of the same class) leads to a Gini impurity of 0.

*Entropy (Information Gain)*:

where S is the set of samples and is the proportion of the samples that belong to class i.

Information gain is then calculated as the decrease in entropy caused by splitting on a given feature. The information gain IG for a feature A is defined as:

where are the different values that A can take, and ​ is the subset of S for which feature A has value v.

**Features & Parameters**: Parameters include the depth of the tree (**max\_depth**), the minimum number of samples required to split an internal node(**min\_samples\_split**), and the minimum number of samples required to be at a leaf node (**min\_samples\_leaf**).

**Advantages**: Easy to understand and interpret; can handle both numerical and categorical data.

**Disadvantages**: Prone to overfitting; sensitive to noisy data and outliers.

* **Gradient Boosting**

**Function Class & Objective**: Gradient Boosting is an ensemble technique that builds models sequentially, each correcting its predecessor, thus decreasing bias. It uses a gradient descent algorithm to minimize the loss when adding new models. Each new model is a weak learner added in a way that improves the overall model's accuracy.

*Initialize the model*:

*For m=1 to M (M is the number of boosting stages)*:

*Compute pseudo-residuals*

*Fit*

*Find*

*Update*

*Output*

**Features & Parameters**: Key parameters include the number of boosting stages (**n\_estimators**), the depth of individual trees (**max\_depth**), and the learning rate (**learning\_rate**).

**Advantages**: Highly effective on a wide range of problems, including both bias and variance.

**Disadvantages**: Can be computationally expensive and prone to overfitting if not tuned properly.

* **Random Forest**

**Function Class & Objective**: Random Forest is an ensemble of Decision Trees, typically trained via the bagging method. The general objective is to improve the predictive accuracy and control over-fitting. It builds numerous decision trees at training time and outputs the class that is the mode of the classes of the individual trees.

*Initialize: Start with B empty trees*

*For b = 1 to B:*

*Generate a bootstrap sample of size N from the training data*

*Build a tree using :*

*While the node n is not a leaf:*

*Randomly select m features from the total M features.*

*Determine the best split using these m features.*

*Split the node into daughter nodes based on the best split.*

*For a new data point x, predict using:*

*Classification:*

*Regression:*

**Features & Parameters**: Important parameters include **n\_estimators** (number of trees in the forest), **max\_features** (the size of the random subsets of features to consider when splitting a node), and **max\_depth**.

**Advantages**: Reduces variance compared to regular decision trees, less likely to overfit.

**Disadvantages**: Slow real-time prediction, difficult to implement, and complex.

* **KNN**

**Function Class & Objective**: A non-parametric method used for classification and regression. Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. Distance metrics such as Euclidean, Manhattan, or Hamming distance are used to find the closest training examples.

**Features & Parameters**: Main parameter is K, the number of nearest neighbors to consider. It is crucial to scale features because KNN is sensitive to the range of data points.

**Advantages**: Simple and effective, no assumptions about data.

**Disadvantages**: Computationally expensive as it stores all training data, performance degrades with high dimensionality (curse of dimensionality).

* **SVM**

**Function Class & Objective**: SVM is a powerful classifier that works by finding a hyperplane that best divides a dataset into classes. The objective function of SVM is to find the hyperplane that maximizes the margin between the closest points of different classes, which are called support vectors.

*Objective function*:

*subject to*

**Features & Parameters**: Key parameters include **C** (regularization parameter), **kernel** type (e.g., linear, poly, rbf, sigmoid), and **gamma** (kernel coefficient).

**Advantages**: Effective in high dimensional spaces, memory efficient.

**Disadvantages**: Not suitable for larger datasets and does not perform well with lots of noise and overlapping classes.

Each model was implemented using Scikit-learn, a powerful Python library for machine learning. The models were evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Techniques such as cross-validation were employed to ensure the models' robustness. Hyperparameter tuning was conducted using GridSearchCV to optimize each model's performance.

1. **Implementation Details**

The section meticulously outlines the methodologies and analytical strategies employed in our fetal health classification project. It begins with the preprocessing of data, addressing missing values and normalization to ensure consistency across all features. We then describe the selection and setup of various machine learning models, detailing the rationale behind each choice and the specific configurations used. The section further delves into hyperparameter tuning through GridSearchCV and the use of cross-validation to validate model performance robustly.

* **Data Preprocessing**

In the data preprocessing stage, missing values were addressed either by removing the affected rows or imputing them based on the mean of the columns, contingent upon the nature and importance of the missing data. This ensures that the models operate on clean and reliable data, which is crucial for accurate predictions. Additionally, given the varied range of data points in the dataset, a StandardScaler was applied to normalize the feature set. This normalization process is pivotal for models such as Logistic Regression and Support Vector Machines (SVM), which are sensitive to variance in data scales, thereby ensuring consistent performance across all features.

* **Model Selection and Setup**

Separate pipelines were constructed for each model including Logistic Regression, Decision Tree, Gradient Boosting, Random Forest, KNN, and SVM. Each pipeline incorporated a standard scaler followed by the classifier, ensuring a clean and efficient workflow that facilitates easy adjustments and consistency in model training. The initial hyperparameters were set based on the defaults provided by Scikit-learn to establish a baseline performance for each model.

* **Hyperparameter Tuning and Model Optimization**

A comprehensive grid search was conducted for each model to find the optimal set of hyperparameters using GridSearchCV. This process iteratively tests combinations of parameters to maximize the model's performance based on cross-validation scores. For example, in the Gradient Boosting model, parameters such as learning rate, number of estimators, and tree depth were varied, testing values like learning rates of 0.05, 0.1, and 0.25, number of estimators at 100, 200, and 300, and max depths of 3, 5, and 7 to identify the best combination for maximizing predictive accuracy. Furthermore, k-fold cross-validation, with k equal to 10, was employed to ensure robust model evaluations and to provide a reliable estimate of model performance, surpassing the reliability of a single train-test split.

* **Model Training and Evaluation**

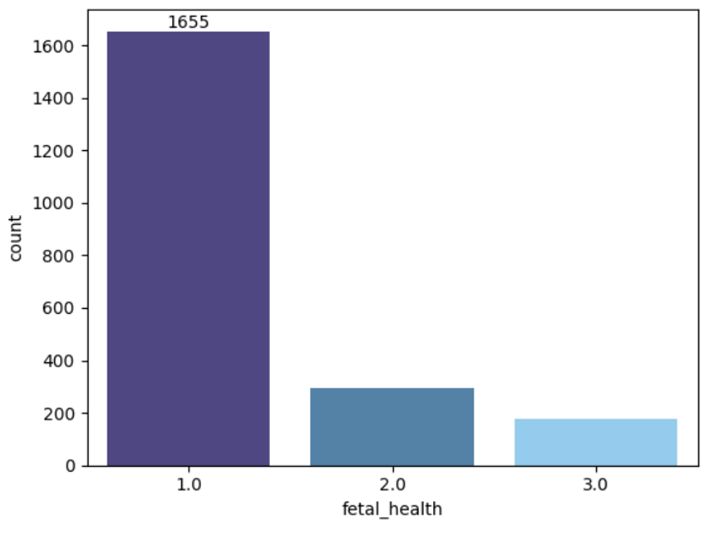
Each model was trained on the training set, which constitutes 80% of the entire dataset. The training process involves adjusting the model weights to minimize prediction errors using the specified hyperparameters. Post-training, models were evaluated based on several metrics including accuracy, precision, recall, F1-score, and AUC (Area Under the Curve). These metrics provide a comprehensive understanding of model performance, highlighting strengths in terms of both overall accuracy and class-specific performance.

* **Results Interpretation and Reporting**

Confusion matrices were generated for each model, which assisted in visualizing the true positives, false positives, true negatives, and false negatives. This matrix is especially useful in medical classification tasks to comprehend the model's capability in distinguishing between different classes of health conditions. Moreover, detailed classification reports were generated to provide insights into the precision, recall, and F1-scores for each class, helping to assess how well the model is performing for each category of fetal health—normal, suspect, and pathological.

1. **Results**

The dataset is heavily imbalanced, with nearly 78% of samples labeled as 'Normal.' The 'Suspect' and 'Pathological' classes constitute approximately 14% and 8% of the dataset, respectively, indicating a significant minority. A train-test split of 80% for training and 20% for testing was used to validate the models. The evaluation metrics chosen were accuracy, kappa, recall, precision, AUC (Area Under the Curve), and F1 Score. Accuracy provided a baseline measure of overall correctness, while kappa offered insight into the agreement between model predictions and actual outcomes, correcting for chance. Recall was particularly crucial in the context of medical predictions, as it measures the ability of the model to identify all relevant cases. Precision and AUC were also important, with AUC being valuable for its interpretation in the unbalanced classes that were initially present in the dataset. Finally, the F1 Score was employed as it combines precision and recall into a single metric, which is particularly useful when the class distribution is uneven.



**Fig 1. Distribution of fetal health situation**

Among the evaluated models in **Table 1**, Gradient Boosting Classifier (GBoost) performed best, achieving high accuracy and AUC. Logistic Regression and SVM models underperformed due to their limited ability to separate minority classes. Therefore, further tuning of the GBoost model was performed to improve the predictive performance.

**Table 1. Performance Metrics of ML Models Fitted on the *Unbalanced* Dataset**

|  | *Accuracy* | *Kappa* | *Recall* | *Precision* | *AUC* | *F1* |
| --- | --- | --- | --- | --- | --- | --- |
| **LR** | 0.8474 | 0.5333 | 0.6676 | 0.7420 | 0.8692 | 0.6966 |
| **DT** | 0.9014 | 0.7245 | 0.8092 | 0.8247 | 0.8569 | 0.8160 |
| **RF** | 0.9249 | 0.7845 | 0.8285 | 0.8770 | 0.9787 | 0.8499 |
| **GBoost** | 0.9272 | 0.7942 | 0.8550 | 0.8801 | 0.9546 | 0.8638 |
| **KNN** | 0.8944 | 0.6921 | 0.7496 | 0.8300 | 0.9042 | 0.7844 |
| **SVM** | 0.8545 | 0.5554 | 0.6388 | 0.7578 | 0.8898 | 0.6838 |

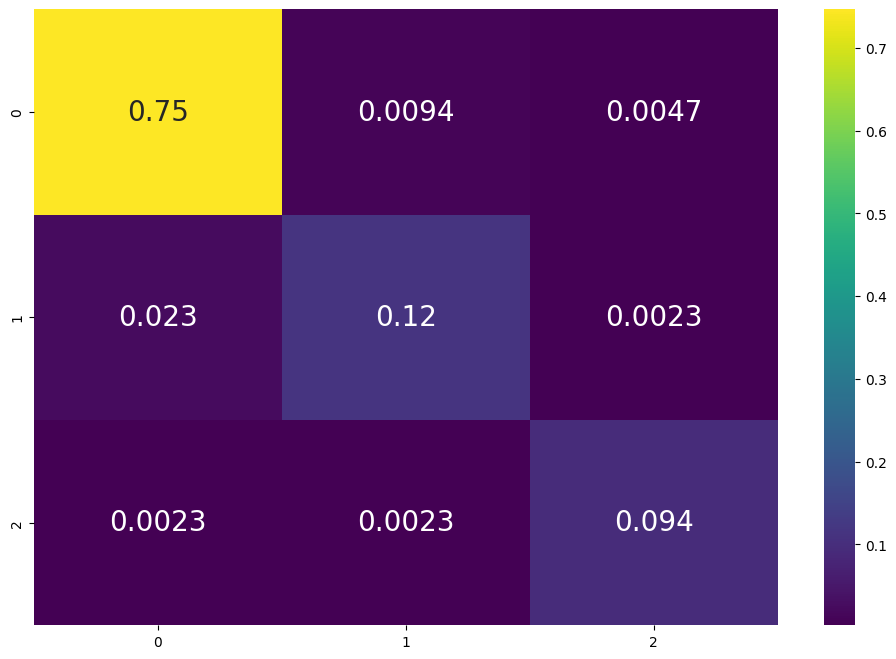
The tuning involved adjusting the hyperparameters of the GBoost model to obtain the optimal combination that yields the highest performance. After Grid Search Cross Validation, the final GBoost Classifier achieved the following performance metrics on the test dataset:

**Table 2. Performance Metrics of the Tuned Gradient Boosting Classifier.**

| *Class* |  |  | *Precision* | *Recall* | *F1* | *Support* |
| --- | --- | --- | --- | --- | --- | --- |
| **1(Normal)** |  |  | 0.97 | 0.98 | 0.97 | 324 |
| **2(Suspect)** |  |  | 0.91 | 0.82 | 0.86 | 60 |
| **3(Pathological)** |  |  | 0.93 | 0.95 | 0.94 | 42 |
| **Accuracy** |  |  |  |  | 0.96 | 426 |
| **Macro Avg** |  |  | 0.93 | 0.92 | 0.92 | 426 |
| **Weighted Avg** |  |  | 0.95 | 0.96 | 0.95 | 426 |

Separately, the high precision and recall scores indicate that the final model classifies the 'Normal' class effectively. Lower recall (0.82) suggests that some 'Suspect' instances are misclassified. However, the precision (0.91) remains high. The model performs well in identifying 'Pathological' cases despite the limited support. Overall, the tuned gradient boosting model achieves an accuracy of 0.96 and consistently high scores across all other metrics, indicating a well-balanced performance across all classes.

A confusion matrix provides insights into the performance of a classification model by detailing the counts of correct and incorrect predictions across classes. Rows represent actual classes (ground truth), columns represent predicted classes. We can find that the model performs well in predicting Class 1, with high precision and recall. For Class 2, 49 instances were correctly classified as 'Suspect' and 11 instances misclassified as 'Normal', which shows significant misclassification between these two classes. For Class 3, the model achieves excellent classification performance for 'Pathological' images, just 2 instances misclassified as 'Suspect' and 1 instance misclassified as 'Normal'.



**Fig 2. Confusion matrix**

1. **Conclusion**

The confusion matrix and performance metrics reveal that the tuned Gradient Boosting Classifier excels in classifying 'Normal' and 'Pathological' images while struggling to distinguish 'Suspect' images. Further improvements can be achieved by employing advanced data augmentation techniques, data upsampling or adjusting the model's hyperparameters to enhance the predictive accuracy for these minority classes.