Fetal Health Prediction

Fatma Gizem Isir, Emiliana Geronimo, Sreya Mukherjee, Ege Atay

NJIT SPRING 2024 - DS 675 Final Project

Introduction

For this milestone as a group we decided to explore predicting the health of a fetus. Currently in the world, maternity wards are considered one of the most dangerous places for women. There is a lack of proper prediction in figuring out whether the mother or baby are healthy during this period of time. Particularly for our project, we focused on the health of the baby. It is imperative to detect fetal decompensation early in order to allow for a timely and proper intervention that would potentially help save the fetus. The dataset we picked contains information from cardiotocogram (CTG), which helps to monitor fetal health. Interpretation of CTG values is important for both the fetus and the mother's health. The target variable we have chosen for this dataset categorizes the health status of the fetus into 3 different categories (Normal, Suspect, Pathological) with the data obtained as a result of different measurements. There are 21 different measurements in the dataset with a total of 2126 measurement results for each. In other words, it consists of 22 columns and 2126 rows in total. As a result, the goal of the project is to optimize different machine learning algorithms to create a model that successfully classifies the health status of fetuses by using accuracy and F1-scores as modes of evaluation.

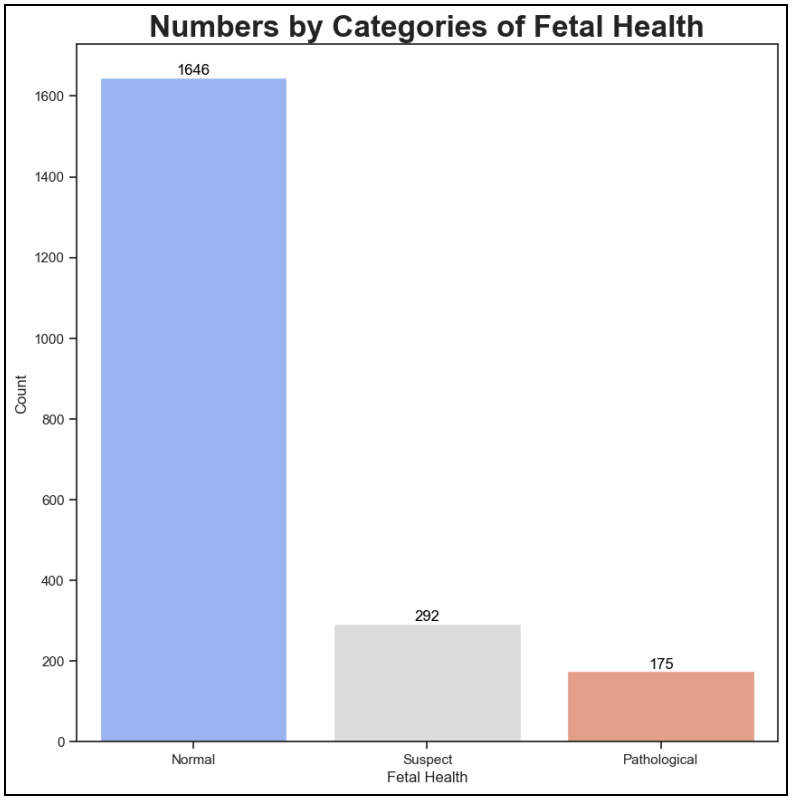
Data Preprocessing

We made changes to the dataset by deleting duplicates and standardizing the data. Deleting duplicates in datasets helps to prevent overfitting, improves generalization, and reduces computational resources by ensuring that each data point is unique and representative of the underlying distribution. There were a total of 13 duplicates. At the same time, standardization of

the training data ensures that features have a mean of zero and a standard deviation of one, enabling models to converge faster during training and making comparisons between features more meaningful. It also prevents features with larger scales from dominating the learning process, resulting in improved model performance and interpretability. Lastly, our data did not have any NULL values to deal with.

Exploratory Data Analysis

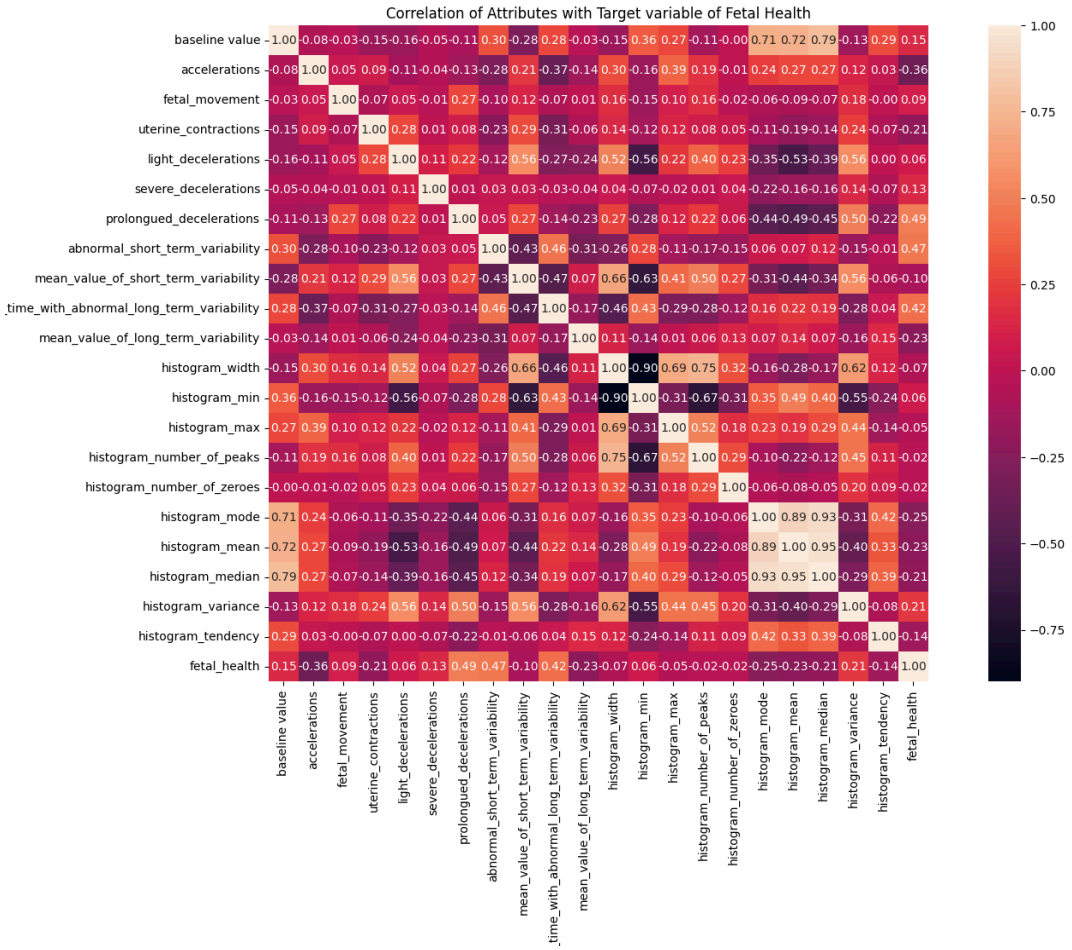
The exploratory data analysis (EDA) section serves as a foundational exploration of the fetal health dataset, providing an initial understanding of its structure, distributions, and key characteristics. We first looked at the distribution of the variables, initially with the target variable. The distribution of the fetal health graph gave us a crucial insight to the classification problem



The vast majority of the data points in our dataset are of normal health, as the pathological or suspect data points are few. This means that it is best practice to optimize our models not only using the accuracy rate but also referring to the misclassification types.

We then plotted and interpreted the distribution of the features. This resulted in mostly normal style distributions with target cases being off-mean.

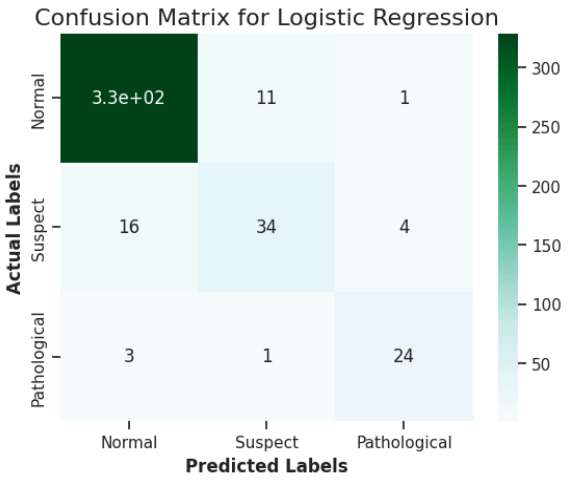
We continued our analysis with a correlation heatmap. This showed no significant correlation except histogram information, such as a negative correlation between histogram width and histogram min ; and very strong correlation between histogram mode, mean, and median values.



Model #1: Logistic Regression

Logistic Regression is a model that can be used for classification problems. For this classification problem we decided to explore the feasibility of of using Logistic Regression Classification.

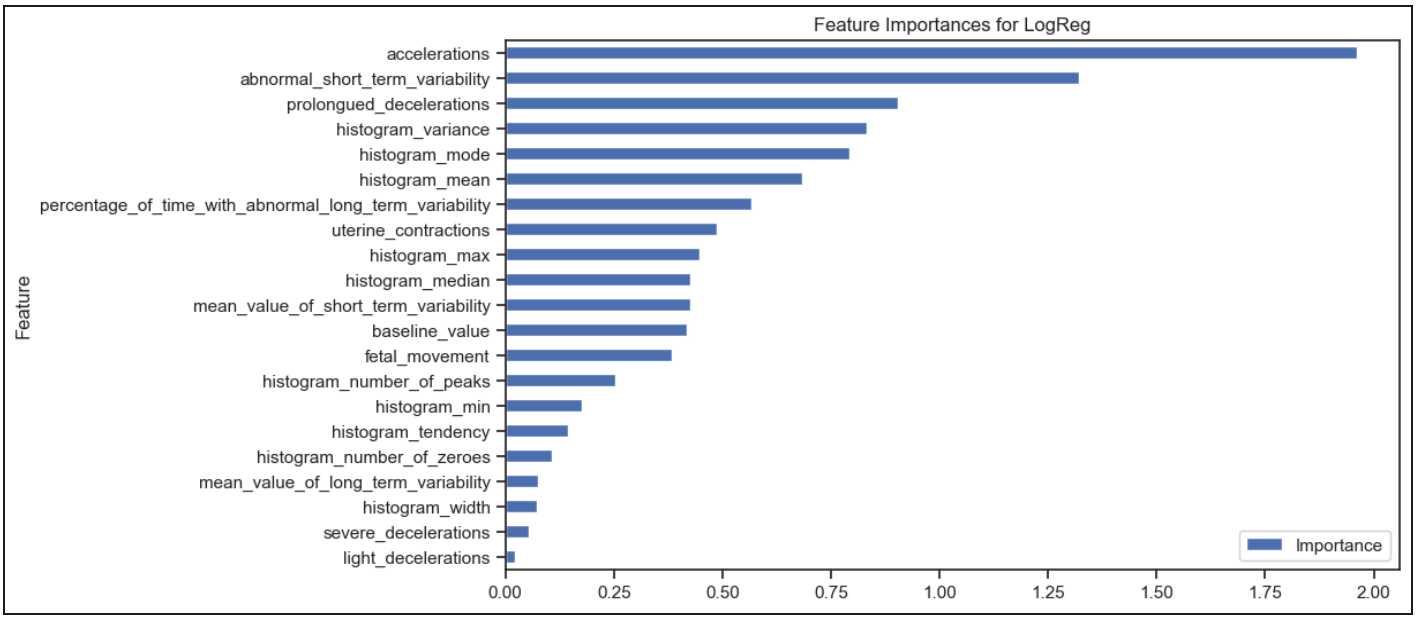
When we used sklearn's default Logistic Regression function plainly, we got an accuracy percentage of 91.48%. The confusion matrix for vanilla LogReg is shown below.



As a standard approach, we decided to use grid search to find optimal parameters for increasing the accuracy of Logistic Regression model. The optimal parameters found by the search were: `{'C': 0.1, 'intercept_scaling': 1, 'penalty': 'l2', 'tol': 0.0001}`. This did not lead to an increase in accuracy percentage nor a better misclassification severity.

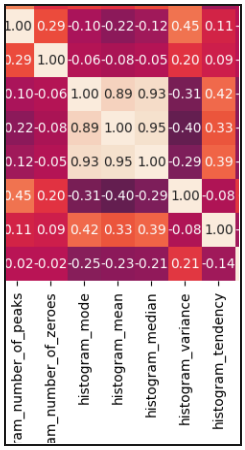
We further explored making changes to the training dataset to improve the accuracy of the model trained on that set. We used SMOTE to oversample the target variable. This did not improve the accuracy percentage, and improved the severity of misclassifications only slightly.

The next process we tried is feature selection. Looking at the importance of each feature on the graph, we decided to drop the five least important features and train the model on the new training set.



This led to a slight increase in the accuracy percentage, and less severe misclassifications. When we decided to experiment further with feature selection.

Although these values do not have low importance, we remember from the exploratory data analysis' correlation heatmap that some values were highly correlated with each other.

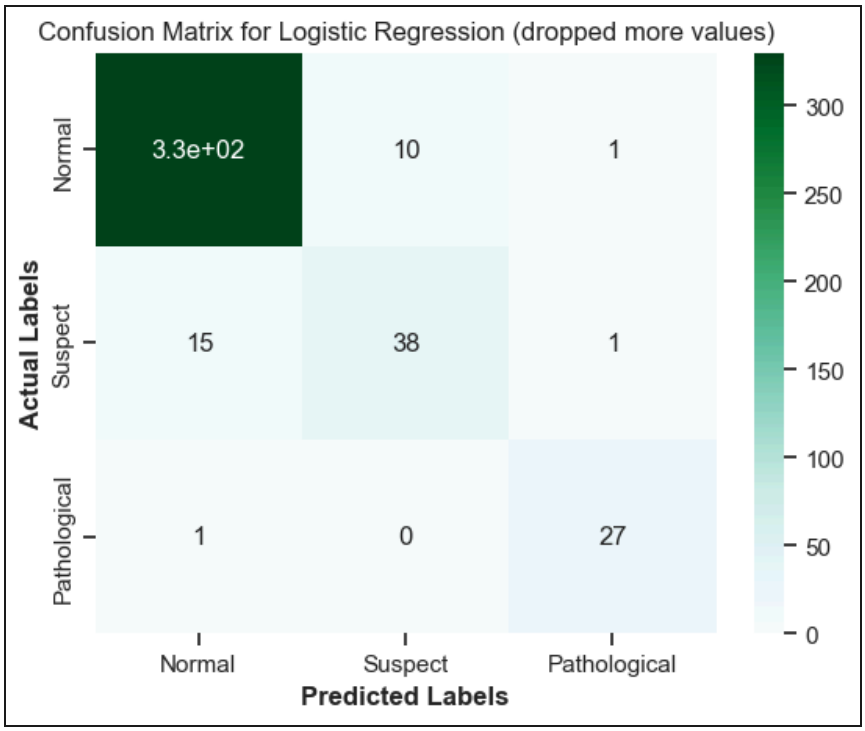


We decided to keep histogram median and drop the two values highly correlated with it: the histogram mode and the histogram mean. After this, the total list columns dropped was

```
['light_decelerations' , 'severe_decelerations', 'histogram_width',
```

`'histogram_mean', 'histogram_mode', 'histogram_min', 'mean_value_of_long_term_variability', 'histogram_number_of_zeroes' ]`.

This led to a significant increase in accuracy percentage, to 93.38% , and a significant improvement in severe misclassifications; there is only one pathological case in our test dataset that is misclassified as normal. The confusion matrix after this transformation is given below.



We were not able to increase the accuracy of Linear Regression models further than this. We believe that the unique value distribution patterns on our dataset makes it hard for a linear regression model to accurately classify the target variable.

Model #2: K-Nearest Neighbors

K-Nearest Neighbors (KNN) is favored for classification tasks due to its simplicity and non-parametric methods, making it easy to understand and adaptable to various datasets without assuming any specific data distribution. We implemented the default call of KNN to the dataset:

```
knn = KNeighborsClassifier(n_neighbors=3, metric= 'euclidean')
```

The classifier was then fitted onto the training data while the predict function was used to create the first model. Accuracy was used as a metric of evaluation as the basis of the project. However, the dataset's target value is imbalanced for the beginning of the KNN experiment, therefore we decided to also report the F1 score. The first model created an accuracy of 90.54% and had an F1 score (represented in the figure below): 0.96, 0.71, 0.87. Generally, an F1 score closer to 1 indicates better performance, while a score closer to 0 suggests poorer performance. Some ways to improve this model is to increase the F1 score to increase the score for the second class or increase the overall accuracy.

Classification Report

	precision	recall	f1-score	support
1.0	0.95	0.97	0.96	341
2.0	0.75	0.67	0.71	54
3.0	0.89	0.86	0.87	28
accuracy			0.92	423
macro avg	0.86	0.83	0.85	423
weighted avg	0.92	0.92	0.92	423

To first improve this model, a grid search was used to find the optimal parameters for the KNN classifier. Grid search is a great method for KNN and other machine learning algorithms because it allows the exploration of multiple combinations of parameters by evaluating each

combination using cross-validation. We looked through a range of possible combinations:

```
param_knn = {
    'n_neighbors': list(range(1, 31)),
    'p': (1, 2),
    'weights': ('uniform', 'distance'),
    'metric': ('minkowski', 'manhattan', 'euclidean')}
```

We found the final best parameters to be:

```
{'metric': 'minkowski', 'n_neighbors': 4, 'p': 1, 'weights': 'distance'}
```

These new parameters were applied to a new KNN classifier and increased the accuracy to 92.20%, which is a 1.66% increase from the previous model. This model with the optimal parameters was used as the base model for further development.

To continue, to enhance the improved model, we used three other approaches: Feature Selection, Solving Class Imbalance, and a Combination of both. The first feature selection method used was the “SelectKBest” function for ranking feature importance, which ranks features based on univariate statistical tests:

```
print(featureScores)
```

	Feature	Score
6	prolongued_decelerations	507.304309
7	abnormal_short_term_variability	337.703020
9	percentage_of_time_with_abnormal_long_term_var...	335.386156
17	histogram_mean	298.759569
16	histogram_mode	276.382795
18	histogram_median	249.699523
1	accelerations	194.618345
19	histogram_variance	150.955827
0	baseline value	137.833999
8	mean_value_of_short_term_variability	118.050463
3	uterine_contractions	93.647474
12	histogram_min	86.468440
10	mean_value_of_long_term_variability	69.418940
4	light_decelerations	66.750344
11	histogram_width	54.215605
20	histogram_tendency	44.854186
5	severe_decelerations	28.438837
14	histogram_number_of_peaks	11.726828
2	fetal_movement	11.700712
13	histogram_max	2.523350
15	histogram_number_of_zeroes	2.134901

We then set a criteria where we kept: a) Feature Scores greater than 100, b) Feature Scores greater than 200. The Features that had a score greater than 100 had the model with the highest accuracy. As well, we tested all other ranked feature combinations by brute force and still found the accuracy of the top 10 features (score >100) being the highest. We also looked at the f1 score. There was improvement in accuracy (94.33%). As well, the second class f1 score increased to 77% and the third class increased to 98% (which is about a 11% increase):

```
Classification Report
              precision    recall  f1-score   support

         1.0       0.96      0.97      0.97       341
         2.0       0.83      0.72      0.77        54
         3.0       0.97      1.00      0.98        28

    accuracy                           0.94       423
   macro avg       0.92      0.90      0.91       423
weighted avg       0.94      0.94      0.94       423
```

Another method was used for feature selection is Permutation Importance. Permutation importance for KNN involves shuffling the values of each feature individually, evaluating the model's performance with the permuted feature, and comparing it to the baseline performance.

Features with larger performance drops after shuffling are considered more important for the KNN model's predictions. These were the results:

```
    print(f"{i}. {feature}: {importance}")

Feature Rankings:
1. percentage_of_time_with_abnormal_long_term_variability: 0.01970055161544526
2. abnormal_short_term_variability: 0.018991331757289 23
3. uterine_contractions: 0.009062253743104 82
4. accelerations: 0.008747044917257696
5. histogram_mode: 0.007328605200945637
6. prolongued_decelerations: 0.004728132387706863
7. histogram_variance: 0.0044917257683215195
8. baseline value: 0.004097714736012614
9. histogram_max: 0.0022852639873916503
10. light_decelerations: 0.0014184397163120588
11. histogram_median: 0.0011032308904649346
12. fetal_movement: 0.0009456264775413725
13. severe_decelerations: 0.0
14. mean_value_of_short_term_variability: -0.00023640661938534313
15. histogram_mean: -0.0009456264775413725
16. mean_value_of_long_term_variability: -0.0011032308904649346
17. histogram_number_of_peaks: -0.0026004728132387744
18. histogram_min: -0.0035460992907801 47
19. histogram_width: -0.0044917257683215195
20. histogram_number_of_zeroes: -0.004728132387706863
21. histogram_tendency: -0.00606776989755714
```

Using permutation importance, models were tested by brute force to find the best accuracy. It was found that top 7, 14, and 16 features shared the same accuracy of 94.8%, however we focused on the top 7 to reduce complexity. Even though the accuracy increased, the F1 scores stayed the same to the base model, which caused some debate. If the F1 score remains unchanged while the accuracy increases after feature selection in an imbalanced dataset, it likely signifies an improvement in the model's ability to correctly classify instances, particularly those belonging to the majority class. This could result from inadvertent emphasis on features more predictive of the majority class or adjustments in the decision threshold, leading to more accurate classification of majority class instances. However, it's crucial to note that the unchanged F1 score suggests that the balance between precision and recall across classes has not shifted significantly.

To tackle the imbalanced dataset problem, we call SMOTE, which uses oversampling to generate synthetic samples for the minority class. We saw a decrease in accuracy (92.2%),

however F1 scores for the minority classes increased greatly (2.0: 0.79 and 3.0: 0.97). This was used as the new base model.

We then decided to evaluate the best models of both feature selection approaches and combine it with the balanced dataset with hopes to increase accuracy with a simpler model and balanced dataset. We used the best SelectKBest and Permutation Importance models with the SMOTE call. Accuracy, in this case, holds stronger weight for evaluation because the data is now balanced. Compared to the new base model, selecting the best features for the permutation importance approach increased accuracy to 93.38%:

```
Classification Report
              precision    recall  f1-score   support

         1.0       0.99      0.93      0.96       341
         2.0       0.70      0.93      0.80        54
         3.0       0.90      1.00      0.95        28

    accuracy                           0.93       423
   macro avg       0.86      0.95      0.90       423
weighted avg       0.95      0.93      0.94       423
```

Using the best features selected for SelectKBest approach with the new base model, there was an increase in both accuracy (94.33%) and F1 score for all categories.

```
  Classification Report
              precision    recall  f1-score   support

         1.0       0.98      0.96      0.97       341
         2.0       0.76      0.81      0.79        54
         3.0       0.93      1.00      0.97        28

    accuracy                           0.94       423
   macro avg       0.89      0.92      0.91       423
weighted avg       0.95      0.94      0.94       423
```

To sum up, for the previously decided evaluation metric of accuracy, the model with solely the feature selection using permutation importance had the highest accuracy and was used to compare with other machine learning models, even though there was an imbalance of the data.

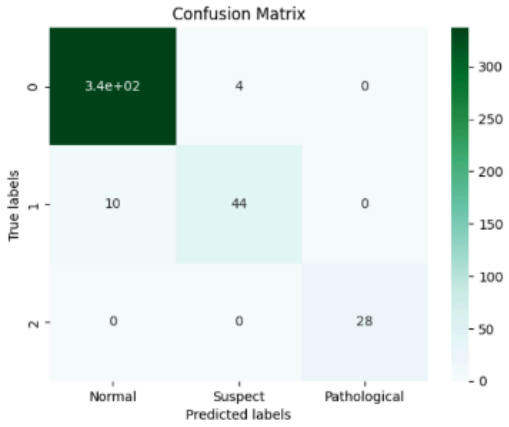
The second highest accuracy was the combination of SelectKBest and SMOTE approach with an accuracy of 94.33% and as well an improvement in F1 scores. Further investigations would include creating more clarity of what is considered a better model: high accuracy model using imbalanced data vs. lower accuracy using balanced data.

Model #3: Random Forest

First, the Random Forest model was implemented with the default hyperparameter. Default hyperparameter (n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1..).

Random Forest Accuracy: 0.966903073286052

	precision	recall	f1-score	support
1.0	0.97	0.99	0.98	341
2.0	0.92	0.81	0.86	54
3.0	1.00	1.00	1.00	28
accuracy			0.97	423
macro avg	0.96	0.93	0.95	423
weighted avg	0.97	0.97	0.97	423



The accuracy score is 96.69%. While the model can completely classify the Pathological class, it cannot completely classify Normal and Suspicious cases. It can be said that from Confusion Matrix, the model misclassified 10 fetuses of 54 Suspect status as Normal and misclassified 4 of 341 Normal status as Suspect.

The next step is to optimize the model with the Random search algorithm. For random grid: n_estimators: Number of trees in random forest, max_depth: Maximum number of levels in

tree min_samples_split: Minimum number of samples required to split a node min_samples_leaf: Minimum number of samples required at each leaf node bootstrap: Method of selecting samples for training each tree, hyperparameters have been tuned.

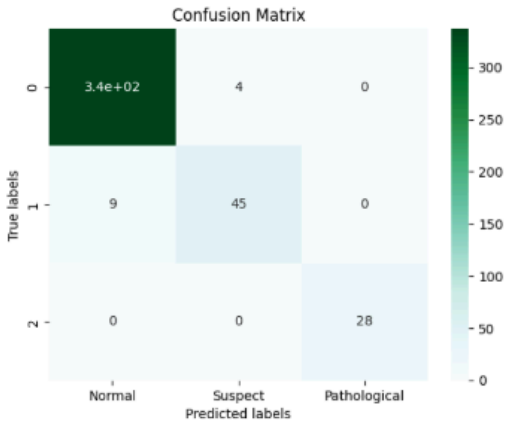
```
rf_random_grid = {'n_estimators': [50, 100, 150, 200, 250, 300, 350, 400, 450, 500],
                  'max_depth': [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, None],
                  'min_samples_split': [2, 5, 10],
                  'min_samples_leaf': [1, 2, 4],
                  'bootstrap': [True, False]}
rf_random_grid
```

As a result, the best parameters were found as follows.

```
rrf.best_params_
```

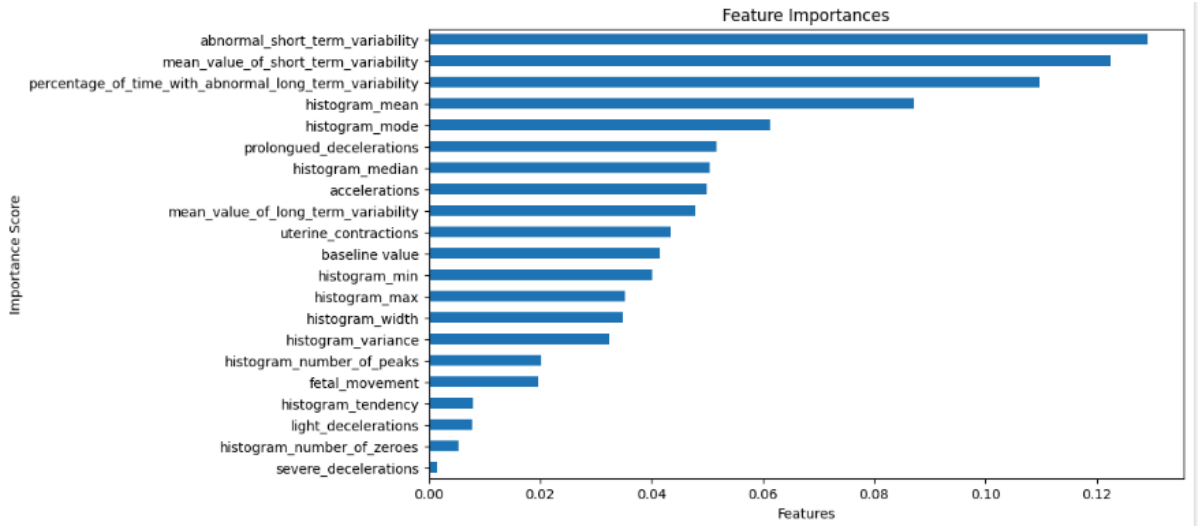
```
{'n_estimators': 450,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_depth': 30,
 'bootstrap': False}
```

```
Random Forest Accuracy: 0.9692671394799054
              precision    recall  f1-score   support

         1.0       0.97      0.99      0.98       341
         2.0       0.92      0.83      0.87        54
         3.0       1.00      1.00      1.00        28

    accuracy                           0.97       423
   macro avg       0.96      0.94      0.95       423
weighted avg       0.97      0.97      0.97       423
```



After Random Search, a 1% improvement in f1-score value was observed for the Suspect class. Accuracy value increased to 96.92%.

To optimize the model's result, feature importances are checked. It sorts according to the features that can provide splitting in a way that decreases impurity in all decision trees.



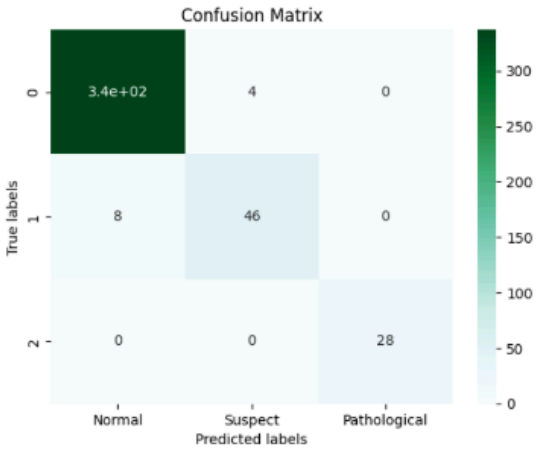
Based on the feature importance sorting, the lowest 4 features were dropped first, but then it was observed that better results were obtained by dropping only the last 3 features. And 'light_decelerations' , 'histogram_number_of_zeroes', 'severe_decelerations' are dropped.

After the 3 least important features were dropped, it was aimed to find the best parameters again with random search.

```
rrf.best_params_
```

```
{'n_estimators': 450,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_depth': 30,
 'bootstrap': False}
```

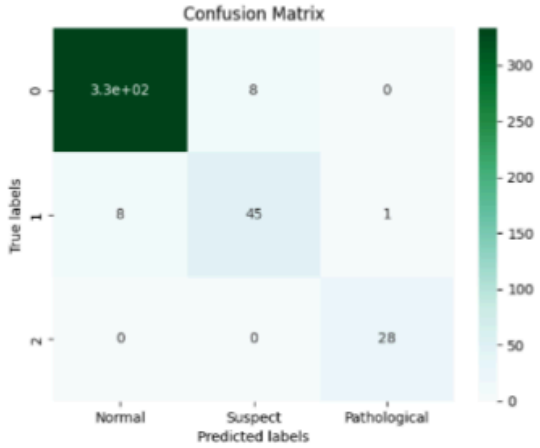
```
Random Forest Accuracy: 0.9716312056737588
              precision    recall  f1-score   support

         1.0       0.98      0.99      0.98       341
         2.0       0.92      0.85      0.88        54
         3.0       1.00      1.00      1.00        28

    accuracy                           0.97       423
   macro avg       0.97      0.95      0.96       423
weighted avg       0.97      0.97      0.97       423
```



After dropped 3 least important features the accuracy value increased to 97.16% and f1 score of suspect class increased by 1% to 88%

Random Forest Accuracy: 0.9621749408983451

	precision	recall	f1-score	support
1.0	0.98	0.98	0.98	341
2.0	0.87	0.83	0.85	54
3.0	0.93	1.00	0.97	28
accuracy			0.96	423
macro avg	0.93	0.94	0.93	423
weighted avg	0.96	0.96	0.96	423

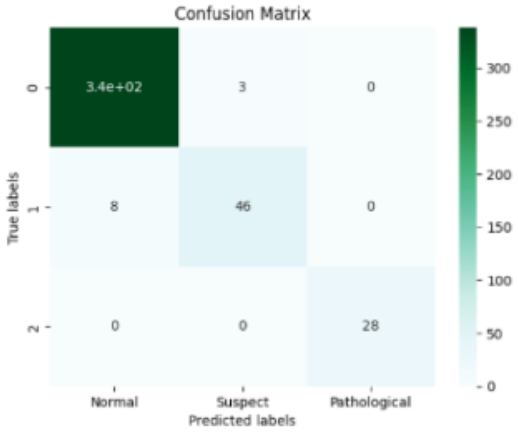


Since the dataset is imbalanced, the model was tried to be optimized by oversampling with the SMOTE method. However, although the imbalance problem was solved, a decrease was observed on f1-score for all classes and on the accuracy. After hyperparameter tuning was performed for the model, results could not be optimized.(best parameters: {'n_estimators': 450, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 30, 'bootstrap': False})

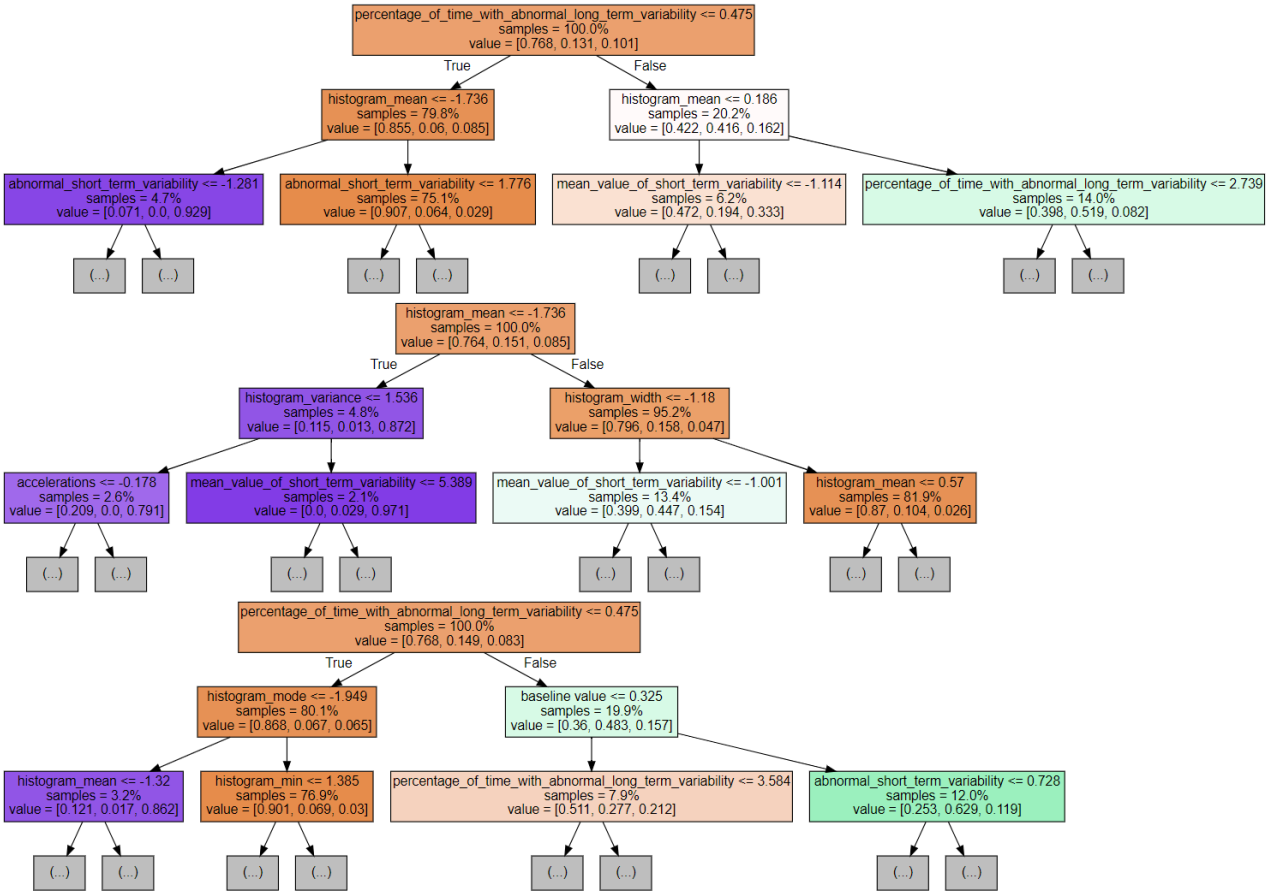
Except from the 3 dropped features, the model was tried to be optimized by dropping 'fetal movement', 'histogram number_of_peaks', 'histogram max' features according to the global feature importance and the model was trained using feature 15. Best parameter after random search: {'n_estimators': 250, 'min_samples_split': 2,'min_samples_leaf': 2, 'max_depth': 10, 'bootstrap': False}

Random Forest Accuracy: 0.9739952718676123

	precision	recall	f1-score	support
1.0	0.98	0.99	0.98	341
2.0	0.94	0.85	0.89	54
3.0	1.00	1.00	1.00	28
accuracy			0.97	423
macro avg	0.97	0.95	0.96	423
weighted avg	0.97	0.97	0.97	423



And as a result, the best results were obtained for the Random Forest model, and the model accuracy increased from 96.69% to 97.40%. The f1 score for Suspect increased to 89%. While a total of 14 data were misclassified in the base model, this number was reduced to 11 in the final optimized model. However, the model still classifies 8 out of 54 data as Normal in the classification of Suspect status.



Model #4: Support Vector Machine

The final model our group worked on was using Support Vector Machines to accurately predict the fetal health of the fetus. The first step in this model was to preprocess the data to make sure it is scaled and balanced prior to selecting the feature.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import accuracy_score

# Initialize linear SVM classifier
svm_linear = SVC(kernel='linear', random_state=150)

# Initialize SelectFromModel with linear SVM
selector = SelectFromModel(estimator=svm_linear)

# Fit SelectFromModel to the training data
selector = selector.fit(X_train, y_train)

# Get the selected features
X_train_selected = selector.transform(X_train)
X_test_selected = selector.transform(X_test)

# Initialize SVM classifier with rbf kernel
svm_classifier = SVC(kernel='rbf', gamma='scale', random_state=42)

# Train SVM classifier on selected features
svm_classifier.fit(X_train_selected, y_train)

# Predict on the testing set
y_pred = svm_classifier.predict(X_test_selected)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy with SVM and feature selection:", accuracy)
```

Then using this code shown above, it selected features based on three classes 1, 2 and 3 which identifies how healthy the fetus is. Below you can see by just running a basic model with SVC kernel classifier and feature selection the highest F1 Score achieved was 0.96 while the average accuracy was 0.94.

```
Classification Report:
              precision    recall  f1-score   support

           1       0.94      0.98      0.96       341
           2       0.84      0.67      0.74        54
           3       1.00      0.89      0.94        28

    accuracy                           0.94       423
   macro avg       0.93      0.85      0.88       423
weighted avg       0.93      0.94      0.93       423
```

Seeing these results, we realized we need to try some other methods in order to improve the F1 Score and accuracy. In order to do this we did two things: Random Search and SMOTE Analysis. Random Search is a hyperparameter technique used specifically in machine learning that is used to help locate the best set of hyperparameters. It is different from grid search as it does not

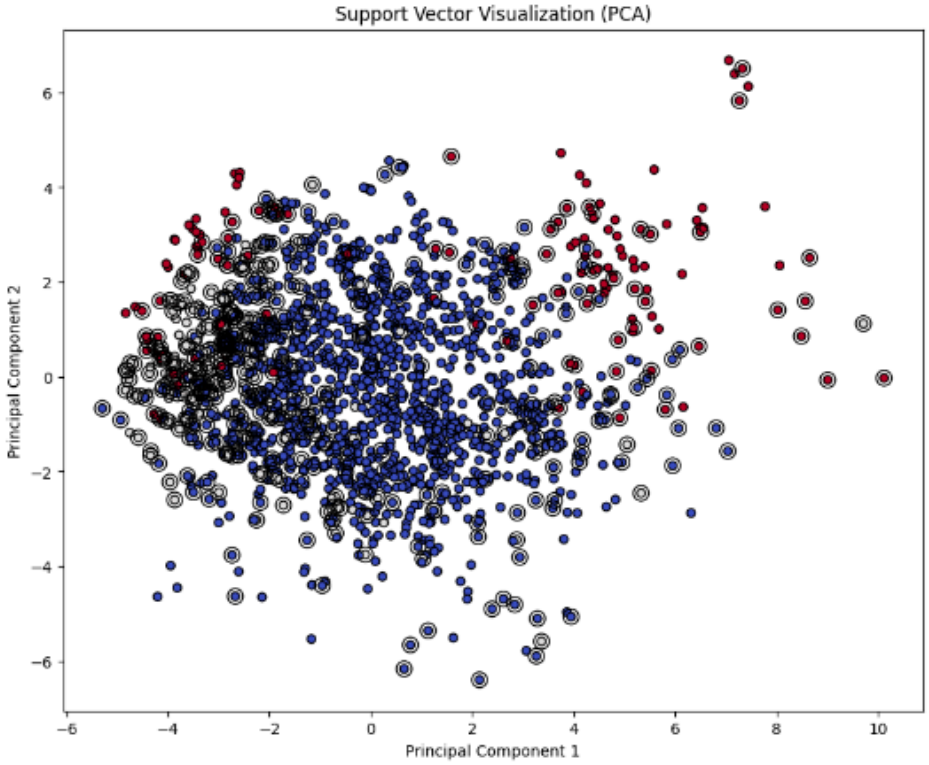
systematically evaluate every possible combination but rather randomly samples hyperparameter combos from a range that is predefined. Random search is a great hyperparameter technique as it explores a broad range of hyperparameters randomly, efficiently searching the space. After using both SMOTE and Random search, both the accuracy, precision and F1 score improved. The results are shown below.

```
Best hyperparameters: {'kernel': 'rbf', 'gamma': 'scale', 'C': 10}
Classification Report:
              precision    recall  f1-score   support

           1       0.97      0.99      0.98       341
           2       0.90      0.80      0.84        54
           3       1.00      1.00      1.00        28

    accuracy                           0.96       423
   macro avg       0.95      0.93      0.94       423
weighted avg       0.96      0.96      0.96       423
```

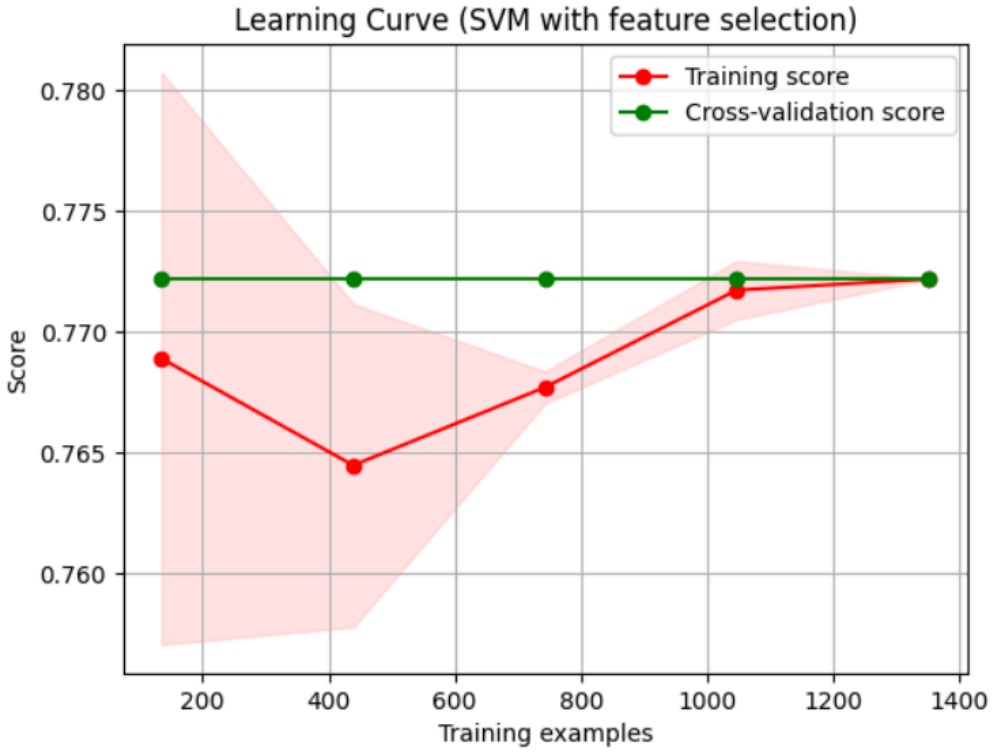
Additionally I used hinge loss to show where my model could be improved.

$$\text{Hinge loss}(y, f(x)) = \max(0, 1 - y \cdot f(x))$$

This is the loss function used for SVM and for us it 32% which is used to measure the error between the predicted output and the true labels.



PCA is a dimensionality reduction technique commonly used in conjunction with Support Vector Machines to visualize high-dimensional data in a lower-dimensional space.We used this visualization in order to reduce the dimensionality of the feature space to two or three dimensions, enabling the visualization of the decision boundaries and the distribution of data points. This is particularly useful when dealing with datasets with a large number of features, where direct visualization becomes challenging which was the case for us as we had 21 features.



At the end of training the SVM model, we coded a learning curve graph in order to see what the performance of the graph was in terms of classification accuracy and mean squared error. This learning curve graph helped us understand how the performance of the model evolves as more data is used for training. By plotting the training and validation performance metrics over time or training examples, we can see that after training our model was overfitting. It also helped us see the bias and variance trade off which is crucial to find the right balance. Overall, learning curve graphs provided us valuable insights into the behavior of the SVM model we created during training and if we wanted to in the future make other hypertuning choices, this model can help us make the informed decisions to do so.

Conclusion

Overall, the purpose of this experiment is to test different machine learning algorithms strong for classification on our fetal health dataset, use various methods to increase our chosen evaluation metric of accuracy, and access which model created the highest accuracy. The model that has shown the highest accuracy was the Random Forest algorithm with an accuracy of 97.4%. As well, all algorithms had the ability to improve their models with different methods, such as, SMOTE, feature selection, grid and random search for hyperparameter tuning. The algorithms ranked best to worst were Random Forest, Support Vector Machine, k-Nearest Neighbors, and Logistic Regression, with Random Forest being the top-performing model. Random Forest being the best suggests that the data likely has complex, non-linear relationships that Random Forest, being an ensemble method, is adept at capturing. Support Vector Machine performs well, indicating the presence of clear class boundaries or separability in the data. K-Nearest Neighbors performs decently, implying that local patterns or similarities are influential. Logistic Regression, ranking last, suggests that the data might not be well-suited for linear modeling or that the relationships between features and the target variable are not strongly captured by linear assumptions. Through this experiment, we also examined the quality of models when working with imbalanced data. Specifically, we questioned whether a model with higher accuracy on imbalanced data is preferable to a model with lower accuracy on balanced data. Opting for a model with higher accuracy on imbalanced data may be preferable when correctly identifying instances of the majority class is paramount, and misclassifying minority class instances carries lower consequences. Conversely, prioritizing a model with lower accuracy on balanced data becomes necessary when achieving equitable performance across all classes is

crucial, especially when misclassifying minority class instances has significant ramifications. Balancing these considerations involves assessing the relative costs of false positives and false negatives, as well as the broader context of the application domain, to ensure that the chosen approach aligns with chosen goals and constraints. Further investigation would entail if more complex machine learning algorithms would perform better on the fetal health dataset.

Link to Milestone 2:

https://drive.google.com/file/d/1bO7aJ3-psSb9oYP_JyHHN1mNdsJnRfcD/view?usp=drivesdk

Link to Code:

https://colab.research.google.com/drive/1t2ocF_7gYo2sys9Le26UpbK9boqpwBJy?usp=sharing

Youtube Link of Presentation: https://youtu.be/ZfCxOSIP4Ic

Works Cited:

- https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification/data
- https://www.kaggle.com/code/saumyanishi/tutorial-unveiling-knn-wizardry