Project Report
On
Fetal Health Classification

IE7275 – Data Mining in Engineering

Submitted by: Group 38
Team Members
Sai Shyam Nidadavolu (NUID 002792342)
Challagonda, Mrudula (NUID 002765266)

Term and Year: Spring 2023
Submitted to: Prof. Sagar Kamarthi

Submitted Date: April 20th, 2023

ACKNOWLEDGEMENT

Apart from the efforts of team, the completion of this project wouldn‟t have been possible without the inputs and encouragement of the Teaching Assistants. We also take this opportunity to express our gratitude to Professor Sagar Kamarthi. This wouldn‟t have been materialized without his support and the knowledge he shared with us throughout the semester.

Our heartfelt appreciation goes to Northeastern University – College of Engineering for givingus this opportunity to test our knowledge and use it for practical purposes.

Table of Content:

Fetal Health Classification

Project Setting

Healthcare Industry is developing at an exponential rate. Technology and healthcare industry are working hand in hand which is leading to reduced mortality rate. While this is true that the mortality rate is going down, there are still a few overlooked healthcare issues. Fetal mortality, that is, loss of the child during the gestational period is one among them. We plan to take up a dataset and analyze the fetal movement and predict the fetal health. Most of these cases bring out a trend that could be very useful for the medical professionals. Our aim is to target such cases where prevention is a possibility.
To understand the features available in the dataset we need to understand what the medical terminology is too. This dataset contains the Cardiotocography data – records with information of continuous heart rate of the fetus and their assigned fetus status by expert obstetrician.

Problem Definition

We plan on understanding trends in the heart rate that affect the health of the fetus over the gestational period. Analyze the heart rate mean, median and the peaks that have been noted and come up with a trend to classify and automate fetal health assessment.

Our dataset also has the time period during which peak heart rates have occurred, finding the relation between percentage of time during the gestational period with peak heart rates and the final status of fetal health is what we plan to target.

Once the model is built and fine-tuned, it can be used to predict the health status of new fetuses and help doctors and medical professionals make better-informed decisions about the health of the fetuses and the best course of treatment. The classification is going to be into the below mentioned three categories:

- Normal
- Suspect
- Pathological

Data Sources and Data Description

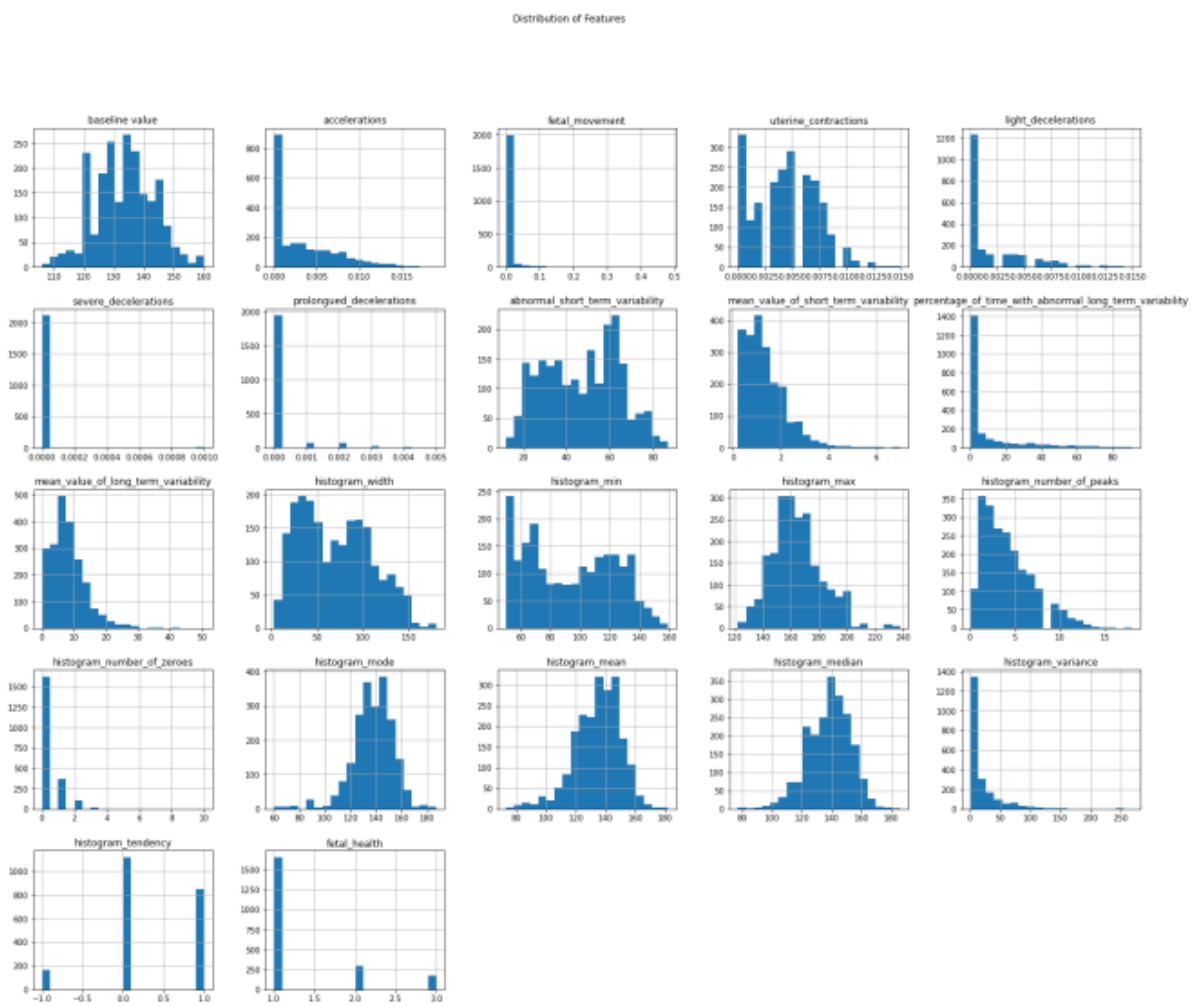
We obtained the data from Kaggle – Fetal Health Classification | Kaggle

Our data set contains 2127 records available with 22 columns. Information such as date of CTG examination, uterine contractions, fetal movement, gestational age, fetal heart rate, deceleration (change in heart rate) - light, severe, prolonged and abnormal.

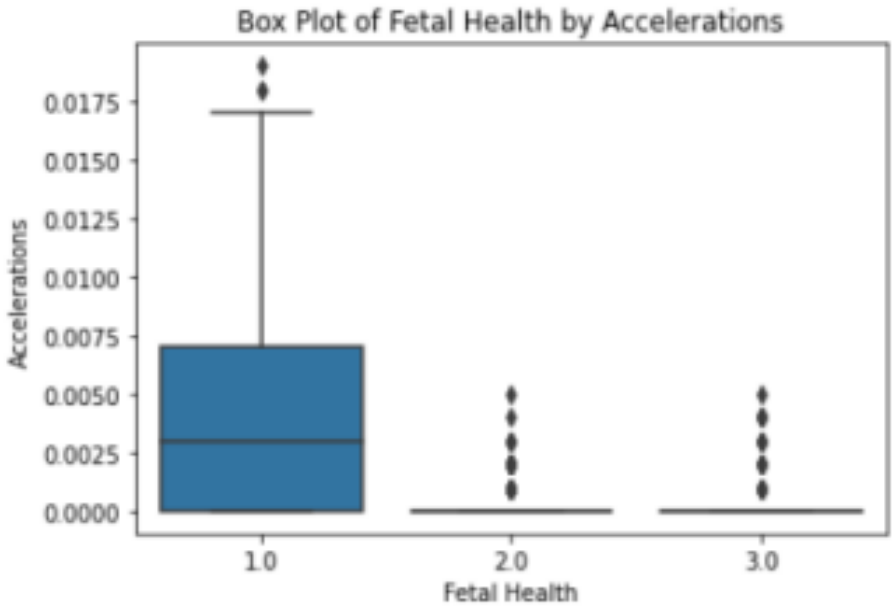
Column Name	**Type**
baseline value	Numerical
accelerations	Numerical
fetal_movement	Numerical
uterine_contractions	Numerical
light_decelerations	Numerical
severe_decelerations	Numerical
prolongued_decelerations	Numerical
abnormal_short_term_variability	Numerical
mean_value_of_short_term_variability	Numerical
percentage_of_time_with_abnormal_long_term_variability	Numerical
mean_value_of_long_term_variability	Numerical
histogram_width	Numerical
histogram_min	Numerical
histogram_max	Numerical
histogram_number_of_peaks	Numerical
histogram_number_of_zeroes	Numerical
histogram_mode	Numerical
histogram_mean	Numerical
histogram_median	Numerical
histogram_variance	Numerical
histogram_tendency	Numerical
fetal_health	Categorical This is the main classifying column – it has entries 1, 2 and 3 that map to the three classes i.e. Normal, Suspect and Pathological. This could be converted to categorical column too.

Data Exploration and Visualization

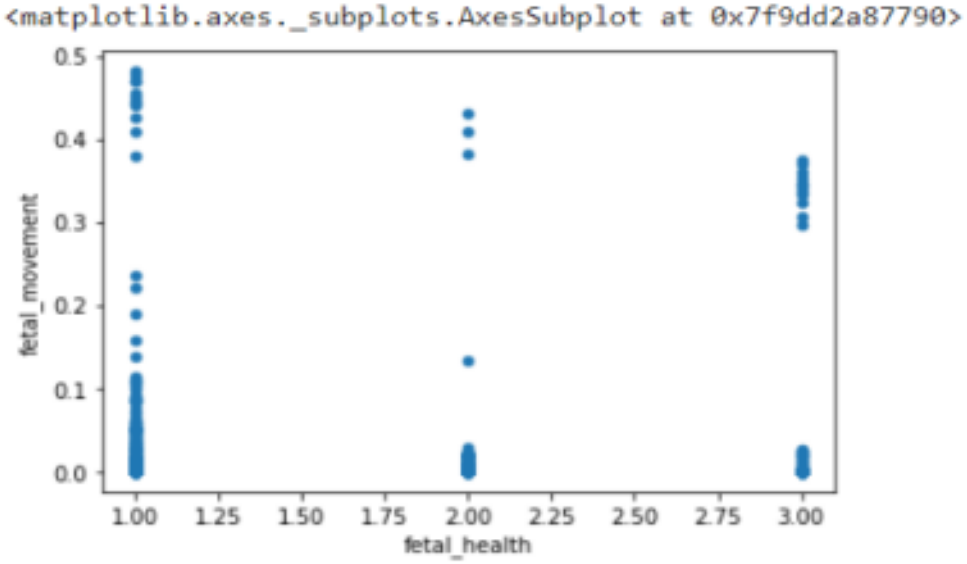
We performed Exploratory Data Analysis to understand the structure of our dataset. First step was to check for nulls and the data types. In all the 22 columns there wasn't a single entry with null value. And all 22 columns are of data type float.
Our main column is fetal_health, this column has 3 unique entries – 1, 2 and 3 that map to the three classes - Normal, Suspect and Pathological. We explored how each feature has an impact for each class. We started off with plotting a histogram for the "accelerations" column to understand the spread.



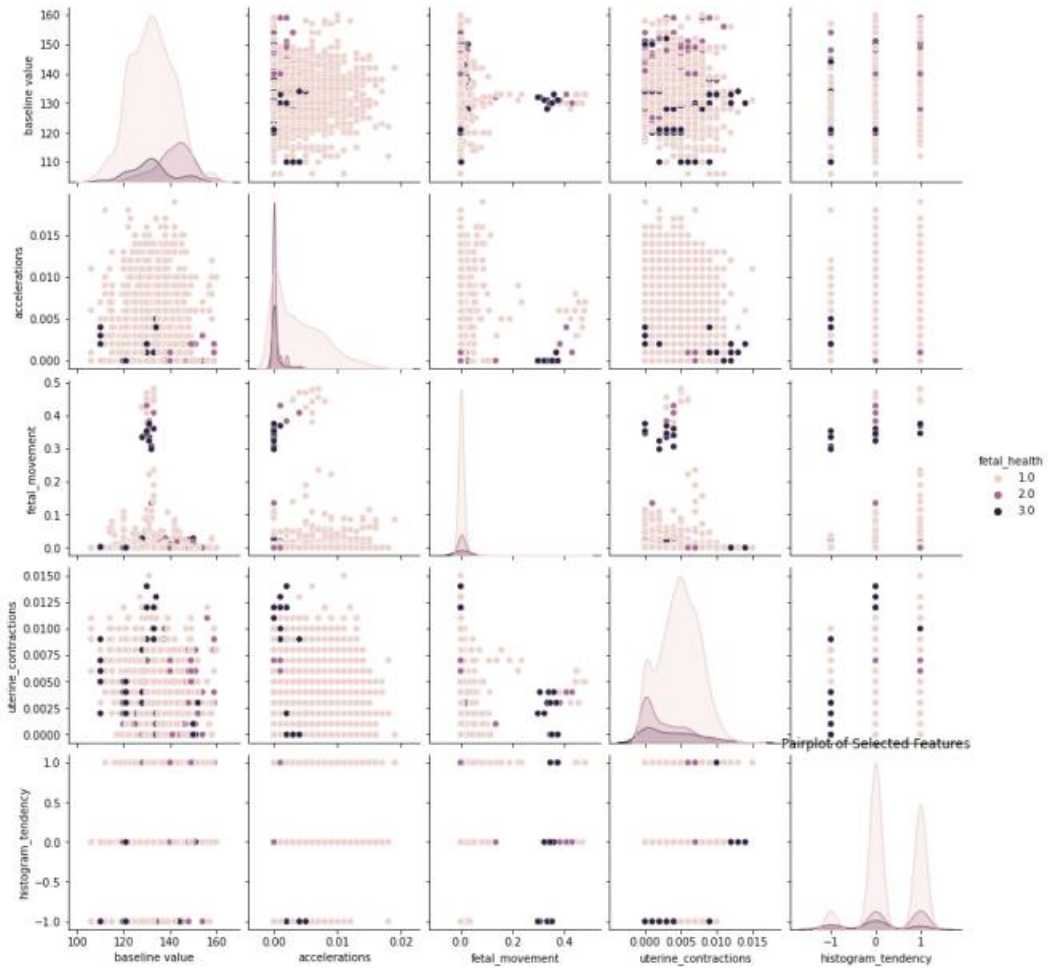
We also plotted a box plot to understand the spread and the presence of outliers. Below is the snippet of side by side box plots of all the three classes. Here the dots represent the outliers with respect to "accelerations" per second.



To understand the effect of fetal movement on the fetal health we plotted a scatter plot with x axis representing fetal health class and y-axis showing the fetal movement per second.

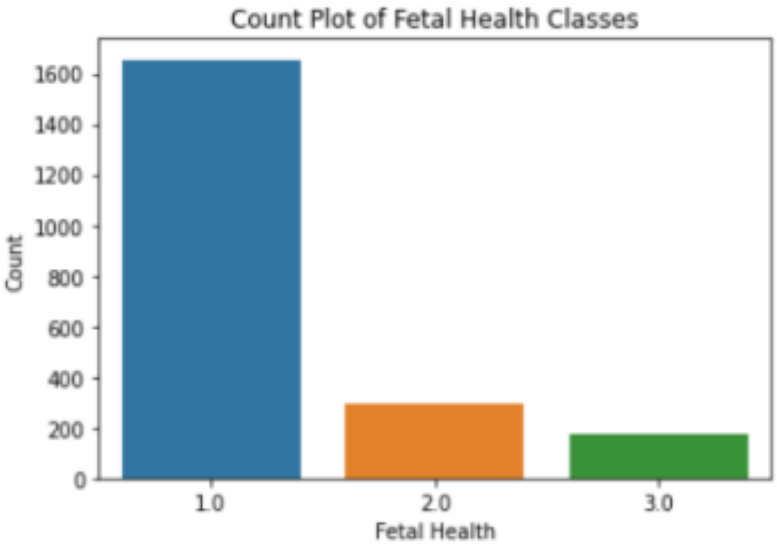


Here if you observe, the fetal movement in the range [0.1,0.25] per second falls only under the normal class. But this correlation does not guarantee causation. We will be performing further analysis to understand the causation.

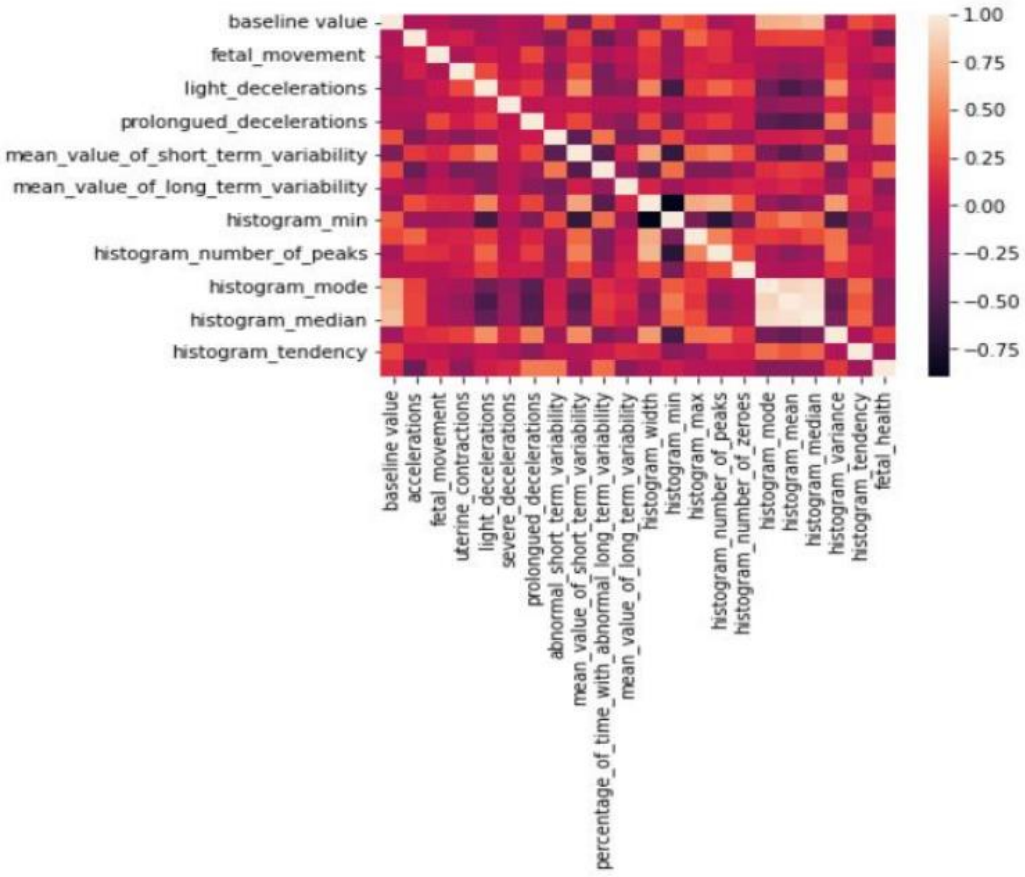


Then to understand the right set of features we plotted pair plot to understand the relationship among the variables. Below is the snippet of a the pair plot with the selected features as 'baseline value', 'accelerations', 'fetal_movement', 'uterine_contractions', 'histogram_tendency','fet al_health'.

To understand the percentage/count of records belonging to each of the class we plotted a count plot. It has been observed that majority of the records fall under the “normal” class which is denoted by “1”. Among the remaining records, “Suspect” class has more records compared to the “Pathological” class. Below is the snippet of the count plot.

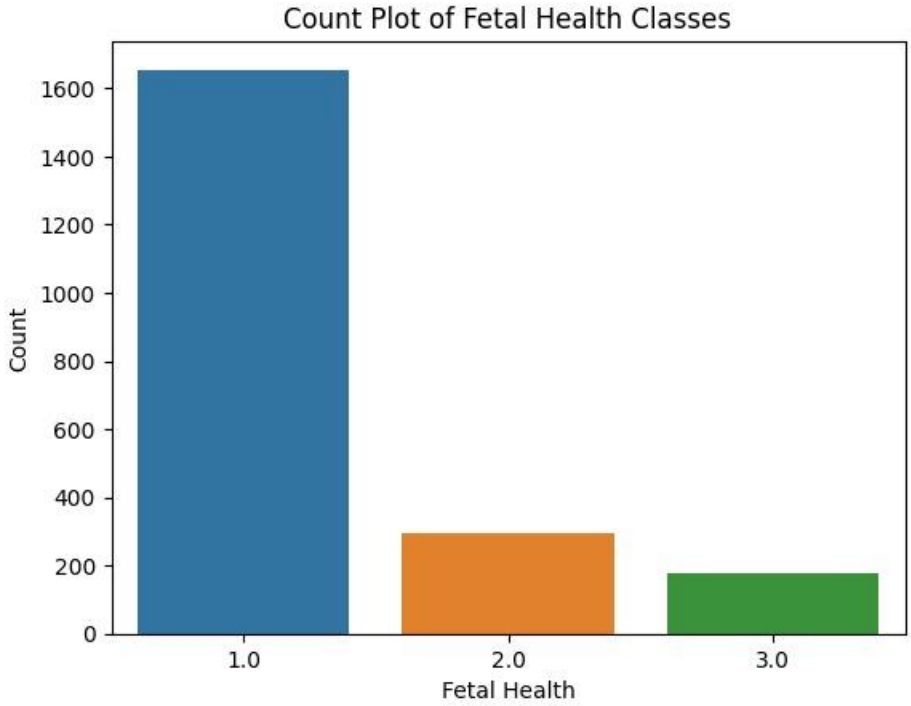


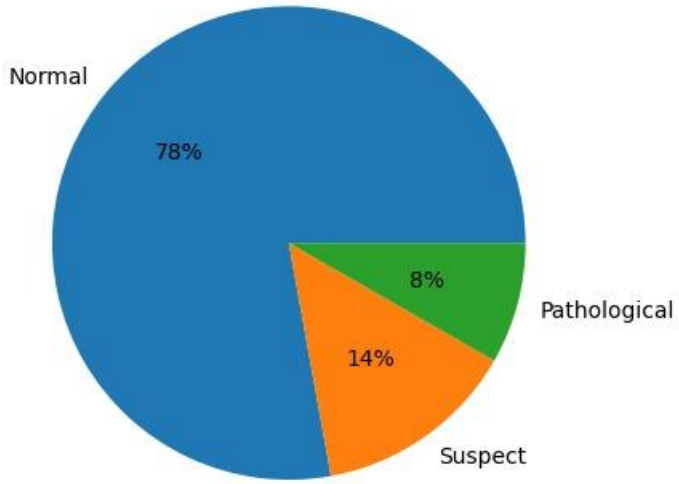
Further, we checked for correlation among the available columns too, to understand more about the features.



The heat map to visualize the above correlation matrix -

We also analyzed the distribution of the three categories in the dataset.

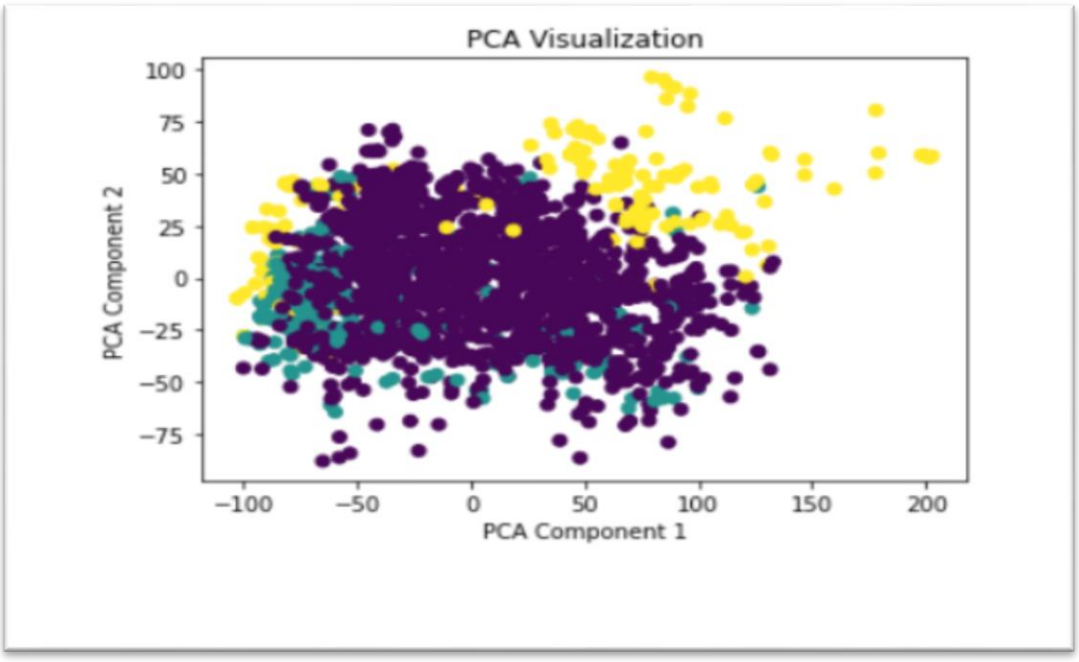




Dimensionality Reduction

As part of dimensionality reduction we performed principal component analysis. In this technique, we reduce the number of features to avoid the curse of dimensionality. In the fetal health dataset that we are working with, we have 21 features. We performed the technique to see if we can reduce the dimensionality and increase the speed up the training process of the machine

learning models that are to be implemented. Below is the snippet of a visualization where the 21 features have been plotted as part of 2 principal components.



Data Preparation

Post data exploration and visualization, it has been observed that the predictor variables show a large scale difference. To handle this and to make sure there is no bias towards any predictors we took measures to eliminate the scaling difference for which we standardized the data using StandardScaler() library from the scikit-learn package, to ensure that all predictor variables were given equal importance in terms of variability.

Data Partitioning

To implement the machine learning models, we first partitioned our dataset into 2 sets – training and testing. The predictor variables are pushed into X_train and X_test respectively. The target variables are pushed into Y_train and Y_test. The ratio we followed to split the original dataset is 75% and 25%. Post partitioning, for the training set has 1594 records and the testing data has 532 records

Data Mining models

The selection of suitable data mining models for fetal health classification depends on several factors, such as the size of the dataset, the number of features, the class imbalance and the performance metrics.

For datasets like fetal health classification, the most common data mining models are Random Forest Classifier, Decision Trees, Support Vector Machine, Logistic Regression, K-Nearest Neighbors (KNN).

Decision Tree Classifier:

Decision tree is a supervised machine learning algorithm which is used for classification tasks that builds a tree-like model of decisions and their possible consequences. This type of classifier can be used on both numerical and categorical data type. Decision Trees can handle class imbalance using different techniques such as over sampling, under sampling and can also handle missing values using imputation and surrogate splits.

The major disadvantage on using this model is, sometimes it may suffer from over fitting if the tree is too dep or if the training data is too noisy or contains irrelevant features.
This disadvantage can also be prevented using different techniques like early stopping, etc.

Steps involved in building a decision tree classifier is:

1) Data Preparation: The first is to prepare the data, this includes handling missing values, treating outliers, feature selections, encoding categorical variables and finally splitting the data into training, testing and validation datasets.
2) Build the Tree/ build the model: The model is built on training data, it involves a root node that includes all the training data. The model repetitively splits the data into smaller subsets based on the features which have been provided. The splitting process continues until all the training data is perfectly classified or until a stopping criterion is reached. One of the stopping criterion is minimum number of instances per leaf.
3) Evaluating the Tree/ Evaluating the model: Once the Tree is built, the next step is to evaluate the performance of the model on testing data. The steps include making predictions, computing accuracy, precision, recall, F1 score and other performance metrics which is used to assess the quality of the predictions made.
4) Parameter Tuning: After performance evaluation, if the performance metrics are not satisfactory then, then the next step is to tune the parameters of the algorithms such as maximum depth, minimum number of instances per leaf. This can be done using cross-validation techniques to find the optimal values of the parameters.
5) Final Predictions: After parameter tuning, the final step involves applying the same preprocessing steps as before and using the modified optimized decision tree to classify the new instances.

Implementation:

The base decision tree has been implemented with the criterion as entropy and random state as 0 resulting in an accuracy of 93%. After hyper parameter tuning the optimal accuracy is obtained

with criterion as entropy, max depth as 7, minimal sample leaf as 1 and minimal sample split as 10.

Parameters used: GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=42), param_grid={'criterion': ['gini', 'entropy'], 'max_depth': [3, 5, 7, 9], 'min_samples_leaf': [1, 2, 4], 'min_samples_split': [2, 5, 10]}

Random Forest Classifier:

Random Forest Classifier is a supervised machine learning model that combines multiple decision tree to improve the accuracy of the classification model. Random Forest is an advanced version of decision tree, random forest generates a large number of decision trees, each trained on a random subset of the training data and its features. Finally the predictions of all the trees are aggregated using a weighted vote to make the final prediction.

It has several advantages over decision trees such as, reduces overfitting, scalability, Interpretability, improved robustness, etc.

Steps involved in building a Random Forest Classifier:

1) Data Preparation: The initial step is same for almost all the models, data preparation involves cleaning the data, removing irrelevant data, handling missing values, dealing with outliers, feature selection, encoding categorical variables and finally splitting the data into testing, training and validation sets.
2) Building Random Forest/ building the model: To build a random forest model, it is done by creating a set of decision trees using the training data. Each decision tree is built using a random subset of training data. Here, the number of trees in a forest is a hyper parameter which can be tuned using cross validation technique.
3) Evaluation of the Forest: After building the model, it is evaluated using performance metrics on test data. This step involves in making predictions for each instance in the test data. The metrics which are considered for evaluation is accuracy, precision, recall, F1 score.
4) Hyper parameter Tuning: If the performance of the RF did not appear to be satisfactory then we perform hyper parameter tuning such as adjusting the number of trees, adjust tree depth and the optimal values of these can be obtained using cross validation techniques.
5) Final Predictions: Finally, after hyper parameters are tuned the final step is use the update optimized RF classifier on data which classifies into new instances.

<u>Implementation:</u>

The base random forest model has been implemented with the criterion as entropy, n estimators =10 and random state as 0 resulting in an accuracy of 93.9%. After hyper parameter tuning the optimal accuracy is obtained with max depth as 10, criterion as entropy, max features as sqrt and n estimators as 100.

Parameters used: GridSearchCV(cv=5, estimator=RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0), param_grid={'max_depth': [None, 5, 10], 'max_features': ['sqrt', 'log2'], 'n_estimators': [50, 100, 200]}, scoring='accuracy')

K-Nearest Neighbors (KNN) classifier:

KNN classifier is a supervised classifier used for classification and regression tasks. KNN finds the 'k' closet neighbors of a new class in the space, and it assigns the most frequent label of its neighbor.

The major advantage of KNN is its simplicity, because it does not require any complicated model training and it be used on both linear and non-linear boundaries.

Steps involved in building KNN classifier:

1) Data Preparation: The data consists of set of instances, each described by set of features and a target variable indicating the class or label. Furthermore, data cleaning such as outlier treatment, handling missing values and finally splitting the data into test and training sets. The features are normalized to avoid bias or scaling effect.
2) Model Selection: KNN requires selection of a distance metric . The most common distance metrics which are used are Euclidean, Manhattan distances, which have different trade-offs between sensitivity and robustness.
3) Model Training: The KNN model is trained by computing the distances between test instances and training instances. The K nearest neighbors of the test classes are selected based on their distance scores and their labels.
4) Model Evaluation: The performance of KNN classifier can be evaluated using various performance metrics such as accuracy, precision, recall, F1-score, ROC curve. The hyper parameters of KNN include distance metric and number of neighbors 'k', can be tuned using cross-validation techniques.

<u>Implementation:</u>

The base KNN model was run over 11 iterations and the optimal k has been found to be 5. The distance metrics which were uses are – Euclidian and Manhattan. The accuracy has been found to be 90%.

Parameter used: GridSearchCV(cv=5, estimator=KNeighborsClassifier(), param_grid={'n_neighbors': [3, 5, 7, 9, 11]}), estimator: KNeighborsClassifier)

Logistic Regression:

Logistic regression is supervised classification model which can be used for both binary and multiclass classification problems. Logistic regression models the probability of each class label given the input features to a probability between 0 and 1.

The advantage of logistic regression over other classification models is that it provides interpretable coefficients which understands the importance of features and predicts the outcome. Apart from that logistic regression prevents overfitting and improves the performance.

Steps involved in building Logistic Regression:

1) Data Preparation: The initial step is same for almost all the models, data preparation involves cleaning the data, removing irrelevant data, handling missing values, dealing with outliers, feature selection, encoding categorical variables and finally splitting the data into testing, training and validation sets.
2) Building the Model: The model is trained on training data using maximum likelihood estimation approach. During this process, the model learns the optimal values of the coefficients that increase the likelihood of labels given by the input features.
3) Model Evaluation: The performance of the model would be evaluated using performance metrics such as accuracy, precision, recall and F1 score. These metrics would provide an estimate of how well the model is able to predict the correct class label.

Implementation:

The basic logistic regression model was executed, resulting in an accuracy of 88%. After performing hyper parameter tuning an accuracy of 90% has been obtained with penalty as 12, solver as liblinear.

Parameters used: GridSearchCV(cv=5, estimator=LogisticRegression(random_state=42), param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100], 'penalty': ['l1', 'l2'], 'solver': ['liblinear']}

Naïve Bayes Classification

This is a classification algorithm that considers joint probability and labels the classes using bayes theorem.

This is a simple and efficient model which can be implemented on large datasets, irrespective of the datatype, that is, Numerical and Categorical. It works well even with high dimensional data. It can even handle missing values.

Disadvantages: This model does not perform well when the correlation among the features is large.

Steps involved in building a Naïve Bayes Model:

1) Data Preparation: The initial step is same for almost all the models, data preparation involves cleaning the data, removing irrelevant data, handling missing values, dealing with outliers, feature selection, encoding categorical variables and finally splitting the data into testing, training and validation sets.
2) Model training: The model is trained on the preprocessed dataset. During this training process, the model estimates the probability of each input feature given for each label.
3) Model testing: Once the model is trained, it can be used to predict the class labels. The prediction is based on Bayes theorem and the class label with highest probability is then assigned as the predicted class label.
4) Model evaluation: The performance of the model would be evaluated using performance metrics such as accuracy, precision, recall and F1 score. These metrics would provide an estimate of how well the model is able to predict the correct class label.

Implementation:

Upon implementing Naïve Bayes classifier an accuracy of 71% has been found with least precision scores and F1 scores.

Performance Evaluation

The performance metrics which we are going to consider for evaluating the models are:

1) Accuracy: The percentage of correct predictions out of all predictions.
2) Precision: The percentage of true positives out of all positive predictions.
3) Recall: The percentage of true positives out of all actual positive cases.
4) F-1 Score: It is the weight average of both precision and recall.
5) ROC AUC: The area under ROC curve is a plot of true positive rate vs false positive rate.
6) MSE: Mean Squared Error is the average squared difference between predicted and actual data.
7) RMSE: Root Mean Squared Error is square root of MSE, which provides better measure of error than MSE.
8) MAE: Mean Absolute error is the average of absolute difference between actual and predicted data.
9) AE: Absolute error is the absolute difference between actual and predicted data.
10) R2: It measures the proportion of variance.

The models which have been selected for fetal health classification is KNN, Decision Tree Classifier, Random Forest Classifier, Logistic Regression and Naïve Bayes. Based on the

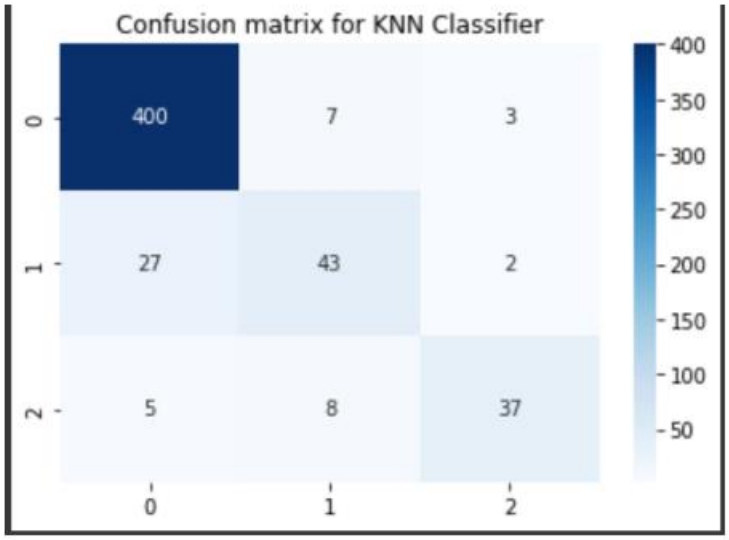
evaluation metrics achieved during the model implementation, it has been found that Random Forest Classifier model appears to be the best and optimal model for this type of classification. It achieved highest accuracy score, precision, recall and F1 score. Apart from that is also observed that Random Forest Classifier had high ROC AUC value and low MSE, RMSE, MAE and AE values.

	Accuracy	Precision	Recall	F1-Score	ROC AUC	MSE	RMSE	MAE	AE	R2
KNN	0.902256	0.849419	0.770944	0.805335	0.938762	0.142857	0.377964	0.112782	60.0	0.648784
Decision Tree	0.930451	0.882793	0.846757	0.863790	0.884954	0.092105	0.303488	0.077068	41.0	0.773558
Random Forest	0.939850	0.910805	0.866346	0.887051	0.959854	0.077068	0.277611	0.065789	35.0	0.810528
Logistic Regression	0.896617	0.830209	0.807186	0.818180	0.952916	0.137218	0.370430	0.114662	61.0	0.662648
Naive Bayes	0.714286	0.650325	0.761644	0.649005	0.883163	0.387218	0.622268	0.319549	170.0	0.048019

It is clearly visible that the classification models exhibit low error scores, which indicates that these models neither underfits nor overfits the data.

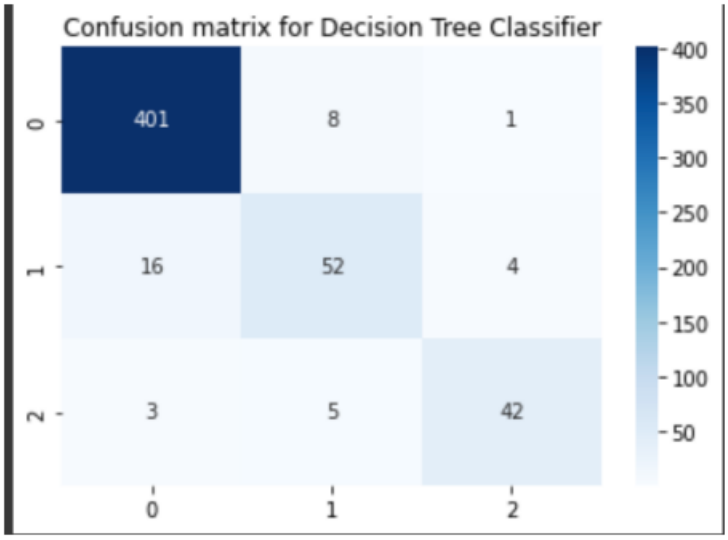
	Training error	Testing error
KNN	0.043287	0.110902
Decision Tree	0.043287	0.110902
Random Forest	0.001255	0.060150
Logistic Regression	0.139272	0.163534
Naive Bayes	0.340652	0.387218

KNN is a non-parametric machine learning algorithm that classifies the instances based on their distances to other instances. KNN model when compared to Random Forest is almost best and demonstrates a strong accuracy with an accuracy score of 90%. However, it had lower precision and recall when compared with others. Which indicates that KNN may have misclassified some instances.



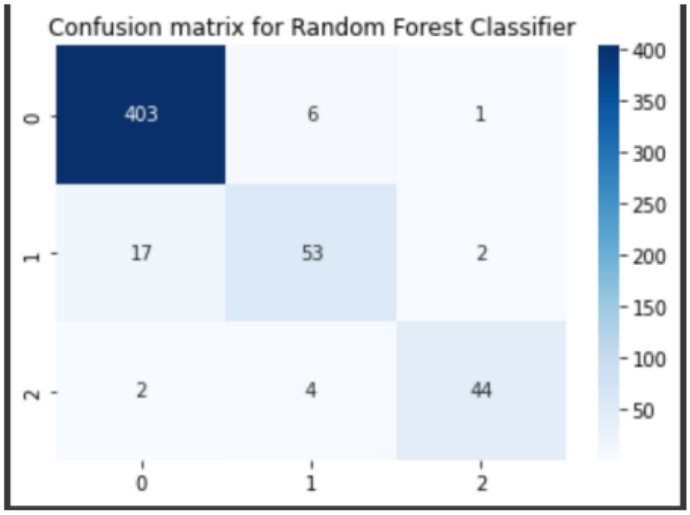
	precision	recall	f1-score	support
1.0	0.93	0.98	0.95	410
2.0	0.74	0.60	0.66	72
3.0	0.88	0.74	0.80	50
accuracy			0.90	532
macro avg	0.85	0.77	0.81	532
weighted avg	0.90	0.90	0.90	532

Decision Tree classifier unlike KNN achieved an accuracy of 93% similar to Random Forest it also had higher recall and precision values. Decision trees model is almost similar to random forest classifier and the evaluation metrics are almost same when compared with each other



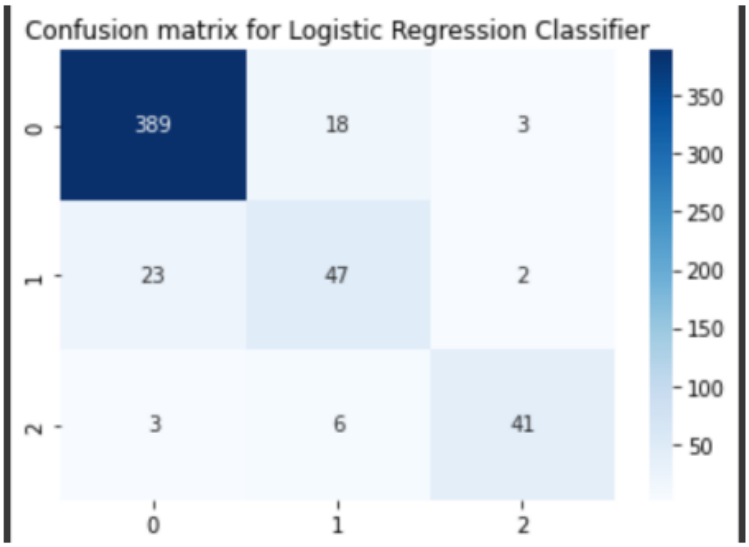
	precision	recall	f1-score	support
1.0	0.95	0.98	0.97	410
2.0	0.80	0.72	0.76	72
3.0	0.89	0.84	0.87	50
accuracy			0.93	532
macro avg	0.88	0.85	0.86	532
weighted avg	0.93	0.93	0.93	532

Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to improve accuracy and prevent over fitting. It has been observed that random forest model has achieved the highest accuracy of 94% and precision of 91% and had high recall values when compared with rest of the models.



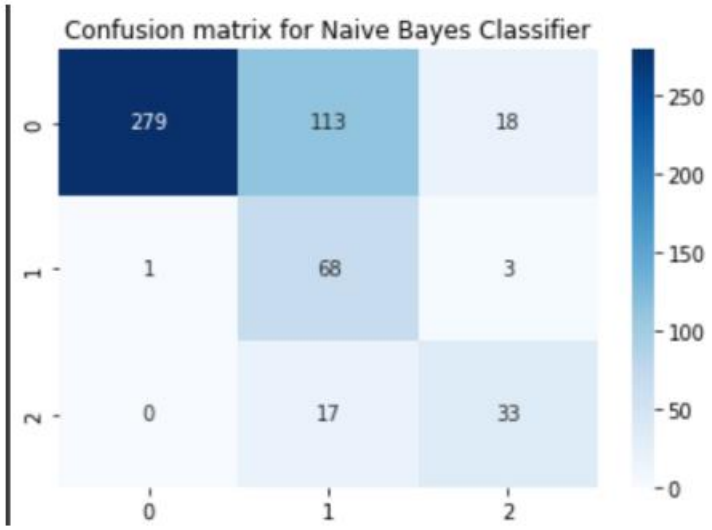
	precision	recall	f1-score	support
1.0	0.95	0.98	0.97	410
2.0	0.84	0.74	0.79	72
3.0	0.94	0.88	0.91	50
accuracy			0.94	532
macro avg	0.91	0.87	0.89	532
weighted avg	0.94	0.94	0.94	532

Logistic Regression is a linear classification algorithm that uses logistic function to model the probability of certain event. Even logistic regression model has achieved an accuracy of 90% and has high ROC AUC value of 0.95 which interprets that it has high true positive rate. The drawback which appeared in this model are it had lower precision and recall values when compared with random forest and decision trees.



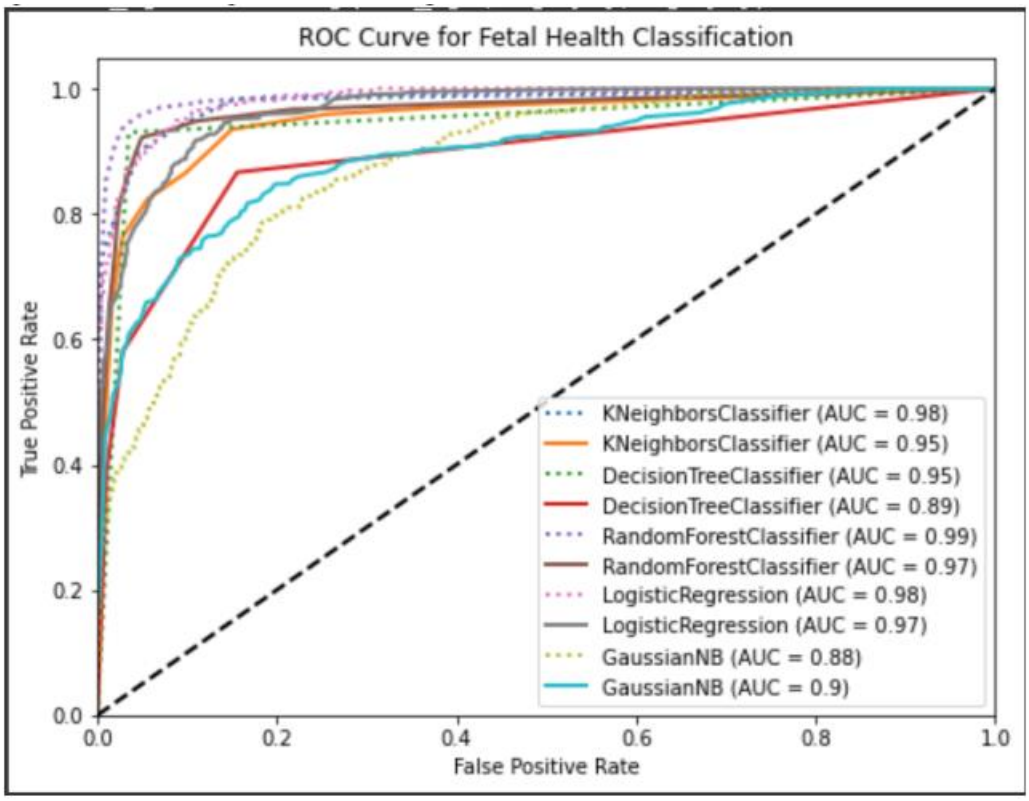
	precision	recall	f1-score	support
1.0	0.94	0.95	0.94	410
2.0	0.66	0.65	0.66	72
3.0	0.89	0.82	0.85	50
accuracy			0.90	532
macro avg	0.83	0.81	0.82	532
weighted avg	0.90	0.90	0.90	532

Naïve bayes is a probabilistic algorithm that uses Bayes's theorem to predict the probability of a class based on the prior knowledge of conditions related to the class. It has been observed that Naïve bayes had the least accuracy of 71% when compared with every model. This suggests that Naïve bayes is not suitable for this type of classification.



	precision	recall	f1-score	support
1.0	1.00	0.68	0.81	410
2.0	0.34	0.94	0.50	72
3.0	0.61	0.66	0.63	50
accuracy			0.71	532
macro avg	0.65	0.76	0.65	532
weighted avg	0.87	0.71	0.75	532

In this classification it has 3 classes hence it comes under multiclass classification. We can observe two AUC curves for each model because of this reason. Here we calculate the AUC and ROC curve for each class separately and then average the results to get overall AUC score. The solid line represents average AUC score. This method gives equal weight to each class. The dotted line represents micro average AUC score. This gives more weight to larger class and less weight to smaller class. It is calculated by total of TP, FP, FN across all classes. In summary solid line represents the performance of model across all classes and dotted line represents the overall performance taking class imbalance into account.



Overall, based on the provided metrics, it seems that Random Forest model is best optimal model for this classification dataset. It achieved the highest accuracy scores compared to rest of the models. It even had high recall and F1 scores as well

Challenges Faced:

During the analysis of the fetal health classification dataset, several challenges were encountered. Some of them are:

- Imbalanced dataset: The fetal health classification dataset had an imbalanced distribution of the target variable, which made it difficult to train the models to predict the less frequent class accurately.
- High dimensionality: The dataset had a large number of features, which could lead to overfitting of the models. This required the use of feature selection techniques such as PCA to reduce the dimensionality of the dataset.
- Multiclass classification: The fetal health classification problem had three classes, which made it a multiclass classification problem. This required the use of appropriate evaluation metrics such as precision, recall, F1-score, and ROC AUC score, which are different from binary classification.

- Model selection: Several machine learning models were evaluated for their performance on the dataset. However, choosing the best model was a challenge, as each model had its strengths and weaknesses. Hyperparameter tuning was performed to optimize the performance of each model

Conclusion:

In this analysis, we used the Fetal Health Classification dataset to build and evaluate several machine learning models, including K-Nearest Neighbors, Naive Bayes, Random Forest, Decision Tree, and Logistic Regression. We started by exploring the data and performing preprocessing steps, such as checking for missing values, encoding categorical variables, and scaling the data.

We then trained and evaluated each model using various evaluation metrics such as accuracy, precision, recall, F1-score, ROC AUC, MSE, RMSE, MAE, AE, and R2. Additionally, we visualized the performance of the models using ROC curves, confusion matrices, summary tables, and lift charts.

Furthermore, we performed parameter tuning for each model to optimize its performance. We also conducted a chi-squared test for correlation analysis and explored the use of PCA to reduce the dimensions of the dataset.

Based on our analysis, we found that the Random Forest model had the highest accuracy among all models. Decision Tree and KNN models also performed reasonably well. Naive Bayes and Logistic Regression models had relatively lower accuracy scores.

In conclusion, we can say that the Random Forest and Decision Tree models are good options for predicting fetal
health classification, given their high accuracy scores. However, the decision of choosing the best model ultimately depends on the specific requirements and constraints of the problem at hand.

Impact:

Based on the results of the analysis, some potential impacts could include:

- Improved fetal health outcomes: By accurately predicting the fetal health status using machine learning models, healthcare providers can take appropriate interventions and treatments to improve the health outcomes of the fetus and ultimately, the mother.

- Cost savings: Early detection of fetal health issues using machine learning models can lead to timely interventions, potentially avoiding costly medical procedures and hospital stays.
- Enhanced patient care: Machine learning models can assist healthcare providers in making informed decisions about fetal health and improving the quality of patient care.
- Advancement in medical research: The analysis of the fetal health classification dataset using machine learning models can contribute to the advancement of medical research in the field of obstetrics and gynecology, potentially leading to new discoveries and innovations in fetal health monitoring and care.