UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL

Preterm Labor Prediction using Uterine Electromyography with Machine Learning and Deep Learning Models

Inês Maria de Freitas Martins

Mestrado em Bioestatística

Trabalho de projeto orientado por:
Prof. Maria Helena Mouriño Silva Nunes
Prof. Arnaldo Manuel Guimarães Batista

2023

Acknowledgments

Quero agradecer ao Professor Arnaldo e à Professora Helena pelo apoio e pela confiança, sem eles esta dissertação não tinha sido possível.

Adicionalmente, quero mencionar o meu agradecimento a todas as pessoas envolvidas no Uterine Explorer Project da Universidade Nova de Lisboa, por toda disponibilidade, conselhos e ajuda mostrada.

Com risco de um dia alguém ler isto e ficar com vergonha alheia, deixo aqui os meus agradecimentos da forma mais simples e livre de constrangimentos que arranjei. Adicionalmente, escrever estes agradecimentos com um bom português, revelou-se quase mais complicado do que escrever esta tese. Então aqui vai:

Obrigada à minha mãe e ao meu pai que sempre me apoiaram em todas as minhas decisões, me inspiraram noutras e me incentivaram a ser mais ambiciosa e aventureira todos os dias. À minha mãe que se orgulha demasiado comigo, desde o primeiro dia em que partilhou os meus exames com as amigas por email até ao dia que se emocionou a ouvir-me a tocar a música dos Parabéns a Você na flauta. Ao meu pai que sempre me ensinou as prioridades e me deu demasiada confiança, quando me encorajou a faltar ao meu primeiro exame de matemática porque estava nervosa ou quando me disse que devia mudar de curso para ser atriz.

Obrigada ao Lucas, por todas as boas memórias, amizade e orgulho!

Obrigada aos meus avós e padrinhos!

Obrigada aos amigos, à Sandra, Carminho, Xana, Catarina, Martine, Eduardo, Hugo, Rui, Sérgio, Zé, Tomás, Clarinha e João, por tudo!

Obrigada aos amigos que fiz por aqui e aos outros que cá já estavam. Um grande obrigada por tudo que veio e ainda está para vir. Menção Honrosa à Catarina, que apesar de usar Quechua, revelou que isso nem sempre é mau e à Bia, que foi o meu primeiro amor à primeira vista! E ao próximo jantar no Guaca!

Obrigada ao meu Du. por todo o apoio e amor, e capacidades de carpintaria que se mostraram muito relevantes!

Por último, fica aqui registado, com esperança de que nenhuma das pessoas aqui referidas leiam os agradecimentos, que a melhor forma de agradecer é com uma celebração final. Mal posso esperar por nos juntarmos todos para celebrar (prometo que levo uma garrafinha).

Abstract

The World Health Organization defines premature birth as the birth of a baby before the completion of 37 weeks of gestation which is considered a high health risk for both the baby and the mother. Prematurity is the leading cause of death in the world for children under 5 years old, therefore monitoring the uterus to predict preterm labor has become essential. Currently, the Intrauterine Pressure Catheter and the External Tocography are the most used monitoring devices, however, they are invasive and don't perform well with high body mass index (BMI) patients, respectively. The Electrohysterogram (EHG) has emerged as a non-invasive method for predicting premature birth with high performance for mothers with high BMI. This method uses electrodes placed on the abdomen to record uterine contractions by producing an electrical signal, that contains important information regarding the electrical activity of the uterus. The study of the EHG signal is one of the most used practices for studying and classifying premature birth using Machine Learning (ML) and Deep Learning (DL) techniques. In this technique, features are extracted from the signal such as frequency, amplitude, and others to represent the signal and inserted into algorithms capable of making predictions based on the signal characteristics. However, this classification method is still in the experimental phase, and there is a gap in the clinical context for automatic birth type prediction. One of the challenges faced by this method is the lack of observations of premature births in the databases used. Oversampling techniques, such as SMOTE, address the lack of observations of premature births in the databases by producing synthetic observations for the minority class.

In this thesis, the Welch estimation of the power spectra of the signal of each contraction from the TPEHG Ljubljana public database is used as features, comprising 200 features. The Minimum Redundancy Maximum Relevance (MRMR) Algorithm was used to search for the most relevant features from this dataset with only 180 showing any relevance, and SMOTE was applied to solve the skewed dataset problem. Four different machine learning algorithms were used, including the Support Vector Machine, the RUSBoosted trees, a Shallow Neural Network, and a Random Forest classifier, moreover, a deep learning network was also tested. These were also optimized with the Bayesian hyperparameter optimization. All algorithms performed with high accuracy, although showing a low predictive power for the test group, probably due to a highly imbalanced test set.

We concluded that the use of spectral features of the contractions as an alternative to the time-frequency features shows promising results with the training dataset, but cannot accurately predict preterm labor in the test set, due to the imbalanced dataset problem. More samples should be collected in the future so more meaningful conclusions can be taken.

Keywords: SMOTE, Machine Learning, Deep Learning, Electrohysterogram, Preterm Birth

Resumo

De acordo com a Organização Mundial da Saúde (OMS) o parto prematuro é definido como o nascimento de bebés antes da finalização das 37 semanas de gestação, sendo considerado um risco de saúde elevado tanto para o bebé como para a mãe. Dois terços destes partos, não tem um diagnóstico específico, enquanto os restantes encontram-se normalmente associados a fatores relacionados com a mãe como várias gravidezes, historial de partos prematuros, uso de drogas, idade inferior a 18 anos, entre outros. A prematuridade é a primeira causa de morte no mundo para crianças com menos de 5 anos, uma vez que quando ocorre o parto, os bebés não se encontram completamente desenvolvidos, podendo vir a sofrer deficiências a nível visual e auditivo e também outras complicações ao nível da saúde como problemas cardiovasculares ou respiratórios. Em Portugal, de acordo com a Sociedade Portuguesa de Pediatria, 8% dos bebés nascem prematuros. Deste modo, a monitorização dos partos de forma a prever partos pré-termo tornou-se fundamental.

Os dois métodos mais comumente usados na monitorização da contratilidade uterina são o Cateter de Pressão Intrauterino e o Tocograma Externo, porém ambos apresentam limitações como o facto de ser invasivo ou de não mostrar eficácia para grávidas de elevada massa corporal, respetivamente. O estudo da atividade das contrações no útero através do Electrohisterograma (EHG) como método alternativo tem sido uma forte aposta na previsão do parto prematuro. O EHG é um método não invasivo realizado através de elétrodos colocados no abdómen, que regista a atividade contrátil do útero e resulta num sinal elétrico. Demonstra eficácia em pacientes com índice de massa corporal alta, sendo capaz de indicar quando as grávidas vão entrar em trabalho de parto.

Atualmente, o estudo do sinal EHG é uma das práticas mais usadas para estudar e classificar o parto prematuro através de técnicas de *Machine Learning (ML)* e *Deep Learning (DL)*. Para isso, utilizam-se características *frequenciais*, temporais, entre outras provenientes do sinal, chamadas de *features*, que vão representar o sinal. Estas são depois inseridas em algoritmos de *ML* e *DL* capazes de fazer previsões com base nas características do sinal. Em literatura as *features* mais utilizadas para representar os sinais EHG consistem na frequência, amplitude, entropia e outras, demonstrando resultados positivos com elevado valor preditivo, tanto em algoritmos de *Machine Learning* como de *Deep Learning*. Desta forma, através do sinal EHG obtido na monitorização do útero será possível prever se a grávida irá ter um parto prematuro ou termo. No entanto, esta classificação ainda se encontra numa fase experimental, existindo uma lacuna no contexto clínico, para uma previsão automática do tipo de parto.

Todos estes trabalhos enfrentam um problema associado à falta de observações de partos prematuros nas bases de dados utilizadas. As soluções propostas para combater o desequilíbrio nos dados envolve a utilização de técnicas de sobreamostragrem, como SMOTE, que consistem na produção de observações sintéticos para a classe da minoria (partos prematuros). O número ideal de amostras a serem produzidas é ainda algo a ser estudado, sendo que a maior parte dos estudos fazem uma compensação dos dados com uma proporção final de observações de 1:1, porém este método pode levar a um decréscimo na habilidade do classificador identificar a classe maioritária e uma previsão irrealista e demasiado otimista. De acordo com os autores, o SMOTE atinge os melhores resultados através da combinação de uma subamostragem da classe maioritária com a sobreamostragem da classe minoritária, através do SMOTE.

Num sinal EHG processado é possível distinguir a existência de contrações como Braxton-Hicks, ondas Alvarez e ondas LDBF (*Longue Durée Basse Fréquence).* De momento, na literatura as features são extraídas do sinal completo e não das contrações, nomeadamente das Alvarez e Braxton-Hicks, que contêm informação relevante para a prematuridade do parto. Contudo, as contrações são séries temporais com um número diferente de observações. Deste modo, a solução apresentada para este problema é a análise

espectral de cada contração, através do espetro de cada contração, obtido através de uma transformação de tempo para frequência, como a Transformada de Fourier, que é capaz de representar um sinal na base de dados. Esta técnica é usada para extração de *features* e classificação no campo de diagnóstico médico. Dentro da estimação espetral existem dois métodos: paramétricos e não paramétricos, sendo que o método Welch é uma abordagem não paramétrica, capaz de calcular o espetro de cada contração detetada no sinal EHG, que demonstrou bons resultados na classificação das contrações noutros trabalhos, representando bem o singal EHG, e apresentando sempre a mesma dimensão, independente da duração da contração.

Neste estudo, foi utilizada a base de dados pública TPEHG (Term Preterm EHG) com um total de 300 registos, 262 pré-termo e 38 termo. A base de dados apresenta 4 elétrodos, com 3 canais bipolares, sendo que apenas um canal foi escolhido, de acordo com a literatura, visto que o sinal vertical tem uma maior variação do potencial de sinal. Este sinal foi depois filtrado para eliminar o ruído materno do ECG, ou outros ruídos relacionados, e processado para uma frequência amostral final de 4 Hz. As *features* foram extraídas através da estimação espetral pelo método Welch, finalizando com um total de 200 *features*. No final, o base de dados utilizado consistia em 4622 observações/contrações, 407 correspondentes a parto prematuro e 2829 parto termo, com 200 *features* cada. Esta base de dados foi depois fornecida a três algoritmos diferentes de ML, incluindo o Random Forest, RUSBoosted Trees, Support Vector Machine, e uma Shallow Neural Network, e o algoritmo Long-Short Term Memory de DL, com o objetivo de classificar os parto prematuros. Até agora, nenhum estudo se focou na utilização de um algoritmo de LSTM, e na utilização do espetro das contrações como *features*.

Neste estudo, as técnicas mencionadas anteriormente foram aplicadas em 5 cenários diferentes nos algoritmos de ML, de modo a obter o modelo mais robusto para evitar situações de overfitting, e obter os resultados mais realistas possíveis, (1) treinar os dados, sem qualquer opção adicional de outros métodos; (2) treinar os dados com os mesmos algoritmos, adicionando uma técnica de sobreamostragem sintética, *SMOTE*; (3) treinar os dados com técnica de SMOTE mais uma técnica de redução de dimensionalidade, PCA; (4) treinar os dados com a utilização de um método de seleção de *features*, MRMR; (5) *tuning* dos parâmetros do modelo, através do método *Bayesian Optimization*. Desta forma, os dados foram treinados, validados, e os modelos com melhores resultados preditivos foram depois testados. Os algoritmos de DL foram apenas testados usando o dataset original e o dataset com SMOTE aplicado. Para todos os algoritmos, a *accuracy, precision, recall, F1-Score, false negative rate, false positive rate e AUC* (exceto para os de DL) foram calculados.

Os resultados indicam que usar os primeiros 200 pontos da estimação espetral pelo método Welch, como *features* frequenciais, não proporciona melhores resultados quando comparando a *features* mais tradicionais, de tempo-frequência, usadas em toda a literatura. Além disso, utilizar a técnica de SMOTE conciliada com uma subamostragem da classe maioritária produz piores resultados quando comparando com a aplicação de só SMOTE, como usado pela maioria dos autores. Os algoritmos de ML têm um melhor comportamento que os de DL, uma vez que são modelos mais simples não dependentes de uma elevada quantidade de dados. Apesar dos resultados promissores no grupo de treino, com uma elevada *Accuracy, F1-score e AUC*, o momento de teste teve uma performance abaixo dos valores esperados e em literatura. Com base nestes resultados, concluímos que apesar da abordagem da aplicação de SMOTE após a separação em grupo de treino de teste ser a mais correta, não permite resultados semelhantes à literatura (em que esta ordem de passos usada é a inversa), uma vez que o algoritmo é processado usando um grupo de teste com uma estrutura muito diferente à de treino, o que pode levar a menor *precision* e *recall*.

Em suma, conclui-se que a utilização do espetro das contrações como *features* frequenciais num dataset sobreamostrado com a técnica de SMOTE, utilizando as diferentes técnicas de ML e DL referidas, não é uma melhor alternativa em relação à utilização de *features* de tempo-frequência presentes em literatura. Contudo, é possível concluir a importância de registar mais dados de partos prematuros de EHG, com vista a melhorar as experiências futuras, e evitar a utilização de técnicas como a de SMOTE. Para além disso, abriu-se também a possibilidade da aplicação de uma rede neuronal complexa como o LSTM, com resultados promissores para o futuro, que podem ser eficazes quando aplicados na classificação de parto prematuro.

Palavras-chave: SMOTE, Machine Learning, Parto Prematuro, Electrohisterograma, LSTM

Table of Contents

List of Figures

List of Tables

Acronyms

AC- Accuracy
ADASYN- Adaptive Synthetic Sampling Method
ANN- Artificial Neural Networks
AUC- Area under the ROC Curve
cGC- Conditional G-causality
CNN- convolutional neural network
DNN- Deep Neural Network
DT- Decision Tree
EHG- Electrohysterography
EMD- Empirical Mode Decomposition
FFT- Fast Fourier Transform
FWH- fast wave high
FWL- fast wave low
IMF- Intrinsic Mode Function
IUPC- Intra-uterine pressure catheter
KNN- K-Nearest Neighbour
LDA- Linear Discriminant Analysis
LDC- Linear Discriminant Classifier
LMNC- Levenberg-Marquardt Trained Feed-Forward Neural Network Classifier
LOGLC- Logistic Classifier
MLP- Multilayer Perceptron
MRMR- Minimum redundancy maximum relevance
RF- Random Forest
PCA- Principal Component Analysis
PERLC- Perceptron Linear Classifier
POLYC- Polynomial Classifier
PSD- Power Spectral Density
QDA- Quadratic Discriminant Analysis
QDC- Quadratic Discriminant Classifier
RBNC- Radial Basis Function Neural Network Classifier
RNNC- Random Neural Network Classifier
ROC- Receiver Operating Characteristic
SAE- Sparse Auto Encoders
SMOTE- Synthetic Minority Oversampling Technique
SONIA- Self-Orgazined Network Inspired by the Immune Algorithm
SSA- Singular Spectrum Approach
SSAE- Stack Sparse Auto Encoders
SVM- Support Vector Machine
T-SNE-T-Distributed Stochastic Neighbour Embedding Method
TOCO- Tocodynamometer
TPEHG- Term-Preterm EHG dataset
TPEHGT-Term-Preterm EHG DataSet with Tocogram

WHO- World Health Organization
WPD- Wavelet Packet Decomposition

1. Introduction

1.1. Framework and Motivation

Preterm birth or premature delivery is defined by the World Health Organization (WHO) as the delivery of babies before 37 completed weeks of gestation and it is considered a serious health issue not only for the fetus but also for the mother.[1] Although some of these births tend to happen spontaneously, some of these are due to early induction of labor or cesarean birth. Moreover, most of the time these deliveries don't have a specific diagnosis, with some of the most common causes being multiple pregnancies, infections, and chronic conditions, like hypertension or diabetes. With 1 in 10 babies being born premature we can say that preterm birth is a modern problem, with its rates still growing. [1]

Prematurity is also the primary cause of death in the world in children under the age of 5 years old. Since babies born before 37 weeks are not usually fully developed, when they survive, they can face lifelong disabilities such as visual and hearing problems, learning and cognitive difficulties, and other health complications, like cardiovascular or respiratory issues, especially in low-income settings. [1]

Like in most health-related issues in the world, low-income settings are the most harmed population when it comes to inequalities in cheap and accessible care like warmth, breastfeeding support, and basic care for infections and breathing problems. The lack of these basic conditions leads to the death of half of the babies that are born at or below 32 weeks, in these types of populations. Even in middle-income settings, the poor use of technology is causing a rise in the number of disabled children who survive the neonatal period after preterm delivery. By comparison, in high-income settings, practically all babies born at this time survive. [1]

So, the problem seems to stand on the early diagnosis so the mother can be treated accordingly thus preventing prematurity.

Currently, the most used techniques are the intrauterine pressure catheter (IUPC), an invasive technique for the mother, and the tocodynamometry (TOCO) which does not work well for high body mass index (BMI) patients. The electrohysterogram (EHG) is a non-invasive method that records a signal related to the electrical activity that propagates through the uterine cells. From this signal, many features can be extracted and consequently analyzed to evaluate the difference between preterm and term delivery[2]. This technique presents the same results present in TOCO but with the advantage of working with high BMI mothers. As of now, instant labor classification is not possible in the raw EHG signals, so currently, there is still a gap in devices that can successfully predict preterm labor for women that don't present any risks. Even though this technique has very little application in the medical field, it is currently gaining more success among clinics and hospitals.

Through the years, countless of articles applied machine learning techniques to various EHG databases, since they are capable of predicting while adapting to new data. More recently deep learning methods have also joined the conversation of the prediction of preterm labor using EHG signals, also showing very promising results, but there is still a gap in the problem of preterm delivery [3]. The biggest gap is the lack of preterm observations, that limits the accuracy of the classifying algorithm. Authors throughout the studies, try to mitigate this problem by using synthetic oversampling techniques, like SMOTE or ADASYN, that can lead to a nun-realistic performances when used as the sole skewed class treatment [4]. Furthermore, to answer this complex problem authors try to use more features, including quantitative features from raw EHG readings on the time domain, frequency domain and time-frequency

domain to classify preterm labor [5]. However, as of now, no author has used the power spectra of the contractions from the EHG signal.

This dissertation was made in the context of the Uterine Explorer (UEX) Project, within the Faculty of Sciences of the University of Lisbon and the Faculty of Sciences and Technology at the New University of Lisbon.

In the present thesis, we're going to apply different machine learning algorithms and a deep learning network, in a publicly available database to predict preterm birth. With practical and cost-effective care, researchers suggest that over three-quarters of babies born prematurely could be saved. Therefore, there is a crucial need for an automated approach to detect and predict labor for pregnant women with a high risk of premature birth [5]. Such an approach can help mitigate the consequences of premature birth and provide better healthcare for both the pregnant woman and the fetus.

1.2. Research Goals and Expected Results

To issue the problems of preterm prediction, mentioned before, one must elaborate a strategy to find a solution to the problem. This dissertation will focus on the application of Machine Learning Techniques and Deep Learning suitable to this exact problem.

The following work tries to answer three main questions: 1) Can each point in the power spectra of contractions be used as features for predicting preterm labor? 2) Can the combination of oversampling the minority class using SMOTE and undersampling the majority class outperform the more common oversampling technique only using SMOTE? 3) Can an LSTM network be used for doing preterm labor predictions?

The main objectives are stated below:

1. **Comprehend the EHG signal and extract features for the TPEHG Database:** study and understand the EHG signal, to identify the best features to originate the study dataset. Additionally, a categorical feature, for classification, must be calculated based on the time of birth of each patient. This first step is essential to build our study dataset and start with the second step of classification.

2. **Apply and experiment different Machine learning algorithms and techniques in the study database:** this step involves applying different machine learning algorithms to classify preterm and term labor, with the objective of predicting preterm labor. Inside this objective, it is necessary to try different techniques associated with improving the performance of the different classifiers, including oversampling of the minority class, feature selection and hyperparameter tuning.

3. **Apply and experiment a Deep Learning algorithm to the study database:** this step involves applying a deep learning network to classify preterm and term labor, with the objective of predicting preterm labor. Within this step an oversampling technique will also be applied to the minority class in order to improve the results.

1.3. New Contributions

Through all the developments achieved inside the preterm labour prediction problem, we are still lacking a way to detect and predict a premature birth. The main problem associated with this is the lack of observations for preterm labour, which limits significally the algorithms classification power.

In literature, the authors tried to mitigate this problem with the use of techniques like SMOTE and using more and more innovative features.

Since the main purpose of this thesis was to find a way to predict preterm labour, we decided to innovate by using as features each point of the power spectra of the contractions extracted from the EHG signal and by using SMOTE as a technique of data compensation, before the train-test split, something never done before in literature.

1.4. Document Organization

This document in organized in six main chapters, which in a whole allow us to reach and present the goals and the expected results from this thesis. The theoretical background necessary to understand this work is introduced in **Chapter 2**, including the literature review, and in **Chapter 3** that introduces the concepts of preterm birth, the anatomy and physiology of the uterus and the EHG signal. Then, **Chapter 4** explains the materials and methods used in this thesis, including the dataset explanation, all Machine Learning Algorithms, as well as oversampling, feature selection and hyperparameters tuning techniques, and Deep Learning Algorithms. In **Chapter 5** the results and discussion are presented, and **Chapter 6** concludes this thesis, with some final remarks about future works.

2. Literature Review

Being that the subject of this thesis is "Preterm Labor Prediction Using Electromyography and Deep Learning Models", it was crucial to understand what type of research and discoveries have already been done, with the final goal of trying to innovate. For that reason, literature research is essential for our work of preterm labor classification, to grasp what has been tried and what is still missing.

Therefore, the literature research will be conducted around every work related to labor classification using electromyography data, including machine learning and deep learning techniques for classification. It is important to point out that some exceptions were made for papers that seemed relevant for the study, with other types of main objectives, excluding classification for preterm labor.

This research was summarized in Table 1, where it displays the author and the study objective, the database used in that study, the features used for classification, the algorithms used as classifiers and the imbalanced class treatment method performed, if applied, for each paper. The best classifying methods and the correspondent metrics results are also displayed, with some additional details that stood out from the study.

Table 1: Literature Review

Author	Database	Features	Classifiers	Imbalanced Class Treatment	Research Goal	Best Achieved Results	Comments
Idowu et al. (2014) [6]	TPEGH	Root mean square, median frequency, peak frequency and sample entropy	Artificial neural network classifiers: BPXNC, LMNC, NEURC,RBNC, RNNC,PERLC	SMOTE	Preterm Birth Classification	LMNC with 96% sensitivity,92% specificity,95% AUC and a 6% mean error rate.	The use of oversampling techniques influences the good results.
Ryu et al. (2015) [7]	TPEHG	Sample entropy	Linear classifiers	Subsampling a balanced dataset of 38 term and 38 preterm records, 100 times	Preterm Birth Classification	60,49% AUC	PCA is used for dimensionality reduction.
Hussain et al. (2015) [8]	TPEHG	----------	Recurrent Neural Networks: Elman, Jordan network and Layer recurrent neural network where each layer has a recurrent connection with a tap delay associated with it (layrecnet).	----------	Classification of EHG signals for prediction of term and preterm birth	RNN's can capture temporal behavior of the signals.	Applying RNNs as filtering method to increase uterine EHG signal to noise ratio value.
Hussain et al. (2015) [9]	TPEHG	Root mean square, median frequency, peak frequency, sample entropy	MLP,SONIA,K-Nearest neighbour, Decision Tree, Support Vector classifier,Fuzzy-SONIA, DSIA	Oversampling using a min/max technique.	Preterm Birth Classification	SONIA with 91,23% sensitivity, 94,51% specificity, 94,9% positive predicted value, 90,6% negative predicted value, 92,77% accuracy	----------

Sahid-Ahmed et al. (2017) [10]	TPEGH	8 frequency related features are extracted from 2 IMF's, using the periodogram as estimator of PSD	Support vector machine	----------	Preterm Birth Classification	Combination of 2 channels (7 features)with a 95,7% accuracy,98,40% sensitivity, 93% specificity and 95% AUC.	The 12 IMF's are obtained through the Huang-Hilbert transform (HHT).
Hoseinzadeh et al. (2018) [11]	TPEHG	Extraction of the features using an AR model followed by PSO for feature selection.	SVM with RBF kernel function	ADASYN	Preterm Birth Classification	SVM with 97.1% accuracy rate, 95% sensitivity, and 99% specificity	Application of EMD for extraction of the IMF's for the calculation of the wavelet coefficients for each IMF.
Jager et al. (2018) [12]	TPEHGT DS	Sample entropy, median frequency of the power spectra, peak amplitude of the normalized power spectra	Cross-Validation	----------	Preterm Birth Classification	Accuracy of 100% and 99,44% AUC for all records.	Features from the dummy intervals are better than the features obtained from the contraction intervals.
Shahbakhti et al. (2019) [13]	TPEHG	Root mean square of two IMF's using different channels	SVM	----------	Preterm Birth Classification	99,56% accuracy, 98,95% sensitivity and 99,30% specificity.	Application of EMD for feature extraction, trough the IMF's.
Chen et al. (2019) [14]	Icelandic 16-electrode Electrohysterogram	Sample Entropy	Stacked sparse autoenconder (SSAE)	----------	Preterm Birth Classification	90% accuracy, 92% sensitivity and 88% specificity, and 90% of AUC.	----------
Saleem et al. (2020) [15]	TPEGHT (13 term and 13	cGC, uGC, mGC, cDI, uDI, mDI measures	Quadratic discriminant	----------	Preterm Birth Classification	91% accuracy, 94% sensitivity, 95% specificity, 97%	Extract the features using Granger causal analysis of contraction

	preterm records)		analysis (QDA) based classifier			AUC when using features related to dummy and contraction intervals.	and dummy intervals. It focused on the synergy between electrical and mechanical conduct of the uterus during contraction and dummy intervals.
Peng et al. (2020) [2]	TPEHG	31 linear and non-linear features (root mean square, sample entropy, peak frequency, among others).	Random forest	ADASYN	Preterm Birth Classification	93% accuracy, 89% sensitivity, 97% specificity, and 80% AUC.	The data was divided into two groups, before and after the 26^{th} week of gestation, but the results were the same. The sample entropy was the feature that weighted the most on the results.
Oliver et al. (2020) [16]	TPEHG	RMS, median frequency, peak frequency, sample entropy.	Support Vector, Naïve Bayes, KNN, Gradient Boost, Decision Tree	----------	Preterm Birth Classification	SVM with 92% sensitivity, 94% specificity, 96% accuracy and gradient boost that also showed similar results.	Extract the features using PCA for dimensionality reduction of the data.
Degbedzui et al. (2020) [17]	TPEHG	Feature vector of time-varying spectral content of the EHG signal	KNN-Cos, KNN-Cor, SVM-RBF, SVM-Gauss	ADASYN	Preterm Birth Classification	SVM with RBF kernel function with 97,10% accuracy, 95% sensitivity, 99% specificity.	The spectral properties of the signals are extracted using a centroid frequency method.The classification is done for the extracted features from each of the three channels and then compared.

Chen et al. (2020) [18]	TPEGHT	20 entropy features.	Sparse autoencoder based deep neural network (SAE-based DNN network)	----------	Preterm Birth Classification	SSAE with the DNN classifier with 98,2% sensitivity, 97,74% specificity and 97,9% accuracy.	Features are extracted by sample entropy and wavelet entropy, from the EHG and TOCO signals before classification.
Esgalhado et al. (2020) [19]	Icelandic 16-electrode Electrohysterogram	Signal envelope features	5 energy burst delineation methods: Wavelet energy, Teager energy, root mean square (RMS), squared RMS, and Hilbert envelope	----------	Contraction detection using EHG	RMS with 97,15% accuracy for contraction detection and 89,43% for delineation and 0,63% of FNR. The wavelet energy method was the second-best method.	
Xu et al. (2021) [4]	TPEGH	Root mean square, peak Frequency, median frequency, sample entropy	Linear Discriminant Classifier (LDC), Support Vector Classifier (SVC), Decision Tree (DTC), Gradient Boosting Classifier (GBC)	ADASYN and SMOTE at the optimal sample balance coefficient.	Preterm Birth Classification	The SVC-based classifiers are the top performers. The use of SMOTE showed better results than the ADASYN.	The authors create a sample balance coefficient, that has a positive correlation with the overall accuracy. Additionally, training with the optimal number of synthetic samples leads to an improved performance. The importance of the features decreases when the number of synthetic samples is bigger.
SAĞLAM	TPEGH	Average Frequency, Median Frequency, Peak Frequency, Peak	Naïve Bayes, SVM, Kstar, Random	SMOTE	Preterm Birth Classification	CNN shows the best results with 87,67% accuracy, followed	SMOTE does not improve the results in the SVM, Decision

et al. (2021) [20]		Magnitude root mean square ratio, Sample Entropy, Shannon Entropy, among others	Forest, Decision Table, KNN, CNN			by SVM, Random Forest and decision table with 87,33%.	Table (worsens them both) and RF (no difference in the final results).
Allahem et al. (2022) [5]	7271 30 minute-long datasets extracted from the 5 datasets: Icelandic 16-electrode Electro hysterogram Database, TPEGH, TPEGHT, CTU-CHB Intrapartum Cardiotocography Database, OB-1 Fetal ECG Database.	**Machine Learning Approach:** Mean frequency, peak frequency and median frequency features. **Deep Learning Approach:** Mean Frequency, Peak frequency and Median frequency, pregnancy gestational age, pregnant woman's age and parity.	**Machine Learning Approach:** Decision Tree, Random Forest, Support Vector Machine, Naïve Bayes **Deep Learning Approach:** ANN	Creation of new EHG balanced datasets using Gretel Labs, Inc. service on Python (https: //gretel.ai/)	Labor Detection	**Machine Learning Approach:** RF showed better results with 95,7% AC, 99% AUC, 3,6% FNR and 4,6% FPR. **Deep Learning Approach:** the ANN classifier has better results than the RF with 98% AC, FNR 0,01% and 0,00,1% FPR.	Amplitude and Frequency are essential to analyze uterine contractions. Grid Search is used as tuning technique.
Mohammadi et al. (2022) [21]	TPEHG	Root mean square, sample entropy, mean Teager-Kaiser energy (MTKE)	kNN, SVM, Decision Tree	Stratified 10-fold cross-validation	Preterm Birth Classification	SVM with polynomial kernel	The features are extracted from 2 IMFs, after empirical mode decomposition of the signal.

With the research shown in **Table 1**, it is possible to see all the different classification algorithms that have already been utilized. Through all of them, we were able to pick out some papers that stood out for our work specifically. Hussain et al. (2015) experimented with recurrent neural networks for the prediction of term and preterm delivery, with positive results. SAĞLAM et al. (2021) [20] showed that SMOTE in general improved the classification results while pointing out that CNN was the highest-performing algorithm, concluding that LSTM networks should be studied more in depth. Allahem et al. (2022) [5] innovated when using an amalgamation of databases in order to combat the imbalanced data problem, although mentioning the application of a synthetic oversampling technique, but inferring that the use of this would lead to deceiving classification results. Additionaly, Mohammadi et al. (2022) [21]also tried to issue the combat the skewed class problem by applying a Stratified 10-fold cross-validation, a technique usually used for validation in imbalanced datasets. In general, in the works that used an imbalanced class treatment technique, like ADASYN and SMOTE, they all showed very positive results [17]. In terms of the type of features used, most of the authors used similar features like sample entropy, peak frequency, root mean square, median frequency, among others, extracted from the pre-processed EHG signal [16][13], [21].

It is also important to mention the limitations encountered throughout our literature research. Vandewiele et al.[22] , points out that synthetizing new samples before partitioning the dataset into training and testing sets can lead to un-realistic and an overly optimistic prediction results on imbalanced data. This can be misleading since most authors choose this technique. In the other hand since the TPEHG Database contains a very small number of observations, by introducing synthetic samples after splitting the dataset might lead to worse results.

Throughout the literature review, we can see that a lot of different algorithms were used for classification, from SVM, RF to Naïve-Bayes. With the purpose of seeing the most predominant classifers, a simple wordcloud was done using as keywords the acronym of each type of classifier. The wordcloud results are present in **Figure 1.** The words that appear in bigger and bolder lettering meaning that they are the most common in our research, which includes the RF, SVM, DT and KNN algorithms.

Figure 1: Wordcloud representing all the different algorithms present in literature. SVM, DT, KNN and RF appear in bigger and bolder lettering, meaning that they are more predominant

Another limitation appears on the number of samples to be synthetized when using SMOTE. According to the Chawla N. et al. [23], the authors behind the SMOTE technique, the combination of oversampling the minority class and undersampling the majority class can achieve better classifier performance. In most papers that use SMOTE, the number of samples synthetized in the minority class are the same as in the majority [20].

Lastly, most authors don't report important evaluation metrics like F1-Score (F), Area under the Curve (AUC), false positive rate (FPR) and the false negative rate (FNR) [5].

3. Preterm Birth and the EHG Signal: Theoretical Context

3.1. Premature Birth

Preterm birth or premature delivery is defined by the World Health Organization (WHO) as the delivery of babies before 37 completed weeks of gestation and it is considered a serious health issue not only for the fetus but also for the mother. [1]

According to data from the US National Vital Statistics this is also a problem that affects more multiple pregnancies in comparison with singleton. All these pre-term births can be divided into three sub-categories: extremely preterm (less than 28 weeks), very preterm (28 to 32 weeks), late preterm (32 to 37 weeks) as shown in **Figure 2** [24]

Prematurity is also the primary cause of death in the world in children under the age of 5 years old. Since babies born before 37 weeks are not usually fully developed, when they survive, they can face lifelong disabilities such as visual and hearing problems, learning and cognitive difficulties and other health complications, like cardiovascular or respiratory issues, especially in low-income settings.

According to the Portuguese Pediatrics Society (SPP), 8% of babies in Portugal are born premature and 1,2% of these labors occur below 32 weeks. Although this is a high number, Portugal is in the 9th place between 162 countries with the lowest mortality rate, with a rate of 1,8 in 1000 live births, a consequence of early detection and intervention in premature birth. [25]

Category	Gestation (weeks)
1. Extremely preterm	Under 28
2. Very preterm	28 to under 32
3. Moderate preterm	32 to under 37
a) Early moderate preterm	32 to under 34
b) Late moderate preterm	34 to under 37

Figure 2: Preterm Birth subcategories according to gestational week [26]

3.1.1. Causes of Premature Birth

These premature labors can result from three clinical conditions: medically indicated or medically induced, preterm premature rupture of membranes (PPROM) and spontaneous preterm birth, with the last one being responsible for most of the cases. Almost 2/3, or 67% of these deliveries don't have a specific diagnosis [27], with the other 33% being associated with multiple pregnancies, infections, and chronic and genetic conditions, like hypertension or diabetes. In the following **Table 2**, it is shown some of the risk factors associated with preterm birth [28].

Table 2: Recognized risk factors associated with clinical presentation of preterm birth [28]

Medically induced preterm birth		Preterm premature rupture of membranes	Spontaneus preterm birth causes
Maternal Causes	**Fetal Causes**		
Pregnancy hypertension and vascular disorder Medical illness or chronic conditions Obstetrical compilation Antepartum bleeding Maternal age >35 years	Intrauterine growth restrition Unstable fetal condition Fetal anomaly Multiple pregnancies	Infection Uterine distension Cervical anomalies Afro-American ethnicity Disadvantaged population	Previous preterm birth, preterm labor Low body mass, poor weight gain Strenuces physical workland, ergonomic factors Uterine anomalies Psychesocial stress Lifestyle, smoking Drug abuse Maternal age <18 years Unknown

3.1.2. Consequences of Preterm birth

Premature birth has severe health consequences or even risk of death in newborns, especially in low income countries. Like in most health-related issues in the world, low-income settings are the most harmed population when it comes to inequalities in cheap and accessible care like warmth, breastfeeding support and basic care for infections and breathing problems. The lack of this basic conditions leads to the death of half of the babies that are born at or below 32 weeks, in these types of populations. Even in middle-income settings the poor use of technology is causing a rise on the number of disabled children who survive the neonatal period after preterm delivery. By comparison, in high-income settings, practically all babies born at this time survive. Another factor that contributes to a disparity in the mortality rate between poor and rich countries is that often premature babies will need further hospitalizations during development, creating an even bigger problem in countries where the medical care is lacking and where those families will be hindered [1][3].

One of the ways of minimizing premature birth rate and all its consequences in a way that is inclusive to all settings, is to improve the early detection of labor, by monitoring the woman's biochemical or biophysical signals throughout the pregnancy, so that early medical intervention to the fetus and the mother is provided as soon as possible. This will allow the reduction of the treatment cost on children born premature, decreasing the number of deaths in children born premature [1][3].

The most common and practical procedures for monitoring pregnant woman is uterine activity carried out by medical staff. [17] Some of the methods that are also used to measure contractions in the pregnant uterus, include intrauterine pressure catheter, tocodynamometry and more recently, electrohysterography that are explained in this chapter.

3.2.Uterine Contractility

Uterine contractions are connected to electrical events in the myometrium. The myometrium is the middle layer of the uterine wall, as seen in **Figure 3**, and it is composed of smooth muscle cells, called uterine myocytes, as seen in **Figure 4**. These cells are the ones responsible for the underlying electrical activities in the form of action potentials in the uterus, resulting in uterine contractions. The propagation of the electrical activity happens due to a grouping of connexin proteins in the gap junctions of the myometrial cells, that allows them to be electrically connected. These cell-to-cell contacts are usually low, with a small electrical conductance, however when contractions occur these gap junctions increase allowing coordinated and effective contractions. This electrical activity and cell contact is a direct cause of the uterine volume (chronic stretch) and ovarian hormones levels (mostly estrogen) on resting membrane potentials. [29], [30]

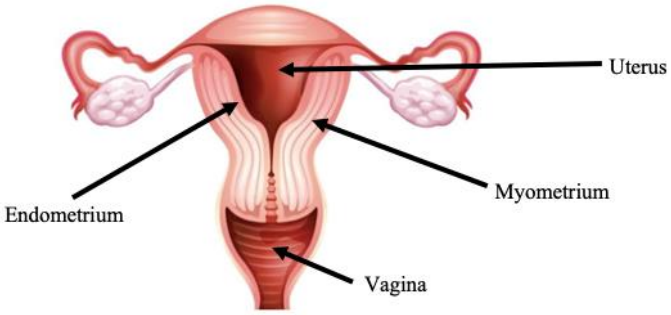


Figure 3: Anatomy of the Uterus [31]

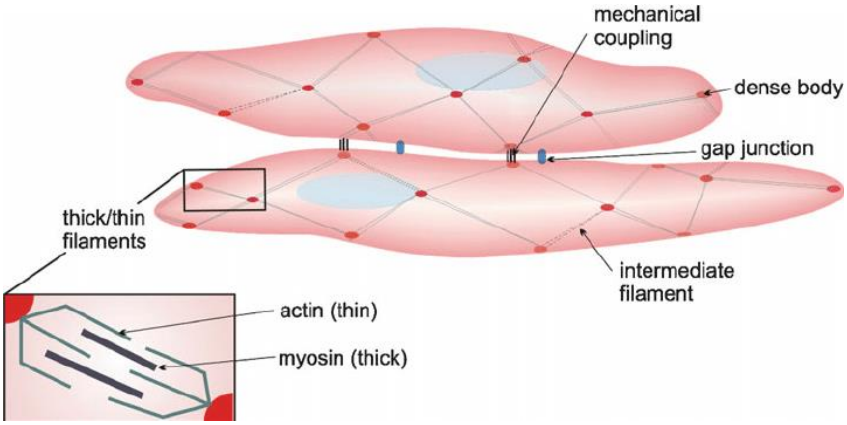


Figure 4: Architecture of myometrial cells [31]

In terms of the biochemical process the action potentials are a result from voltage- and time-dependent changes in membrane ionic permeabilities of calcium (Ca^{2+}), potassium (K^{+}) and sodium ions (Na^{+}). In the uterus an inward current of Ca^{2+} ions and Na^{+} ions will cause a depolarizing phase on the action potential. In the preterm myometrium, a "plateau-type" action potential occurs from a combined effect of a constant inward of Ca^{2+} ions or Na^{+} ions and a decrease in the voltage-sensitive outward current. [32], [33]

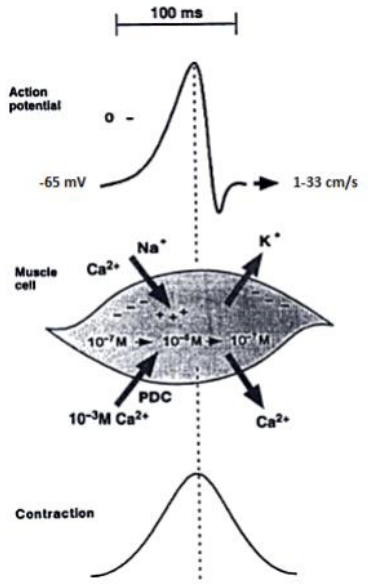


Figure 5: Action Potentials in the Uterus [32]

The amplitude, frequency and duration of these contractions are determined by how frequent the burst of energy happens across the myocytes, the total number of cells that are simultaneously active during the burst and how long they last. Therefore, uterine contractions are related to the electrical properties and excitability or conductivity of the uterine myocytes. [32]

For premature birth, there exists an early onset of synchronized contractions of the uterine muscle cells. These activities can be measured through electrohysterogram (EHG). [29]

3.3. Uterine Contraction Monitoring Devices

With the goal monitoring uterine contractions, a series of systems were created that are still in use now, with both of them being inadequate for preterm labor risk evaluation. These are the following and are represented in **Figure 6** [34]:

Intrauterine Pressure Catheter (IUPC): it is considered the Golden Standard for monitoring uterine contractions since allows the analysis of the frequency and the intensity of contractions more accurately. One of the big downsides is that this technique involves the insertion of a catheter, which requires the woman with ruptured membranes, and it might lead to fetal and placental damage, infection and/or uterine perforation.

External Tocography (TOCO): this method in also widely used in contractions monitoring although it shows some limitations. The TOCO shows a degraded predictive value since there is a mismatch between the tocogram amplitude and the strength of the uterine contraction. Additionally, it is inaccurate for high body mass index mothers (BMI) since the ultrasound has difficulties penetrating fatty issue [35].

(a) **(b)**

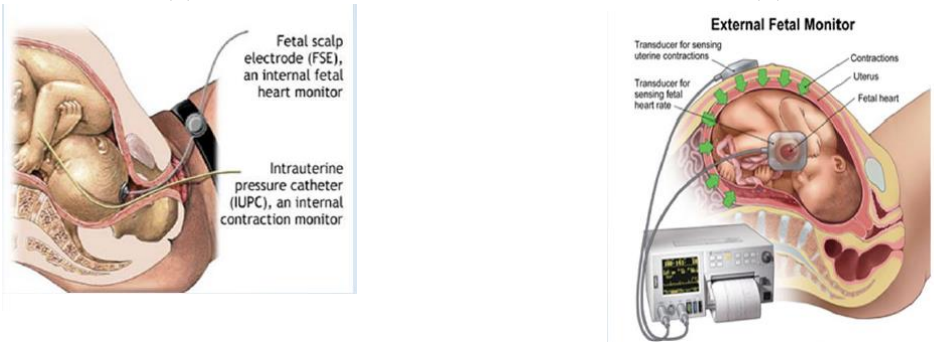


Figure 6: Uterine Contraction monitoring devices used more frequently in a clinical setting. (a) Intrauterine Pressure Catheter (IUPC). (b) External Tocography (TOCO) [34]

3.4. Uterine Electrohysterography (EHG)

Electrohysterogram (EHG) is technique dating as early as 1931, consisting in a non-invasive method that involves the placing of electrodes on the abdomen of pregnant women, that will record a signal related to the sum of the electrical activities that propagate through the uterine cells to the abdomen of pregnant women. This allows that information like frequency and length of uterine contractions, and contraction power of the uterus, are recorded and then used to identify if labor is occurring or not. [3]

This is an advantageous technique since it gives us information from the electrical activity generated at the muscle fiber level in a non-invasive way.

During most of the pregnancy the uterine electrical activity is very low, consisting of infrequent and low amplitude signals. However, during preterm and term labor EHG activity is higher in frequency and with large amplitude, a consequence of the changes in intrauterine pressure and pain sensation. Therefore, this technique can be very helpful for contraction-monitoring in term and preterm labor since it can differentiate preterm and term contractions, by distinguishing the transition from non-labor to labor states in the myometrium, something other techniques encounter challenges on [29][36]

There is an extensive research background that found measuring electrical activity from electrodes placed directly on the uterus very successful for monitoring uterine contractility.

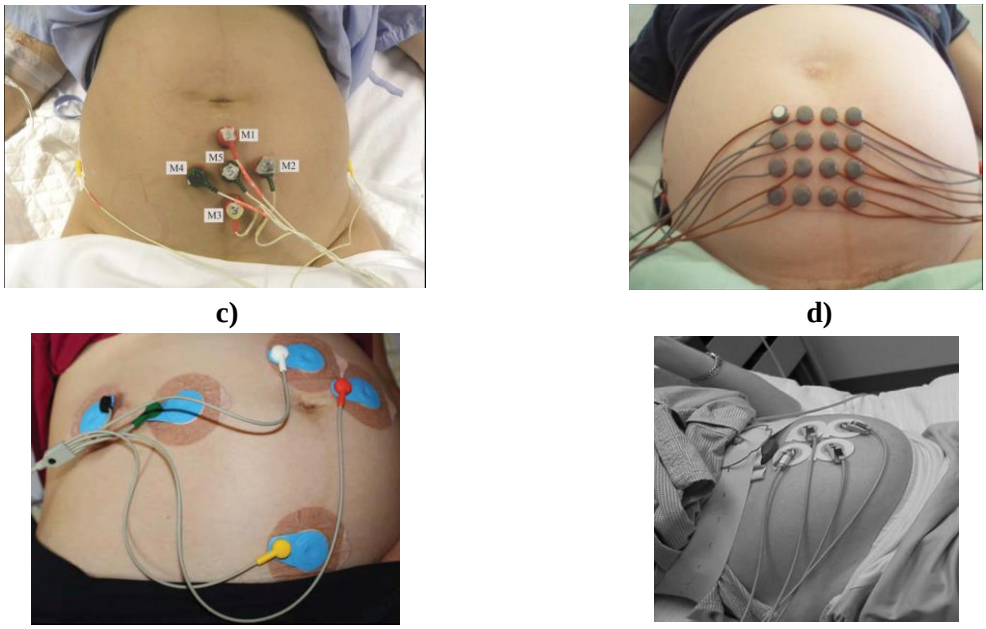
Comparing to other techniques, EHG produces very similar results to tocodynamometry (TOCO) and Intrauterine Pressure Catheter (IPC), although with better results than the TOCO device and with the advantage of being non-invasive when comparing to the IPUC. Therefore, EHG is gaining more attention each day, since it is a non-invasive, low-cost, and real-time technique [37].

Even though EHG as gained a lot of attention recently by producing very good results, its applicability has still raised some issues. One of these is the interpretability of the results since the signal records are complex and hard to interpret. Additionally, the collection of records is still very hard, especially for preterm labors since this technique is not a clinical practice [29].

As mentioned before, raw signals from the myometrium are obtained through the placement of bipolar electrodes adhered to the abdominal surface. Typically, four electrodes are used in most studies, although some studies have used 2, 16 and even 64 small electrodes. The problem with a small number of electrodes is the structural and functional complexity of the uterus. The uterus is an assembly of stochastic, nonlinear biological mechanisms interacting with a fluctuating environment. For that reason, a higher

number of electrodes will be preferred [38]. In **Figure 7** it is possible to see how the electrode placement was done in some studies, since different layouts are on trial by different research groups.

a) **b)**



c) **d)**

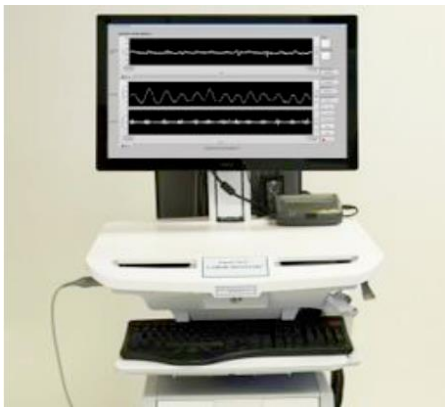
Figure 7: Electrode placement on the EHG from different authors. a) Ye-Lin Y et al.[39] **b) Alexandersson A. et al.** [40] **c) Hayes-Gill B. et al.** [41] **d) Garfield R. et al.**[42]

3.4.1. Commercial Monitoring Systems for the EHG signal

(a) Nemo Healthcare [43]

(b) Monica HealthCare Inc.[44]

(c) The SureCALL® Labor Monitor®[45]



(d) OB Tools[46]

Figure 8: Existing monitoring devices for the EHG signal

At present, the EHG signal can be obtained through different monitoring equipment available to the public, with one of upsides of the increased sensitivity to patients with a high BMI. These systems were designed to detect uterine contractions and fetal electrocardiogram, replacing TOCO, but still fail to provide contractions identification as well as labor prediction, being uncapable of assessing if there is a risk of preterm labor. In **Figure 8** it is possible to see the different available systems using the EHG with the American Food and Drug administration (FDA) and European Commission (EC) marks.

3.4.2. EHG Signal Characteristics

As mentioned before EHG is one of the best techniques to measure non-invasively the electrical activities of the muscle cells related to uterine contractions, but it has a very weak signal (from 0 to <5 Hz). This comes with a price, since the frequency range of other types of electrical activities from the mother and the fetus can also be recorded (called noise), overlapping the spectra of the signal of interest. One of the strongest signals that corrupts the EHG signal is the maternal electrocardiogram (ECG), that remains present even after the analog filtering during the assessment. Others like fetal electrocardiogram, maternal

respiration, motion artifacts, electromagnetic noise from external devices, as seen in **Table 3,** also contribute for the signal noise. Therefore, the solution is performing a de-noising step before utilizing the signal data, to improve the accuracy of preterm classification. [3]

Table 3: Frequency range for the main noise in the EHG signal [3]

Noise	Frequency Range
Maternal ECG	1.38 to 1.5 Hz
Maternal Respiration	0.2 to 0.34 Hz
Electromyography noise	Above 30 Hz
Power supply interference	50 or 60 Hz

The frequency of the EHG signal can be classified in two waves: fast wave and slow wave. The frequency on the fast wave varies between 0.01 to 0.03 Hz, and the slow can also be divided into two categories, fast wave high (FWH) with frequency ranges of 0.2 to 0.45 Hz and fast wave low (FWL) with frequency ranging from 0.8 to 3 Hz, including. These last waves are connected since FWH is related to the excitability of the uterus while FWL is related to the electrical activity of the uterine muscle cells. Moreover, the phasic uterine contractions are triggered by slow and fast waves.[3][38]. **Figure 9** shows an example of the representation of an EHG signal.

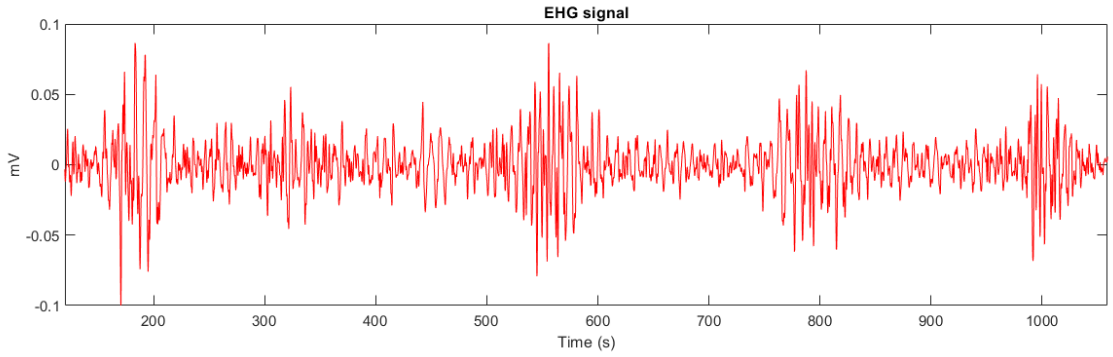


Figure 9: EHG Signal. The peaks correspond to different contractions

A way of denoising the signal is applying digital filters. The problem with this method is that some of the frequencies from the noise overlaps with the EHG signal, as you can see from **Table 3**. Consequently, other techniques must be applied to eliminate noise with a common frequency range that characterizes uterine contraction. Various denoising techniques have been approached in literature. Leman et al. [47] based on the success of the wavelets as a denoising tool, proposed the application of the redundant wavelet packet transform. Hassan et al. 2011 [48] applied a combination of the canonical correlation analysis and empirical mode decomposition (EMD) for removing the noise without losing information on propagation. Ryu et al. 2015, proposes a novel method where multivariate empirical mode decomposition (MEMD) is applied before extracting any features, showing better results compared to the Fourier-based prefilter. [7]

4. Materials and Methods

4.1.Data Description

The electrohysterogram records used in this work belong to the Term-Preterm EHG Dataset (TPEHG) Database stored in PhysioNet [49]. The records were collected from 1997 to 2005 at the University Medical Centre Ljubljana, Department of Obstetrics and Gynecology, during regular check-ups of pregnant women within the 22nd and 32nd week of gestation, with a total of 300 records, whose distribution can be seen in **Figure 10**. From these records, 262 were obtained from women whose delivery was on term (above 37 weeks) and a total of 38 records, were obtained from pregnancies that ended prematurely (below or equal to 37 weeks). The majority class can be easily seen as the term labors and the minority class as preterm labors. From 26 to 29 weeks we can also see that there is a lot of recordings, indicating that the recordings were not continuous during the pregnancy of all the individuals, which can lead to biased results. [49]

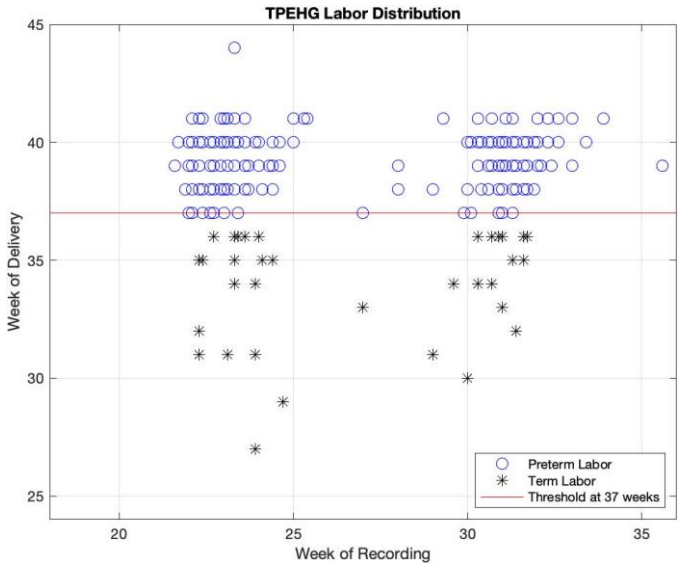


Figure 10: Scatter plot of labors represented by week of delivery in relation to the recording time for the EHG records, for the TPEHG database. At 37 weeks we have the threshold represented since it is the week for differentiating preterm from term birth.

Each record is composed of three bipolar channels (S1, S2 and S3), recorded from 4 electrodes (E1,E2,E3 and E4) that are placed in two horizontal rows, separated 7 cm apart, on the abdominal surface. The placement scheme can be seen in **Figure 11** and is later explained.

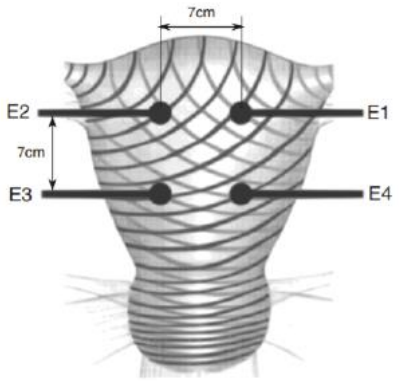


Figure 11: Electrode's configuration in the recordings of the TPEHG Database, placed on the abdomen, above the uterine surface of the pregnant individual. [49]

- The first electrode (E1) was placed 3.5 cm to the left and 3.5 cm above the navel.
- The second electrode (E2) was placed 3.5 cm to the right and 3.5 cm above the navel.
- The third electrode (E3) was placed 3.5 cm to the right and 3.5 cm below the navel.
- The fourth electrode (E4) was placed 3.5 cm to the left and 3.5 cm below the navel.

As mentioned before, 3 bipolar channels were calculated from these records. This was done by measuring the differences between the electrical potentials of the electrodes, following this order:

- First Channel (S1) = E2–E1
- Second channel (S2) = E2–E3
- Third channel (S3) = E4–E3

Bipolar channels have the advantage of reducing noise of the maternal electrocardiogram, electrode movements or respiratory movements of the signal. This happens due to the noise of the monopolar signals being identical, therefore being eliminated when the subtraction is done.

The work presented on this thesis consist in the original dataset, where the final records have a duration of 30 minutes each, with a sampling frequency of 20 Hz per channel with a resolution of 16-bits with the amplitude range of ± 2.5 millivolts. Furthermore, it is important to mention that this dataset has a variety of features collected from the gestation to parity, previous abortions, existence of hypertension, diabetes or bleeding, smoker status, among others. Although not used in this work, this is very important since these categories are all reasons linked to preterm labor.[49] From the original dataset, only channel 2 (E2-E3) was chosen since according to previous work on this matter, the vertical signal has a higher variation of signal potential. [19] [34]

4.2.Uterine Explorer Tool

Additional to the denoising of the signals, extracting the contractions from the original EHG signals is also very important, especially for the work presented in this thesis.

The Uterine Explorer (UEX) project is an application tool developed by the *Department of Electrical Engineer and Computers of NOVA School of Science and Technology*, on *MathWorks*® software, which main goal is the processing of EHG signals, as well as contraction extraction. This tool was implemented in this thesis, to detect and delineate the contractions present in the EHG signals. Below, the UEX welcome screen can be seen in **Figure 12.**

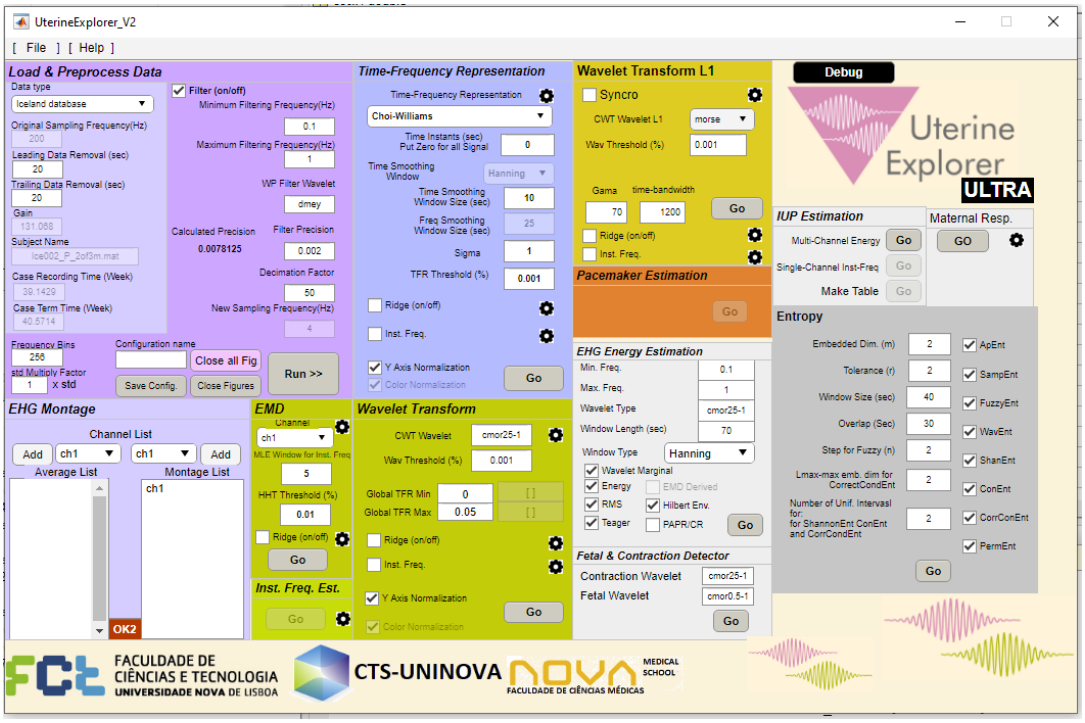


Figure 12: UEX homepage for data processing

4.2.1 Data Processing

The TPEHG database authors choose an initial sample frequency of 20 Hz. The UEX platform includes an array of EHG processing algorithms. In this project we worked with a decimation rate of 5, ending with a final sample frequency equal to 4 Hz. Additionally, a wavelet band-pass filter with bandwidth 0.1 and 1 Hz was selected according to reference values found in literature. An example output can be seen in **Figure 13,** of a preprocessed EHG signal for a specific patient.

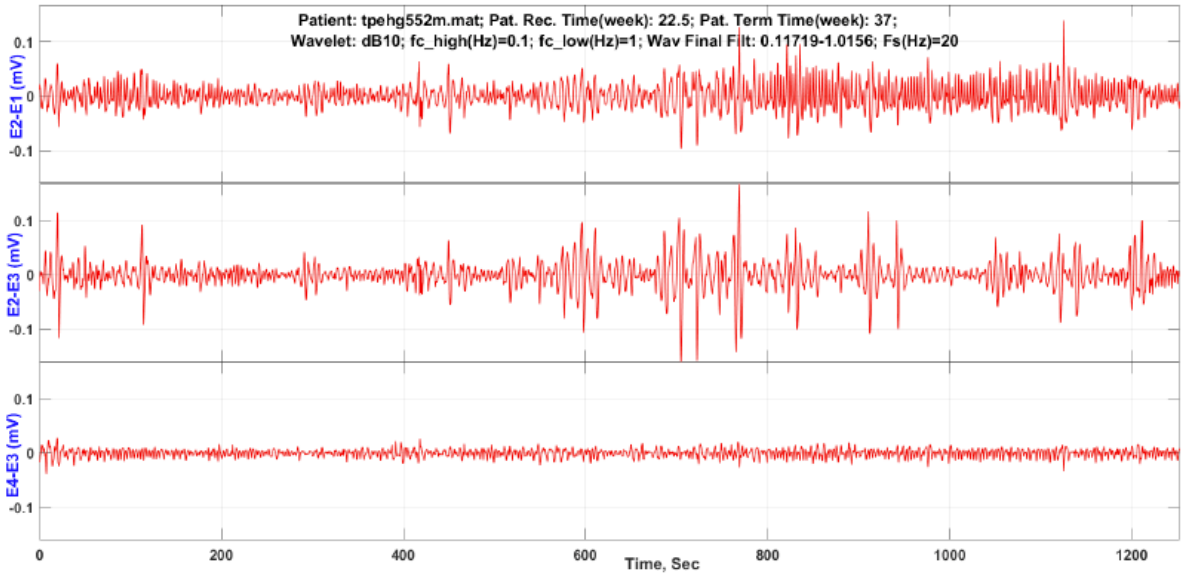


Figure 13: Three pre-processed EHG channels for patient tpehg552m are represented. The contractions correspond to energy bursts above the uterine baseline activity

4.3. Machine Learning

As mentioned before, one of the goals of the present work is predicting preterm birth using machine learning (ML) techniques. So, what is machine learning?

ML is a branch of artificial intelligence (AI) and computer science that uses data and algorithms to mimic the way humans learn while experiencing things, whilst improving its performance at each use or experience. [50] Examples of machine learning pass through the first game of checkers on an IBM 7094 computer in 1962 to Netflix's recommendation algorithm. [51]

Through the years machine learning has become one of the most used tools to solve problems related with analyzing large and complex data sets. Using statistical methods, algorithms are trained to detect automatically meaningful patterns in the data, in a process that we call classification and prediction. [51]

Machine learning tools are also special since the behavior of the programs adapts to their input data. One can also say that the algorithms are learning or training. This happens every time ML algorithms build models from data. [52].

Machine learning models can be applied to various fields, from psychology to artificial intelligence. Real world problems, like the ones in medical diagnosis, are perfect candidates for applying these techniques since they are highly complex. Therefore, machine learning has the capability to solve a variety of problems. In this way, a learning problem is constituted by three features [53]:

- Task classes (The task to be learnt)
- Performance measure to be improved
- The process of gaining experience

To evaluate the overall performance of a model to see if it can answer to the problem in question, six generic steps must be followed, these are the following (seen in **Figure 14**).

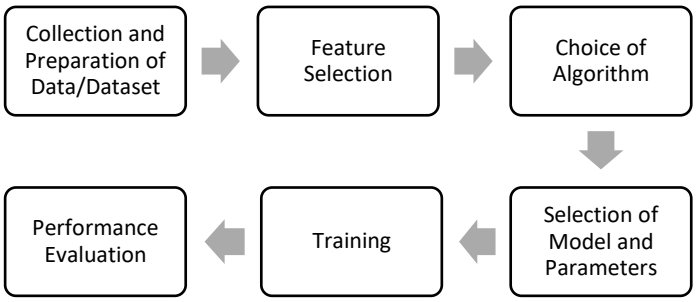


Figure 14: Steps to evaluate the performance of a machine learning model [53]

1. **Collection and Preparation of Data:** the first step is the collection and preparation of data in a structured format to be used in the algorithm. In our case the preparation of the data will include the signal pre-processing.
2. **Feature Selection:** select the features and relevant variables for the learning problem.
3. **Choice of Algorithm:** choose the appropriate machine learning algorithm for the problem.
4. **Selection of Models and Parameters**: manual selection, based on different criteria, of the most appropriate model and values for parameters of the same model.
5. **Training:** training of the chosen model, using a part of the dataset as training data, known as training set.
6. **Performance Evaluation:** this is the last step before the real-time implementation of the system, where the model is tested with new data, known as testing set, in order to evaluate through different performance parameters like accuracy and precision, if the model is learning and can indeed be validated and used.

Based on the previous information, one can say that Machine learning is, after all, a learning process where the data is trained and, in the end, tested. ML is constituted by several subfields that will include all these steps. Here we will be talking about the two most relevant subcategories inside ML to this research: supervised and unsupervised, mentioned in **Figure 15**. [53]

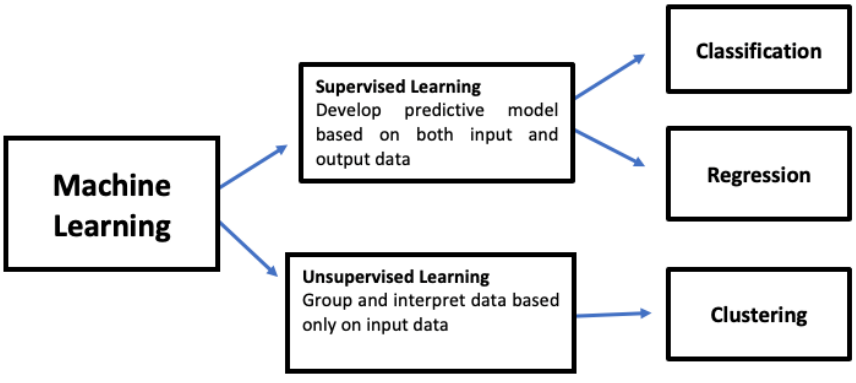


Figure 15: Machine Learning Branches (Adapted from [53])

4.3.1. Supervised Learning

Supervised learning is the most important and most used domain in machine learning. In supervised learning, algorithms learn through examples, just like humans. Humans use the knowledge gained from past experiences in order to improve their capacity towards real-world tasks, but since algorithms cannot experience real-world events, they will learn from previously collected data called a training set. The learning algorithm will then find patterns in the data and construct mathematical models that will be used to classify/predict values from previously unseen data [54]. After this the models will be evaluated on their qualificative/predictive capacity through different statistical measures. [55]

A training set can be defined as a set of labeled data, that is: various input observations, called features and the corresponding correct output, called labels [54]. In this phase the algorithm will compare the current output with the correct one, with the goal of finding the mistakes and improving the precision of the model. After the training, the learning model will predict the label of an observation based on a known set of features on the testing dataset, where the precision of the model is also evaluated.

A general scheme of the process can be seen below in **Figure 16:**

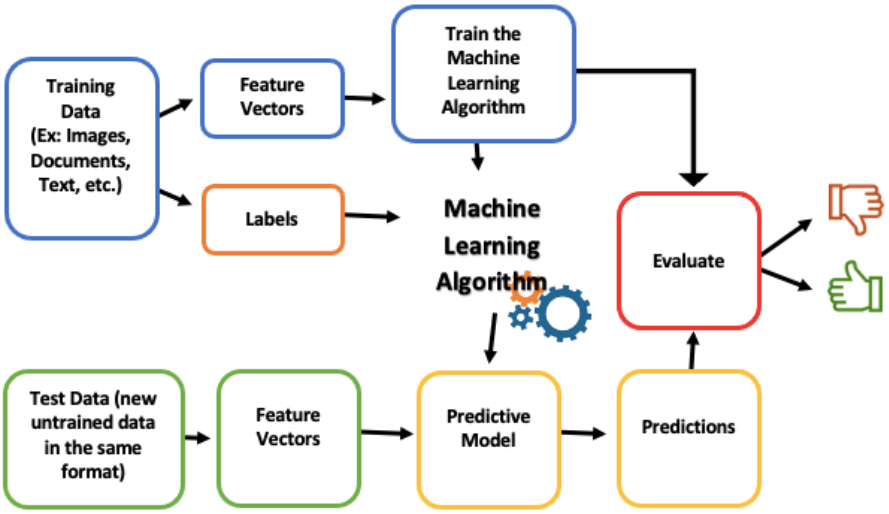


Figure 16: Supervised Learning Training and Testing Process Scheme (Adapted from [56])

Supervised learning can be divided into two main models, based on the type of the labels: classification models (classifiers), where the resulting label is discrete, and regression models, where the label is continuous. This means the following: [54]

- **Classification model**: predicts discrete values, in other words, classifies data into different categorical outcomes (labels). Ex: whether a patient has cancer or not.
- **Regression model:** predicts the numerical value of an item. Ex: predicting house prices. [57]

Although they have different purposes, the goal remains in achieving the best and more precise model, minimizing the difference between the predicted value and the real value.

The previous scheme can be easily explained with a simple example of a classification model (shown in **Figure 17**). Suppose that the training set is constituted by different types of fruit shapes, the features group, while the name of said fruit shapes are the labels for each one of them. This set is then used to train the machine learning algorithm, that then gives a predictive model. To construct the predictive model we input data containing a set of known features, called the testing set, that will return the labels for each one, so the corresponding name to each fruit shape [57]. These features are then evaluated according to the precision of the prediction, and if it does not perform well, the process repeats. In this example, all of our fruits in the prediction pile were rightly labeled, concluding that our machine learning algorithm can classify each fruit shape into the correct name (category).

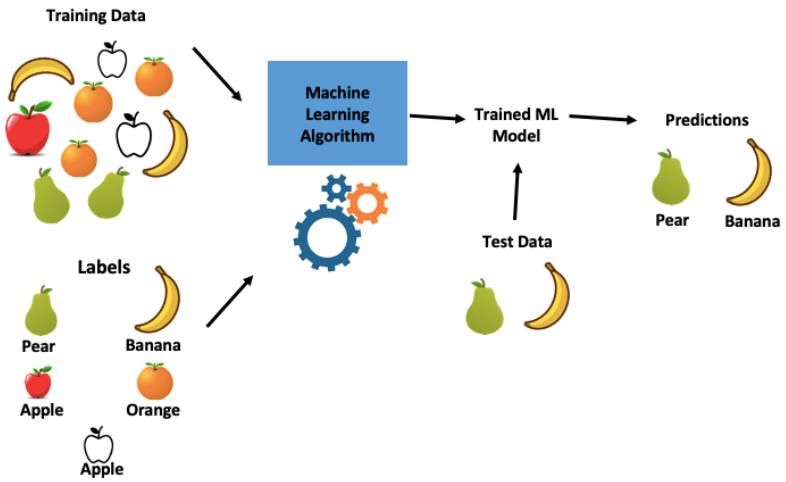


Figure 17: Classification Model Mechanism

A selection of supervised learning algorithms examples is shown below:

- **Decision Trees:**

A decision tree is one of the most popular approaches, either for classification or for regression problems. A decision tree has a hierarchical tree shape, consisting of a set of nodes and branches that form a root tree. There will be three kinds of nodes: a root node, a single node with no incoming branches, and two outgoing branches from the root: the decision nodes, which will help us split the data with a specific condition, and finally the leaf nodes that will help us decide the class of a new data point, by representing all the possible outcomes within the dataset. [58], [59]

The root node represents the complete dataset, that will be split into consecutively smaller *subdatasets*, following different decisions. The splitting process will be repeated in a top-down, recursive manner until most of the objects have been classified into a specific label. The classification starts in the root node and finishes in the last leaf node. In a way of avoiding overfitting and bias, decision trees are more suitable for smaller datasets, since when a tree grows, it becomes increasingly difficult to maintain the purity of each leaf node [59] . In **Figure 18** shows the architecture of a decision tree model.

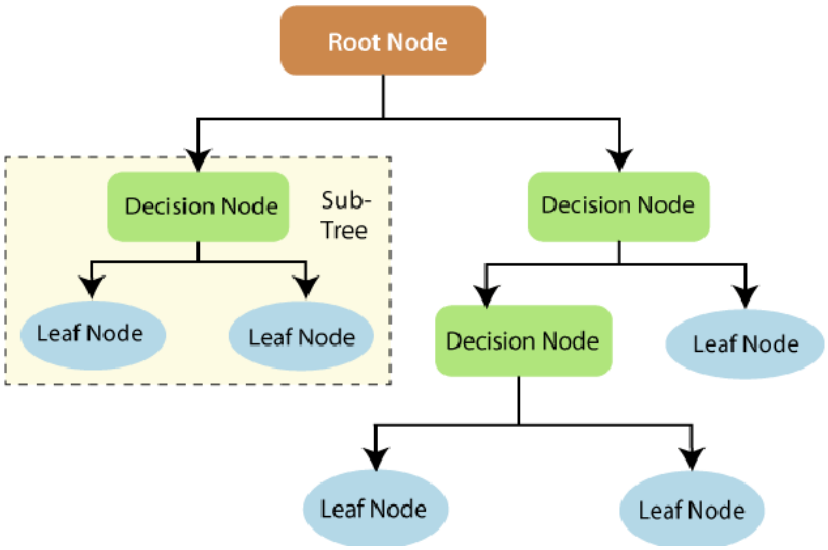


Figure 18: Architecture of a Decision Tree Model [59]

- **Random Forest (RF):**

Random forest, a very commonly used ML algorithm, can be used for both classification and regression problems. As previously mentioned, decision trees can be associated with overfitting and bias when dealing with larger and more complex datasets. The Random Forest algorithm also faces that problem by combining the output of various decision trees to reach a single result, giving a more precise prediction.

For this algorithm three main hyperparameters will be selected: node size, number of trees and the number of features sampled. Each decision tree consists of a bootstrap sample, where data is drawn repeatedly with replacement from the training set. Through feature bagging, where the algorithm randomly selects a subset of features, the randomness in the dataset is increased and the correlation problem between the trees is reduced. Therefore, for each decision tree in RF, the algorithm will train different features and consequently give different predictions.

These predictions will vary depending on the type of problem. In the regression model each individual decision tree will be averaged, whereas in the classification model there is a voting, in which the most frequent categorical variable in all the trees, will indicate the predicted class. In **Figure 19** shows architecture of this model [60].

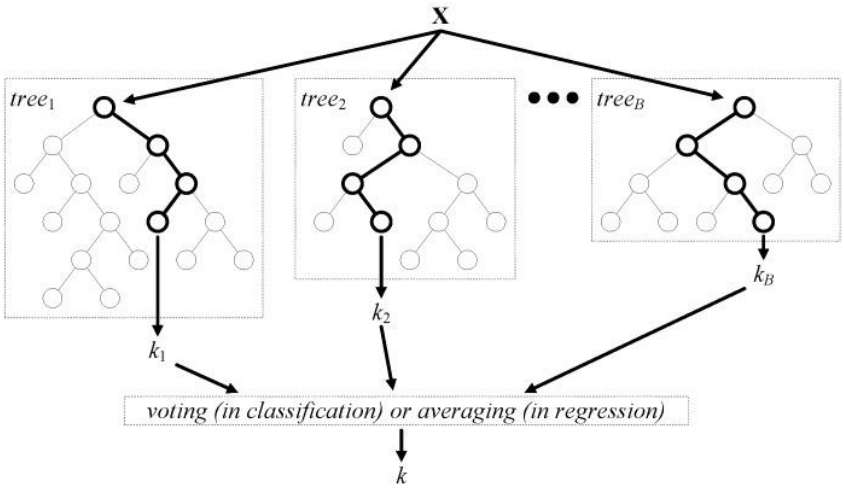


Figure 19: Architecture of a Random Forest Model [61]

- **Support Vector Machines (SVM):**

Support Vector Machines algorithm consists of a classification and regression tool that works very well analyzing very large datasets, since it maximizes the predictive accuracy without overfitting the training data. It works by mapping the data to a high-dimensional feature space, where the data will separate by their corresponding category, even if the data is not linearly separable. When the separator between these categories is found, the data are transformed in such a way that the separator could be drawn as a hyperplane. The prediction of the label for each value will happen based on the characteristics of the new data [62] [63] . In **Figure 20** the architecture of the model can be seen.

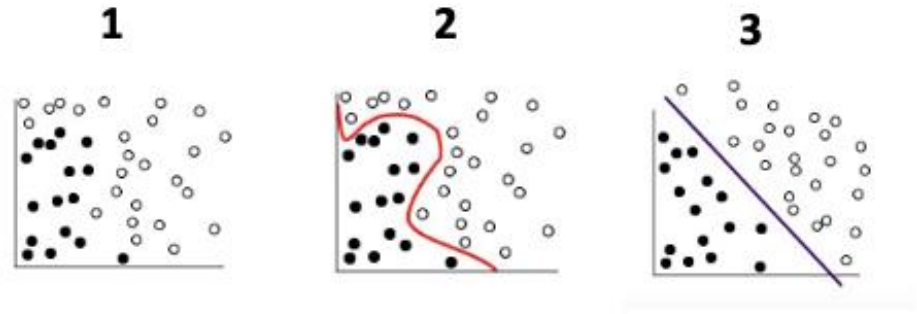


Figure 20: Architecture of a SVM Model. 1: original dataset; 2: the data can be separated into two categories as seen by the the red curve; 3: this boundary of the two categories is called a hyperplane, and is show in image purple line (adapted from [63]**)**

4.3.2. Unsupervised Learning

Previously we referred to about supervised learning. This method learns to correctly label a desired input through a set of inputs with the corresponding output given before. In unsupervised learning, the algorithm learns by using a created model that can extract the similarities in unlabeled data, where the

number of classes is unknown. In that way the model's main goal is to find meaningful and, most of the times, hidden patterns in the data and consequently organize it into different groups or classes, to be used for the prediction of inputs, decision making, among others. Essentially, unsupervised learning will differ from supervised learning since, although it receives inputs, the data the algorithm works with is unlabeled and the number of classes is not known, which will lead to a simpler method, but less accurate results [68][1].

Essentially, unsupervised learning comprises three main goals, clustering, association, and dimensionality reduction. Some of the most popular techniques used for unsupervised learning is PCA for dimensionality reduction. Their respective definitions follow below:

- **Principal Component Algorithm (PCA):**

As mentioned before, the principal component algorithm technique is used primarily for dimensionality reduction of the dataset. This technique can be very advantageous in cases where the dataset is very long and complex, by reducing its dimensions and consequently reducing the computational cost, minimizing information loss. PCA reduces the number of features or independent variables in the dataset. Through a linear transformation, this method converts correlated variables into linearly uncorrelated ones, named the principal components, where the components are the direction that maximizes the variance of the dataset, and each principal component is uncorrelated to the other. The direction that each principal component follows is always orthogonal to the prior components with the most variance [69][70].

4.3.3.Ensemble Learning:

The goal of machine learning is to find a model that will predict a desired outcome. In ensemble learning this technique combines several individual models and their corresponding hypotheses in order to produce one optimal predictive model. The main objectives are to decrease bias, variance or improve predictions. Ensemble learning can be divided into two groups:

- Sequential ensemble approaches: here the models are constructed sequentially since they are dependent from each other. An example of this is the AdaBoost technique.
- Parallel ensemble approaches: here the models are constructed in parallel since the models are independent of each other.

Two important concepts in ensemble learning are boosting and bagging. The first is a repetitive technique that adjusts the observation's weight based on the last classification. Bagging or Bootstrap Aggregation combines bootstrapping and aggregation to form one ensemble model, that will apply homogenous models on sample populations by taking the mean of all predictions.

In this work we will be working with the RUSBoosted Tree algorithm, since it specializes in skewed datasets. This classifier will apply random undersampling (RUS) from the majority class, decreasing its number of observations. [71] [72]

4.4. Deep Learning

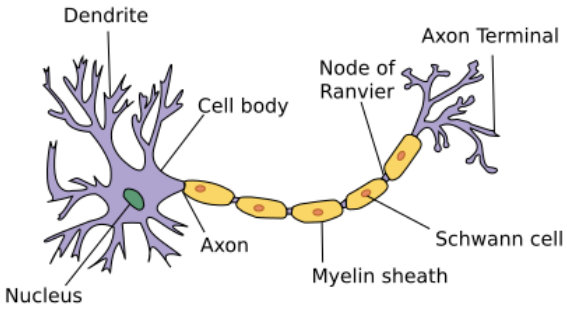
Deep learning is a branch of ML, that is considered as a more robust machine learning technique, with stronger computational power and a better competency in dealing with larger data bases. DL main

strategy is based on learning by example, just like us humans, by a hierarchical learning process with abstract levels.

Although the official DL term only came into perspective in 2006 by Hinton et al. [73] based on the concept of artificial neural network (ANN), it was already used in the form of neural networks in the late 1980s. Nowadays, DL is used to solve various complex problems, like facial or voice recognition.[74], instead of more traditional machine learning algorithms since the former have proved to have better capacity to deal with data comprehension and manipulation, in a more autonomous manner, as the volume of data increases. Even though DL takes more time to train a model due to many parameters, its testing time is shorter than to other ML techniques.

So how does DL work? Deep learning started with ANN, so it is clear that most deep learning methods use neural network architectures to learn. These neural networks were inspired by biological nervous systems, since they are composed by multiple processing layers of neurons, a simple processor that generates series of real-valued activations for the target outcome. The **Figure 21 (b)** shows the representation of the mathematical model of an artificial neuron.

a)



b)

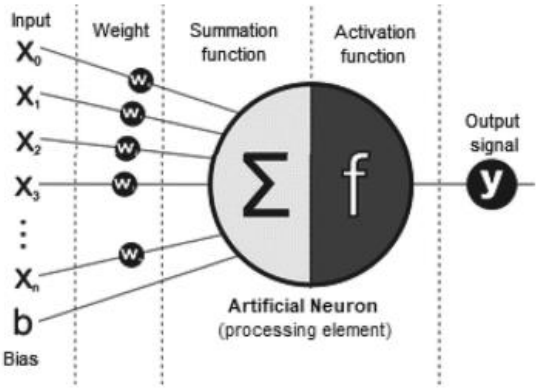


Figure 21: a) Human neuron structure; b) Schematic representation of the mathematical model of an artificial neuron (processing g element), highlighting input (X_i), weight (*w*), bias (*b*), summation function (Σ), activation (*f*) and output signal (*y*)

The scheme shown in **Figure 21 (b)** was proposed by McCulloch-Pitts and its mechanism can be explained using an analogy with the biological neuron, represented in **Figure 21 (a)** that inspired the first. A neuron receives signals from other neurons, that are connect between each other. That information is then

processed through the cell body and the processed signal is passed through the axon. However, the upon receiving the signal, the neuron can either “fire” or “not fire”, depending on the input signals. In an artificial neuron, the computational version of the neuron occurs. Here, the neuron receives several binary inputs (x1,x2,…), that are summed (weighted sum) in E (summation function), also adding a bias term. This value is then processed in an activation function, f, and transmitted in a single binary output, to the following neuron.

This output can either be 0 or 1, and it is determined by verifying if the value of the sum of the weights of each input is bigger or smaller than the value of the threshold, previously defined. If the value is the same or smaller than the threshold the output is 0, if the sum is bigger than the output is 1. [75]

The typical ANN is composed by several layers of neurons, specifically by one input layer, one output layer, that are all connected through several hidden layers, allowing for the decisions taken in one layer to be transmitted to the following layers, as can be seen in **Figure 22.** In the input layer the decision is only taking in consideration the summed weight of the inputs, but as the layers progress all the decisions made in previous layers will be taken in consideration, allowing for a more robust and complex process. In that way DL’s performance improves with the increase of data (**Figure 23**)[74] .

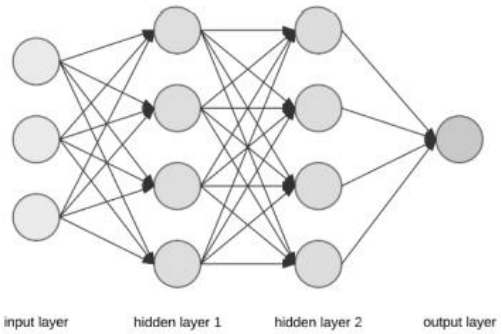


Figure 22: Architecture of an ANN model. Each circle represents a neuron, that outputs a value. The values together form a vector that represents the feature extracted from the input in this layer. The arrows represent the connection between the neurons and the transmission of the data [76]

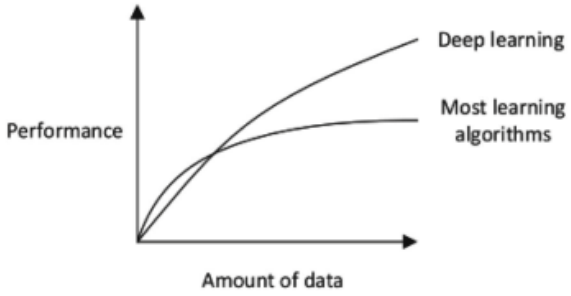


Figure 23: Comparison between deep learning (DL) and machine learning (ML) algorithms, where DL modeling from large amounts of data can increase the performance [74]

The field of deep learning is constituted by several different types of algorithms, just like in machine learning. Some of the algorithms are explained below:

The work of this thesis will focus on Recurrent Neural Networks (RNN) and its main variant, the Long Short-Term memory algorithm. These techniques are explained below:

- **Recurrent Neural Networks (RNN):**

Is an artificial neural network used for time series prediction, with a 'memory'-like property [77]. A time series prediction is the task of forecasting future values or patterns in a sequence of data points that are ordered in time. Time series prediction methods typically involve analyzing the historical data, identifying patterns or relationships, and building mathematical models that capture the underlying dynamics of the series.

RNN consists of multiple layers: input layer, hidden layer(s) and an output layer. Inside the hidden layer, the RNN contains recurrent units that allows the algorithm to process sequence data, by recurrently passing a hidden state from a previous timestep [1]and combining it with an input of the current one. So this means that the first layer has the weight derived from the input layer, and every layer after that will receive weight from the previous layer.[78] In other words, the present layers' output is completely dependent on the outputs of the previous layers. However, RNN has not shown very successful results with long term memory. That is why LSTM are considered a variant of RNN, since it fixes the problem of long-term memory with a forget gate. The architecture of a typical RNN can be seen in **Figure 24.**

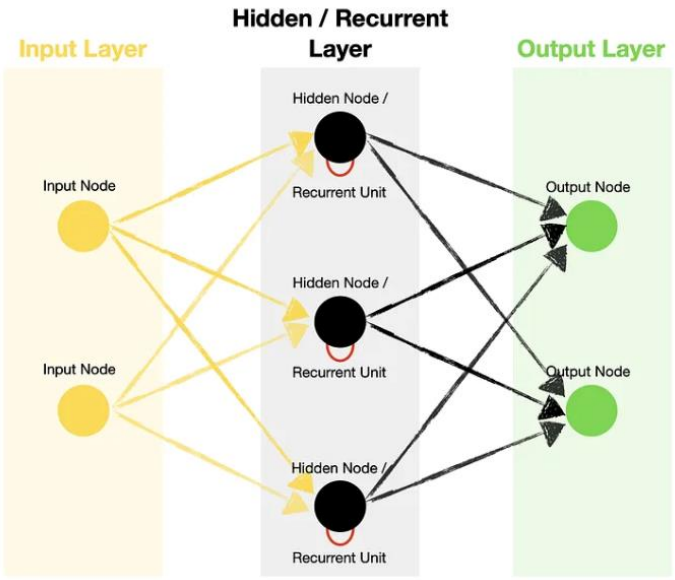


Figure 24: Architecture of a Recurrent Neural Network model [79]

[1] Timestep: single processing of the inputs through the recurrent unit. The number of timesteps is equal to the length of the sequence.

- **Long Short-Term memory (LSTM):**

This is a special recurrent neural network (RNN) algorithm that contains special units called memory blocks in the recurrent hidden layer, not present in a classic RNN architecture, as shown in **Figure 25.** These memory blocks contain three gate types, that can be compared to a filter, and are explored below [81]:

- **Forget Gate:** controls which information is relevant and which information must be forgotten from the cell state in the previous timestep (c_{t-1}).
- **Input or "Memory" Gate:** controls which new input data must be saved into the cell state, c_t. Here the results of the input gate get multiplied by the cell state, with important information chosen by the input gate.
- **Output Gate:** controls which information will be included in the output.

These gates will be the ones that'll help solve the long-term memory issue, by controlling the cell state by adding or removing information. As seen in **Figure 25** we have:

- **Hidden state and new inputs:** the hidden state from a previous timestep (h_{t-1}) combines with the input at a current timestep (x_t) before transmitting copies of the timestep through various gates.[98]
- **Update cell state:** the cell state from the previous timestep (c_{t-1}) gets multiplied by the results of the forget gate .Then new information is added from [input gate x cell state candidate] to get the latest cell state (c_t). [98]

As can be seen in **Figure 25** presented below, LSTM architecture can normally be seen over the t(time)-dimension. The process consists of three steps:

Step 1: the first step in the process happens at the forget gate, where the decisions will be processed by a sigmoid function (σ) with ranges between 0 and 1. Here, c_{t-1} will be multiplied element-wise by a vector, f_t , through the sigmoid function, which will generate a vector with values comprised between 0 and 1. If the value of the output is equal to 0 the information is rejected and if it is equal to 1 the information is considered relevant and is kept and transmitted in the loop. [80]

Step 2: here the new memory network is a hyperbolic tangent (tanh) activated neural network, which means that it will output values to the range [-1,1], so our new vector will have values within that range. Later, the sigmoid function will act as a filter, and transform the data into a vector of values in [0,1], through element-wise multiplication. Then the result is summed up with the previous cell state, resulting in c_t.[98]

Step 3: To stop an overload of information, the output gate will act as a filter. It will be applied to a version of the cell state, so it does not modify the cell state. Here this cell goes through a tanh function so the values are between -1 and 1, and then through the activation function, a sigmoid function, to filter the values in [0,1], through pointwise multiplication, and the new hidden

state is outputted. [81] The latest cell state and the hidden state return to the recurrent unit where the process repeats at timestep t+1, until the loop reaches the end of the sequence.[98]

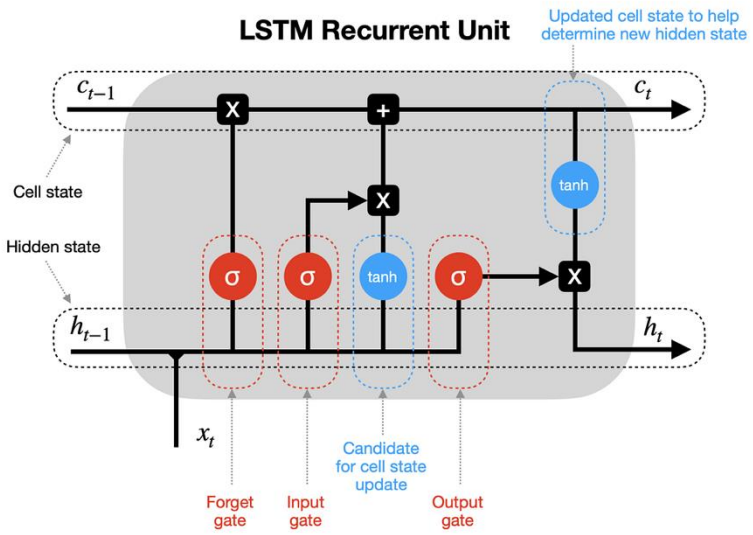


Figure 25: Architecture of an LSTM network. h_{t-1} – hidden state at previous timestep t-1 (short-term memory), where the red circle represents the sigmoid activation function, the blue circle represents the tanh activation function, the black dots represent the states, the red dots represent the gates, the blue dots represent the updates, x- vector pointwise multiplication, + - vector pointwise addition, c_{t-1} – cell state at previous timestep t-1 (long-term memory), x_t – input vector at current timestep t, h_t – hidden state at current timestep t, c_t – cell state at current timestep t, [81], [82] ,[98]

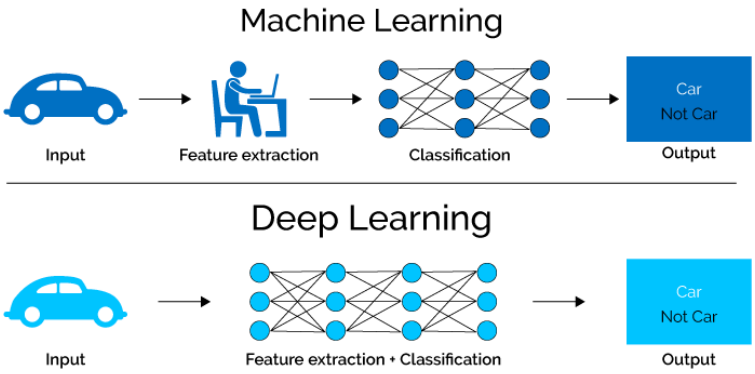


Figure 26: Side by side comparison of the Machine Learning and Deep Learning Process

A comparison scheme of ML and DL, can be seen in **Figure 26.** In short, we can say that machine learning is an artificial intelligence tool that will learn automatically with a need of human interference for learning and training specifically, particularly with feature extraction. Deep learning is a field of machine learning that will learn through a process like the human brain, via artificial neural networks, without the need of constant supervising, especially with feature extraction. The first trains small amounts of data with simple and quicker processes and has a lower accuracy, whereas the second requires big datasets thus taking

more time to train, but with a higher accuracy. DL also learns with its own environment, and can achieve better results with more complex data, in comparison with ML that is in constant need for human intervention to improve.

4.4.1.Deep Learning in biological data

In life sciences research scientists are flooded with large and complex datasets, which can be very laborious for traditional ML techniques to find patterns. The data can be split into three different groups based on its origin: images (e.g.: electromyogram, electrocardiogram), signals (e.g.: radiographs, MRI images) and sequences (e.g.: DNA/RNA sequences), that can produce very heterogenous and dynamic data. Dealing with this type of data with traditional ML techniques can be very challenging, therefore deep learning has been very successful with pattern recognition problems within biological problems.

According to Mufti Mahmud et. al.[77], some of the most used techniques inside DL for biological data, include convolutional neural network (CNN) (with the highest percentage of articles representing the technique), Fully Connected Network (FNN), Deep Autoencoder (DA[E]) and Recurrent Neural Network (RNN) including Long Short-Term Memory or LSTM. Based on the same article, in 2019 CNN was the most used algorithm around all data types, with DA as the second most used one and RNN showing an increased percentage of usage in applications.

4.5.Imbalanced Data

Previously it was mentioned that the objective of this work is the prediction of preterm labor using the data from the TPEHG dataset. However, the use of this dataset comes with some difficulties, in this case an imbalanced dataset, where the number of term recordings is larger than preterm ones. Such problem was emphasised in **Figure 10**.

Imbalanced data refers to an unequal distribution of classes within a dataset. **Figure 27** ilustrates the problem of imbalanced data. In our study it is obvious that most of the births in research correspond to term labors, while only a few are premature. This will generate an unbalance between the data points belonging to the term class and the ones in the preterm class, with our majority class being term labor, leaving the preterm labor class as the minority class.

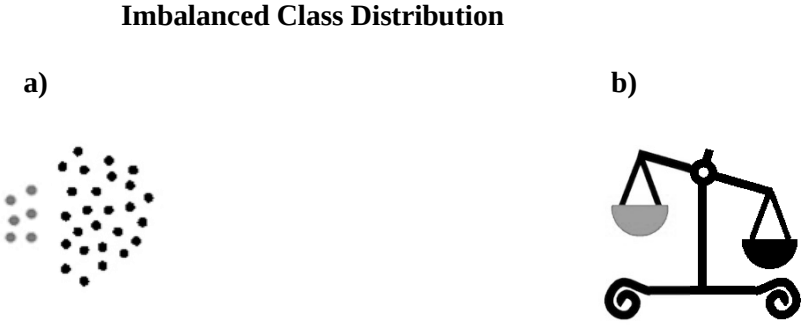


Figure 27: a) Imbalanced Class Distribution. The grey circles correspond to the minority class while the black circles correspond to the majority class. b) Unbalanced scale. Is an analogy that shows that the data represented with the black color has more weight than the one represented with the grey color.

When datasets are not balanced, classifiers like the ones referred to above, will be more sensitive to the majority class rather than to the minority class, causing a biased classification towards the majority class. This can be a huge problem since most of the time the dataset is composed of mainly "normal" examples with minority class being often the most relevant and useful one in the study. [83] On that account, different methods to overcome the imbalanced data problem have been employed, ranging from resampling the dataset to generating synthetic samples, with the two most popular ones being the SMOTE and ADASYN methods. [84] Two approaches can be defined: undersampling, where the majority class is sampled, and in our case specifically reducing the number of the majority class data points so it matches the number of minority class recordings and oversampling, where datapoints from the minority class are replicated to be equal to the majority class, a more accurate method than the former. [22]

In this work, the synthetic approaches are going to be explored, since they are the most used in premature birth literature, as seen in **Table 1.**

4.5.1. Synthetic Minority Over-sampling Technique:

Synthetic Minority Over-sampling Technique (SMOTE) is the simplest synthetic oversampling method, that treats all minority class samples equally. Here the minority class is over-sampled by creating "synthetic" examples. The oversampling is done by first drawing seed samples randomly (random numbers to ensure reproducibility) from all minority class samples, and after by calculating the k-nearest neighbors in the minority class for each seed sample and then introducing new synthetic samples along the line segments between the seed sample and its nearest minority neighbors, by joining them. [85]

The SMOTE algorithm currently runs with five nearest neighbors, but the number used is recalculated based on the percentage of the oversampling needed, so if the oversampling is at 200% only 2 of the neighbors will be used.

Synthetic samples are generated as follows: first take the difference between the feature vector (sample) under consideration and its nearest neighbor, then multiply this difference by a random number between 0 and 1, and finally add it to the previously mentioned feature vector. This leads to selecting a random point along the line segment between two specific features, originating the decision region of the minority class to become more general. [23]

The authors of SMOTE state that combining the oversampling of the minority class (using SMOTE) with the undersampling of the majority class, will lead to higher predictive accuracy results [23].

4.6. Feature Extraction

Feature extraction for machine learning is a necessary step that yields better results than working with the raw data. It can be defined as the process of turning raw data into numerical features that can be processed whilst keeping the information in the original dataset. This procedure can be done manually by first identifying and describing the most relevant features and then extracting the most relevant features using a defined manual method and first, or automatically with an algorithm specialized in automatically and independently extracting features from signals or images, a faster process than the first. For signal data, feature extraction can be challenging due to information redundancy and high data rate. [87] . In **Figure 28** a scheme of the feature extraction process can be seen.

Figure 28: Feature Extraction process in raw signals and time series data for a ML classifier (taken from [87])

4.7. Dataset Features

As seen in **Table 1,** in the literature, the predominant features used in the preterm prediction are time-signal features. Idowu et al. used sample entropy, root mean square, frequency, median and peak frequency, among others as features, Ryu et al. also showed promising results while using sample entropy [6], [7]. Jager et al. used frequency and peak amplitude of the power spectra in the contraction intervals as features [12]. The TPEHG database has several features, from the characteristics of the patients like age, parity, existence of diabetes, smoker status and others, to characteristics of the signal [12].

Overall algorithms perform well with a higher volume of features since they will have more data to learn from, so it would be beneficial to work with more features in this work.

As mentioned before, in this thesis the goal was to use only the contractions instead of the whole signal, which still needs to be done. The problem is that contractions are non-stationary time series, and each contraction can have a different number of samples. For that reason, we chose the signals power spectra density as it represents the EHG signal more accurately independently of the number of samples of the contraction. [34]

Therefore, we will focus on the spectral analysis of all the contractions for each pregnant woman. Time-frequency transformations, such as the short-time Fourier transform (STFT) or the Fast Fourier Transform (FFT) can be used as signal representations for training data in machine learning and deep learning models.[88]

Signal processing is often used for feature extraction and classification in medical disease diagnosis [88], where spectral analysis plays a very important role in signal processing for distinguishing and tracking signals of interest. The goal of spectral analysis is to decompose the data into a sum of weighted sinusoids, that allows one to assess the frequency content of the signal. [89]

Power spectral density (PSD) measures a signal's power content versus frequency and its estimation results are typically used to characterize signals. In other words, PSD represents the power of the input signal over a range of frequencies. It has been proven before that the PSD can be used to analyses EHG signals to evaluate the changes of the power spectra in premature labors[34]. Its methods can be divided into parametric and non-parametric groups. Parametric methods are model-based methods, that match the signal with a model. One disadvantage is that when the selected model is wrong, PSD estimation can contain invalid frequency peaks. The non-parametric spectral analysis estimates the spectral density of a random signal without pre-parameter modeling, being very robust. However, since it needs data windowing, it can lead to a distortion of the PSD. [34] [90]

A spectral estimator is expected to have good statistical properties such as consistency, high resolution, and small variance. For one spectral estimation method, there exists a trade-off between high resolution and small variance. Our created database includes several PSD methods, in this work specifically

we are going to use the Welch method as the PSD estimation method. The Welch method, also called the periodogram method is an average of periodograms across time, consisting of a non-parametric approach. This method will estimate the power spectra by dividing the time signal into successive blocks, forming a periodogram for each block and finally doing the correspondent average. In **Figure 30** a scheme for this method can be seen, step by step [91]. According to the literature, it is common to process signals like the EMGs in blocks, due to their slow varying nature in time [92]. In **Figure 29** the power spectra of two different contractions using the welch method can be seen, showing that the contractions will always be the same length.

By looking at the signals from observation 200 to observation 513 the value of the welch is approximately 0. Based on these values, the correspondent features would be useless to the classification algorithm, so they were not added to our analysis. Therefore, in our final dataset the power spectra for each contraction, present in each signal per individual were used as features, until observation 200, ending up with a dataset with a total of 4622 observations with 200 features. Additionally, a classification column was added representing the labor category that each contraction belonged to based on the Time of Gestation, with a value of 0 when it is term labor and 1 if it is preterm labor. This dataset will be later added to the classification algorithm. In **Figure 31**, a sample of the organization of the dataset is shown.

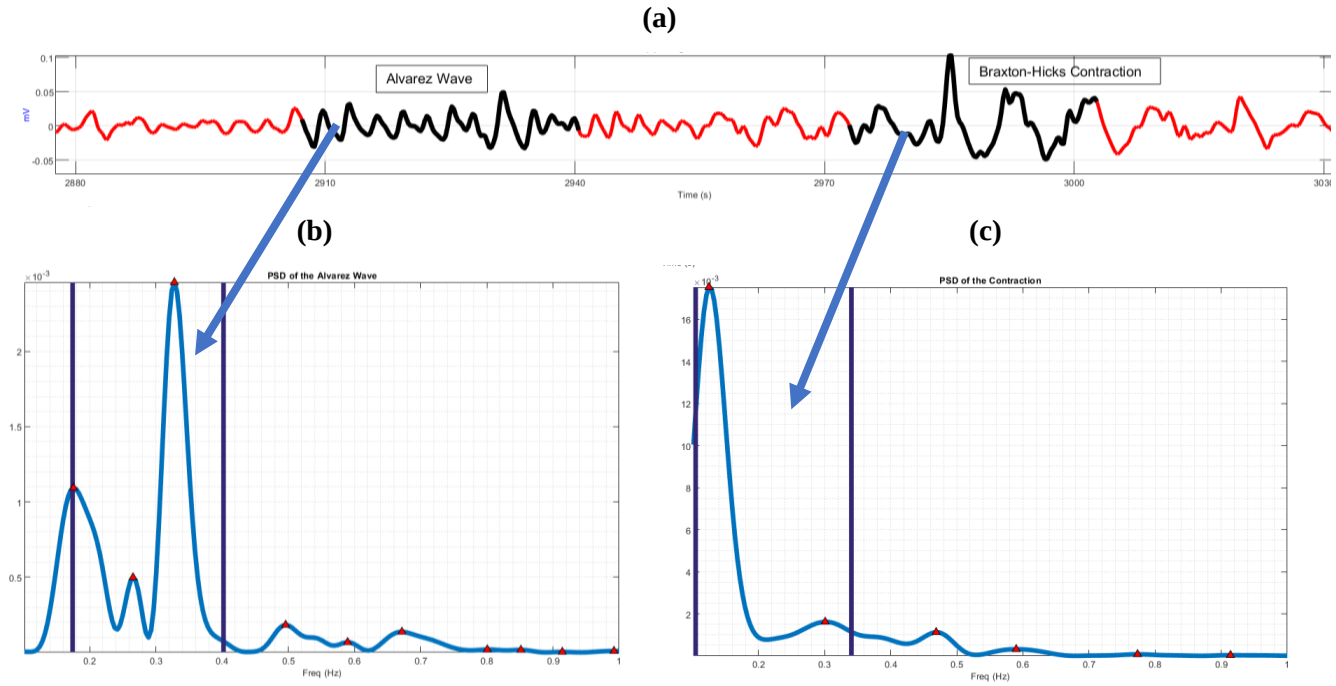


Figure 29: Power Spectrum of a contraction using the Welch Method. By using this method all the contractions present the same length. (a) EHG signal with two different contraction, Alvarez Wave and Braxton-Hicks Contraction; (b) PSD of the Alvarez Wave; (c) PSD of the Braxton-Hicks Contraction.

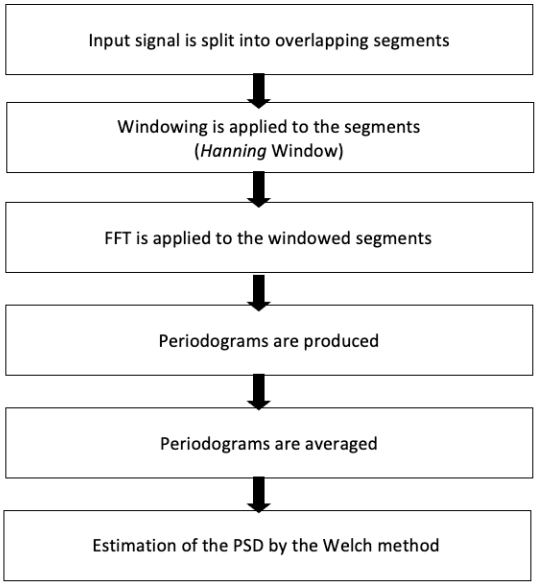


Figure 30: Welch Method Scheme [91]

4622x203 table

	1 Contraction number ID	2 Welch1	3 Welch2	4 Welch3
1	'tpehg546_1of17'	5.4583e-...	1.0895e-...	1.0822e-...
2	'tpehg546_2of17'	4.9279e-...	9.6483e-...	9.0487e-...
3	'tpehg546_3of17'	3.1405e-...	6.2095e-...	6.0008e-...
4	'tpehg546_4of17'	1.3415e-...	3.2577e-...	4.8024e-...
5	'tpehg546_5of17'	2.4368e-...	5.2425e-...	6.3545e-...
6	'tpehg546_6of17'	6.4565e-...	1.9646e-...	4.0627e-...
7	'tpehg546_7of17'	1.7952e-...	3.5828e-...	3.5492e-...
8	'tpehg546_8of17'	6.4221e-...	3.4048e-...	1.2635e-...
9	'tpehg546_9of17'	3.3298e-...	7.4416e-...	9.7004e-...
10	'tpehg546_10of17'	4.6204e-...	9.0165e-...	8.3507e-...
11	'tpehg546_11of17'	2.6302e-...	5.2184e-...	5.0732e-...
12	'tpehg546_12of17'	1.0535e-...	2.1208e-...	2.1531e-...

(...)

200 Welch199	201 Welch200	202 Time of Gestation	203 Labour Classification
2.3779e-06	3.1286e-06	31	1
3.4105e-05	4.3304e-05	31	1
7.1840e-05	6.1195e-05	31	1
1.1640e-05	9.8008e-06	31	1
2.7306e-05	2.6235e-05	31	1
3.7446e-05	4.0883e-05	31	1
1.7259e-05	1.4580e-05	31	1
2.2428e-05	2.7226e-05	31	1
8.3871e-06	6.2089e-06	31	1
2.0126e-04	2.0445e-04	31	1
6.2917e-05	7.1180e-05	31	1
1.7687e-05	1.6370e-05	31	1

Figure 31: Case Study Dataset for Machine Learning

4.8. Algorithm Evaluation

Evaluation of different classifiers is challenging since we can only know if a model is good until it is used. Therefore, its performance must be estimated using available data when we have the target or the outcome. This evaluation consists in a lot more than testing a learning algorithm, it assembles different types of tests with various elements. It includes testing different data preparation schemes, different learning algorithms, and different hyperparameters for well-performing learning algorithms. In the end, the model with the best construction procedure (data preparation, learning algorithm, and hyperparameters) and correspondent best score (with our chosen metric) can be selected and used. [93]

For very large datasets or datasets where the data is well represented and balanced, with all the classes in the problem having the same proportion, a simple procedure of splitting the dataset into two equal parts, one for training the model and the other for testing works. Although splitting the data 50/50 will be ideal, 70/30 or 80/20 splits for train and test sets, respectively, are more frequent.

For unbalanced datasets, k-fold cross-validation is commonly used for validation. This approach consists in splitting the training set into k folds. The first k-1 folds are used to train the model while the holdout kth fold is used as the validation set. This step repeats a k number of times, where each of the folds can be used as the holdout test set and a total of k models are fit and evaluated, and the model's performance is calculated as the mean of these runs. In literature, a value of k=10 folds has shown the most success in a wide range of datasets and different models. However, more often than expected, when dealing with highly unbalanced datasets, splitting the data into two groups will be impossible, since it might happen that one of the groups has no data for the group of interest, in this case premature births. To combat this issue, a stratified k-fold cross-validation procedure can be used. This is a common procedure in cases of imbalanced datasets, since it ensures that the proportion of elements in each class in the original distribution is preserved in all folds. In this process the dataset is randomly split while maintaining the same class distribution in each subset, to protect the imbalanced class distribution in each fold and protecting the distribution in the complete training dataset. [93]

Recently, Mohammadi Far, S. has shown success with 10-fold stratified cross validation in the TPEHG dataset, guaranteeing the existence of both classes in all subsets. [21], [94]

4.9.Feature Selection

While continuously trying to construct the best model, feature selection algorithms come into action. In most classification problems, especially when dealing with many features, it is easy to assume that some of these predictor's variables can be redundant to the classification problem, or just damage the prediction performance. This method consists in reducing the dimensionality of data by selecting a subset of features that improve the classification performance, in a much quicker procedure. There are three different categories in the feature selection algorithm**s** [94]**:**

- **Filter Type Feature Selection:** this method measures the predictors importance based on the characteristics of the features.
- **Wrapper Type Feature Selection:** this method trains several models using a subset of different features and then adds or removes a feature using a selection criterion, then choosing the best performing model.
- **Embedded Type Feature Selection:** this method will rank feature importance while the model is training, as a part of the model learning process. Thus, the features selected will be the ones that work well in the learning process.

The methods used in this work consist of a filter type method for feature ranking, including the Minimum Redundancy Maximum Relevance (MRMR) Algorithm.

- **Minimum Redundancy Maximum Relevance (MRMR) Algorithm**: it works by finding an optimal group of features mutually and maximally unrelated to each other to effectively represent the response variable. This procedure will minimize the redundancy of a feature set to the response variable, by quantifying the redundancy and relevance of the pairwise mutual information of features and mutual information of a feature and the response. [94]

  In *Matlab*® the correspondent function is the *fscmrmr*, and it ranks features in 6 steps:
  - The feature with the largest relevance is selected and added to an empty subset, *S*.

- Searches for features with nonzero relevance and with zero redundancy in the complement of subset, S^c.
- If the complement set does not have this type of features, it goes directly to step 4.
- Otherwise, the feature with the highest relevance value is selected and appended to the subset, *S*.
- Repeat step 2 until the redundancy value for all features in the complement set is different from zero.
- In S^c, selects the feature with the largest mutual information quotient (MIQ^2) value with relevance and redundance different from zero, and appends it to the set *S*.
- Step 4 will be repeated until the relevance is equal to zero for all features in S^c.

The features with zero relevance are randomly added to *S*. [94]

4.10. Hyperparameters

As mentioned before, to construct a good classification model, we must consider the best combination of hyperparameters. Hyperparameters are numeric or Boolean values that the user can adjust before training the model, which will help to improve the training time, performance and prediction of the classifier, and will be used to control the learning process. Thus, as the name states they are "hyper" and therefore very important since they will make training very effective in terms of both time and fit, avoiding overfitting and underfitting. Some examples of hyperparameters are the train-test split ratio, batch size, branches in decision tree or the number of clusters in the clustering algorithm.

Although you can find the most relevant hyperparameters for the classifier manually, currently one can find a few optimization tools capable of identifying good hyperparameters, in a process called hyperparameter tuning/optimization (HPO). In this work, the Bayesian Optimization tool was used, which consists of building a probability model of the unknown scalar objective function, *f(x)* using a probabilistic surrogate model, typically a Gaussian process, to minimize *f(x)* for *x* in a bounded domain. The components in *x* can be reals, integers or categorical. The key elements in this process are:

- A Gaussian process model of *f(x)*.
- A Bayesian update procedure that modifies the Gaussian process model at each new evaluation of *f(x)*.
- An acquisition function *a(x)*, based on the Gaussian process model of *f*, that determines the next point of *x* for evaluation.

The probabilistic surrogate model will provide a representation of the objective function while updating iteratively as new data points are observed. The surrogate function is the Bayesian approximation of the objective function and the acquisiton function selects the next sample of hyperparameters from the search space. The way this process works is:[96], [97]

---

2 $\max_{x \in S^c} \mathrm{MIQ}_x = \max_{x \in S^c} \frac{I(x,y)}{\frac{1}{|S|}\sum_{z \in S} I(x,z)}$ [94]

1. Select a sample by optimizing the acquisition function, finding the best hyperparameters that outperform the rest on the surrogate model
2. Evaluate the hyperparameters in the true objective function
3. Incorporate the new results on the surrogate model
4. Keep repeating the process until the end of the fixed number of iterations (30 is the default number) or fixed time (the default is no time limit)

The problem with finding the optimal hyperparameters is that it is very time consuming, since the evaluation consists in training the model and testing I, and then calculate the evaluation metric on a validation set. The traditional approach for HPO is the "one try" method, were the tunning tool finds the best hyperparameters in just one try. [97]

Bayesian optimization is the model chosen for this research since it can outperform the previous ones mentioned. Grid search and Random search, although better than manual tuning, are still lacking. The problem with these methods is that they don't take in consideration past results and for that reason a big portion of the time is spent evaluating hyperparameters that are unsatisfactory.

The six types of acquisition functions found in Bayesian Optimization are:

- "expected-improvement-per-second-plus" (default)
- "expected-improvement"
- "expected-improvement-plus"
- "expected-improvement-per-second"
- "lower-confidence-bound"
- "probability-of-improvement"

The "expected-improvement-per-second-plus" default option was the chosen as our optimization tool. These expected-improvement functions evaluate the total expected improvement in the objective function, using time-weighting (-per-second-) in its acquisition function and modifying their behavior when they estimate that they are overexploiting an area (-plus). [96]

4.11. Evaluation methodology

To evaluate the performance of our classifiers different statistical measures had to be used. These measures were chosen based on the most popular measures used in literature since the goal is to see if our classifier can achieve better results in comparison to the more traditional ones presented in literature, seen in **Table 1.**

For every classifier and with each combination, for the performance evaluation we will be using a confusion matrix, a two- dimensional matrix table used to rate the performance of a classifier based on testing data [5]. The confusion matrix is a table with 4 different combinations of predicted (described as positive and negative) and the true/actual (described as true and false) values, that can be seen in **Table 32:**

		Predicted Class	
		Negative (0) – Term Labour	**Positive (1) – Preterm Labour**
True Class	**Negative (0) – Term Labour**	True Negative (TN)	False Positive (FP)
	Positive (1) – Preterm Labour	False Negative (FN)	True Positive (TP)

Table 32: Confusion Matrix

In **Table 32** we see that the confusion matrix is constituted by:

- **True Negative Value (TN):** when the algorithm predictes negative, and it is true, corresponding to a correctly identified term labor.
- **True Positive Value (TP):** when the algorithm predicts positive, and it is true, corresponding to a correctly identified preterm labor.
- **False Positive (FP):** when the algorithm predicts positive and it is false, meaning that the algorithm predicted a preterm labor but it is actually a term labor.
- **False Negative (FP):** when the algorithm predicts negative and it is false, meaning that the algorithm predicted a term labor but it is actually a preterm labor.

It is very useful for measuring several performance metrics [5]. These include:

- **Accuracy (AC):** this is the most used metric. It consists in a percentage of how often the classifier is correct in naming term or preterm labor. It is given by the following equation [5]:

$$AC = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP (true positive value), TN (true negative value), FP (false positive value), FN (false negative value) can be extracted from the confusion matrix.

- **Recall (R):** percentage of how often the classifier gives correct predictions, so when it identifies the labor as preterm when it is preterm, giving an actual positive. Recall should be as high as possible. R is given by the following equation [5]:

$$R = \frac{TP}{TP + FN}$$

- **Precision (P):** it gives us a measure of how precise/accurate the model is, so between all the positive predicted values, how many are true positives. In this case, how many preterm labours are correctly identified as preterm. Precision should be as high as possible. It is defined by the following equation [5]:

$$P = \frac{TP}{TP + FP}$$

- **F1 Score:** it is a measure that combines the precision and the recall values, more precisely it corresponds to the harmonic mean of precision and recall. A higher value of F means a better classifier. It is a good metric to compare different models with low precision and high recall. It is defined by the following equation [5] :

$$F = 2 \times \frac{P \times R}{P + R}$$

- **False Negative Rate (FNR):** it measures the percentage of how often the classifier classifies the labor as term when it is preterm. It is defined by the following equation [5] :

$$FNR = \frac{FN}{TP + FN}$$

- **False Positive Rate (FPR):** it measures the percentage of how often the algorithm classifies the labor as preterm when it is term. It is defined by the following equation [5] :

$$FPR = \frac{FP}{TN + FP}$$

- **Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC):** these are tools that will also measure the performance of the classifier. The ROC is also a performance evaluation measurement for binary classification algorithms. It corresponds to a graphic plot representation of the variation in the recall (also referred to as TPR, true positive rate) and the FPR for all different thresholds. The AUC is an area measure that helps us compare the ROC Curves. It is defined as the probability of the classifier ranking a randomly chosen positive instance higher than a randomly chosen negative instance. So it will measure the performance of the classifier, with values between 0 and 1, the closer to 1 the better the classifier is, exhibiting an excellent discrimination power. It is important to know that the ROC curve does not depend on the class distribution, which is particularly

helpful in imbalanced datasets, where the classifiers tend to predict a negative outcome for the majority class [5].

The next chapter will present our case study.

5. Results and Discussion

In this section, we present the results of our study on the classification of preterm and term delivery using both Machine Learning and Deep Learning algorithms. We implemented a total of five algorithms, including four machine learning algorithms: Random Forest (RF), Random Undersampling (RUSBoosted Tree), Support Vector Machine with a Gaussian Kernel (SVM), and a Shallow Neural Network, as well as one Deep Learning Algorithm, a bilateral Long short-term memory (LSTM) network. We chose RF and SVM based in **Table 1**, and the Wordcloud shown in **Figure 1,** where both were the most predominant classifiers with positive results in preterm birth classification using EHG signals. Additionally, we selected RUSBoosted Tree as it is an ensemble algorithm that performs well with imbalanced datasets like our own. The Shallow Neural Network was chosen to provide insight into the deep neural network used in the DL step of the analysis. For the DL approach, we chose the bilateral LSTM network as it is well-suited for sequence models.

Our dataset comprises a total of 200 features belonging to the Power Spectral Density (PSD) of each contraction from each patient. This novel approach differs from the literature, where time-signal and spectral features, such as sample entropy and peak frequency, are typically used. We split the dataset using a 70% holdout technique, with 70% of our dataset as the training dataset and the remaining 30% as the test dataset. The split was chosen to ensure that both classes are represented in both the training and testing sets. After splitting, the training set consisted of 3236 observations, with 2829 term labor and 407 preterm labor observations. The testing set contained 1386 observations, with 1212 term labor and 174 preterm labor observations. **Figure 33** shows the number of observations per class in the test and training sets, demonstrating that both classes are represented despite the imbalanced dataset.

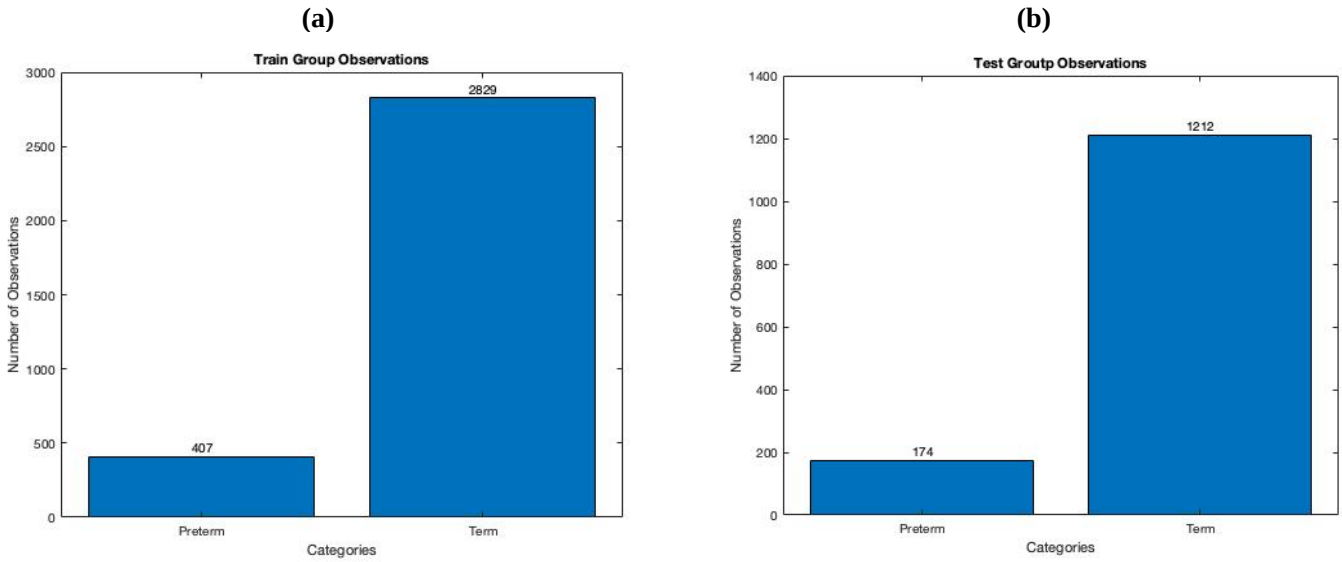


Figure 33: Number of observations for each class (Term or Preterm) in the initial training (a) and the test set (b)

The training set was later inserted in the *Classification Learner App*, to begin the training and validation with each classification model. For the ML approach, presented in **Chapter 4.3** , the trained dataset was trained and validated using a stratified k-fold cross validation method with k equal to 10, since

this exact method has been positively linked to the classification of preterm labor using a skewed dataset, more recently. A k-fold CV is preferable, especially when working with smaller datasets, to evaluate a model since it repeats the process k times. It is also known that this method can be more advantageous with imbalanced data, when comparing to k-fold cross validation, like it was mentioned before in **Chapter 4.8.** A visual representation of this step is seen in **Figure 34**:

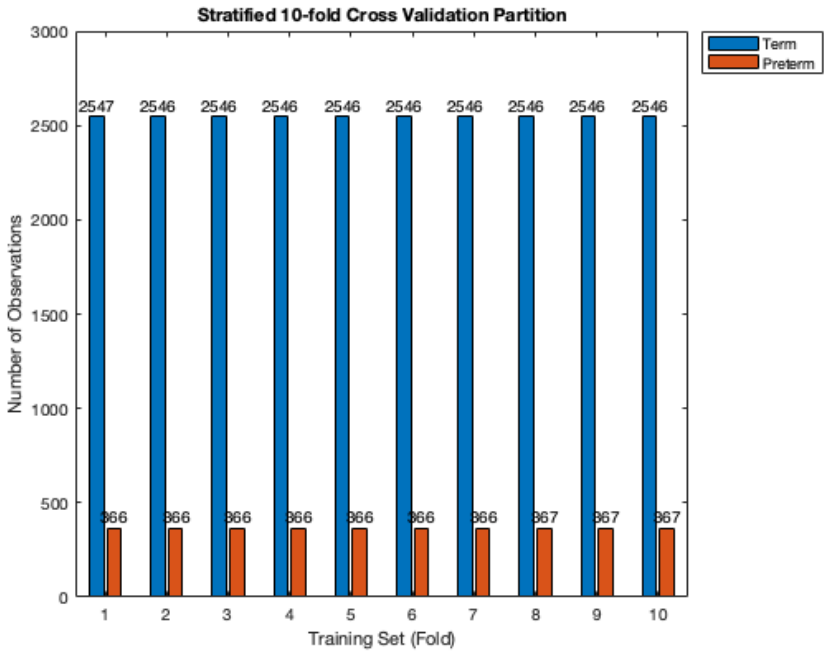


Figure 34: Number of Preterm and Term observations for each fold in the Stratified 10-fold Cross Validation Partition for the original dataset

Later when we achieved a good model performance, the trained model was evaluated using the test set and the different metrics, mentioned in Chapter 4.11, were computed. The workflow, present in **Figure 35** consists of:

1. Training dataset is used to train each model
2. Validation dataset is used to evaluate each model
3. The best trained model is tested and evaluated with the test dataset, after the feature selection and hyperparameter tuning process

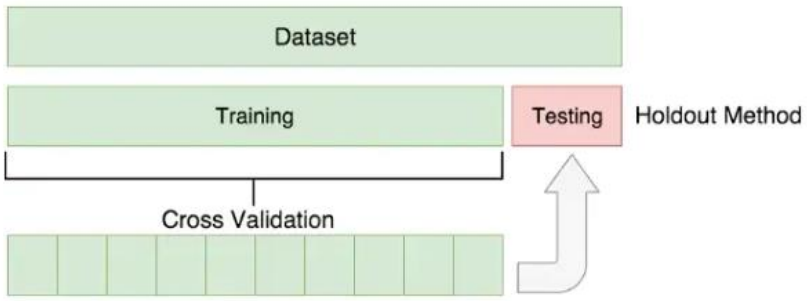


Figure 35: Scheme of the training-validation-testing plan

All the training and validation processes were performed through the *Classification Learner App* from *Mathworks*®.

The following sections will present the performance evaluation results and discussion of the ML and DL approaches. First, for the ML, we'll present the confusion matrix and AUROC plot for each classifier with the untouched data, second using the transformed datasets using SMOTE, third using the SMOTE datasets and an undersampling technique, fourth applying PCA, a dimensionality reduction technique to the dataset, and finally using a feature selection method with hyperparameter optimization. In the last step, we'll also perform the testing using the test dataset with 1386 observations, in the best tuned model. This last step was only done with the best performance classifiers since it would be pointless to test a model that does not predict well our results. It is important to use the test dataset without any modifications, so the machine learning model behavior is tested with observations that he's never seen before and are unbiased. For the DL approach the classifiers were trained with the original dataset and the dataset after applying SMOTE. The best classifiers were then tested with the same test set used in the ML part. In the end both methods, ML and DL, are compared to each other and to the best results found in literature.

5.1.Machine Learning Methods

5.1.1 Results for the study dataset

In this first section the dataset was used without any transformation, so 3236 observations were used for the training dataset, leaving 1386 for the test dataset. The data used in the stratified cross-validation step was split into 10 different groups, with the same class ratio throughout the 10 folds as in the original dataset, as seen in **Figure 34**.

The **Figure 36** represents the training results based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network, shown in **Table 4** to **Table 7**, respectively, and in the ROC curves present in the attachment.

Table 4: RF classifier confusion matrix for the training set of the original dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2814	15
	Preterm Labor	406	1

Table 5: RUSBoosted Tree classifier confusion matrix for the training set of the original dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1541	1288
	Preterm Labor	211	196

Table 6: SVM with the Gaussian kernel classifier confusion matrix for the training set of the original dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2829	0
	Preterm Labor	407	0

Table 7: NN classifier confusion matrix for the training set of the original dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2484	345
	Preterm Labor	338	69

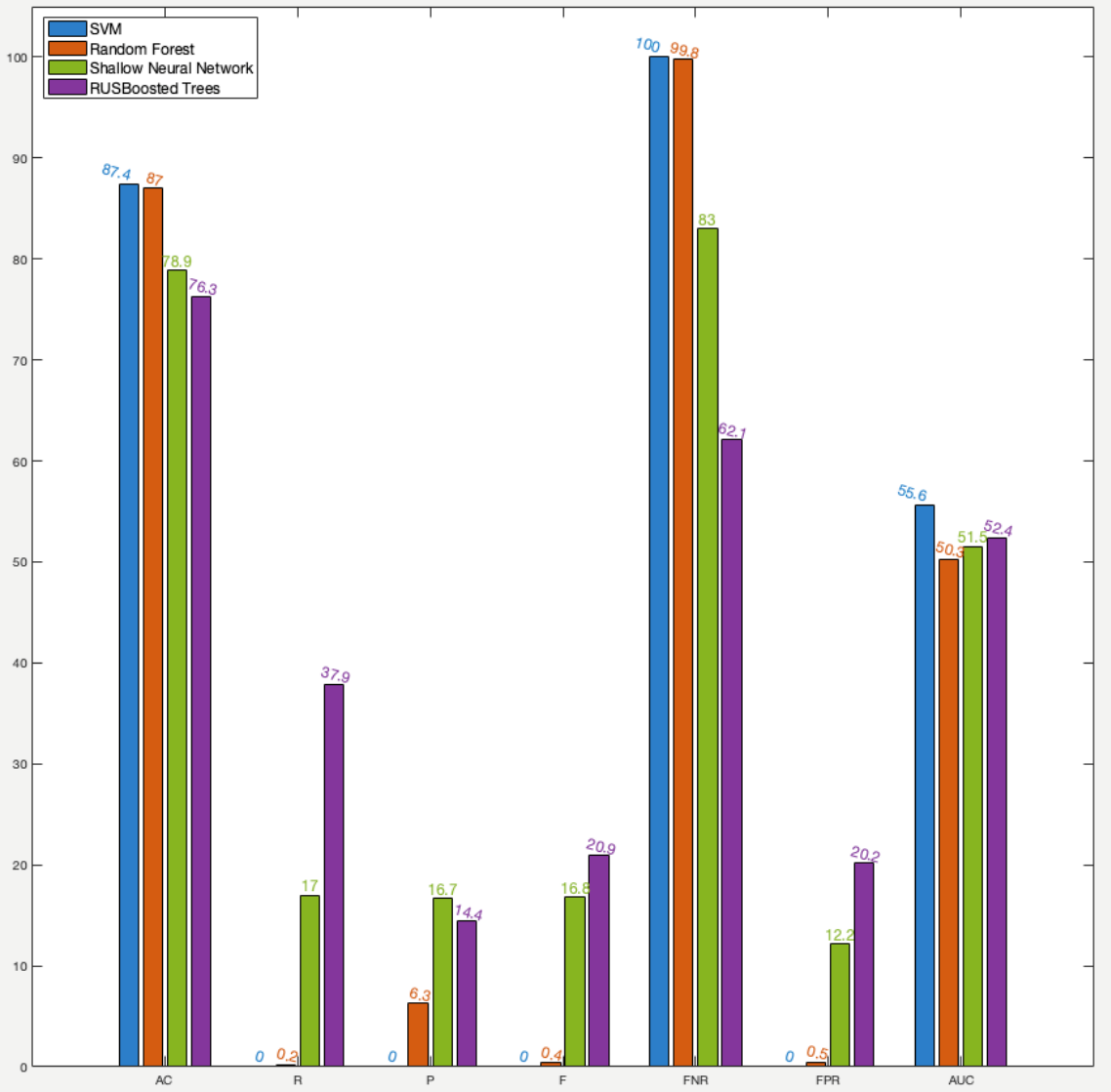


Figure 36:Machine learning performance results for the validation dataset (using only stratified cross-validation) for comparison

As seen in **Figure 36** amongst all classifiers the SVM with the Gaussian Kernel performed the best in terms of the accuracy rate, with both the SVM and the RF scoring very high results, of around 87%. The NN and RUSBoosted Trees also showed a high accuracy of approximately 80%. However, since we are dealing with an extremely unbalanced dataset, based on the confusion matrix of **Table 6** (where the model does not predict any preterm labor), the model simply learned to predict the majority class very well. For that reason, looking at the accuracy it is not the best way to measure the model's performances with this exact dataset. Based on the other measures of the classifiers we can see that the RUSBoosted Tree showed the best results with the highest value for Recall, Precision and F1 Score, and lowest value of FNR. This classifier also shows the highest value for the FPR, and even if the value is notvery high it means that it will classify a labor as preterm when in the dataset is term, which is notgood for a classifier. SVM and RF show a FNR of approximately 100%, meaning that all preterm births were classified as term.

Additionally, looking at the AUC values curves present in **Figure 36,** overall, all classifiers perfomed very poorly, with the SVM showing the best AUC value of 55,6%, indicating that the model is incapable of doing real predictions. Even though the RUSBoosted Tree is linked to a high performance with imbalanced datasets, its performance did not stand out, most likely due to the extremely imbalanced dataset.

5.1.2. Results for the study dataset using SMOTE for oversampling

Since the results presented before were very low due to the highly imbalanced dataset, the next step was to apply the oversampling techniques to the feature dataset. Considering this, the SMOTE method was applied to our contraction's dataset. The oversampling was only applied to the training dataset, after the partition since these yields more accurate results without an overly positive classification measure. It is important to note that the testing set will not suffer any changes from this oversampling method. This is also a new approach different from the usual in literature, where the partition occurs only after SMOTE is applied. However, as it was mentioned before recent studies have found that using oversampling techniques, like SMOTE on the test set might lead to overly optimistic results. [22]

In almost all literature using SMOTE as an oversampling technique for the TPEHG dataset, the method is always to oversample the minority class to equal the majority class, which leads to extremally good results. However, on the original paper on SMOTE [23], the authors suggested combining SMOTE with random undersampling of the majority class, admitting that this should favor even more the results, avoiding overfitting. For that reason, in this step, we are going to compare using the SMOTE with and without undersampling of the majority class.

This comparation approach has the objective of seeing if there is a significant difference in utilizing SMOTE with and without undersampling of the majority class and if it improves the performance of the classifiers.

a) Results for using only SMOTE

First SMOTE was applied to the training dataset, that originally had 3236 observations, with 2829 term labor contractions and 407 preterm labor contractions. After this step the dataset had a total of 5658 observations, with the same number of preterm contractions as term ones, as seen in **Figure 37.**

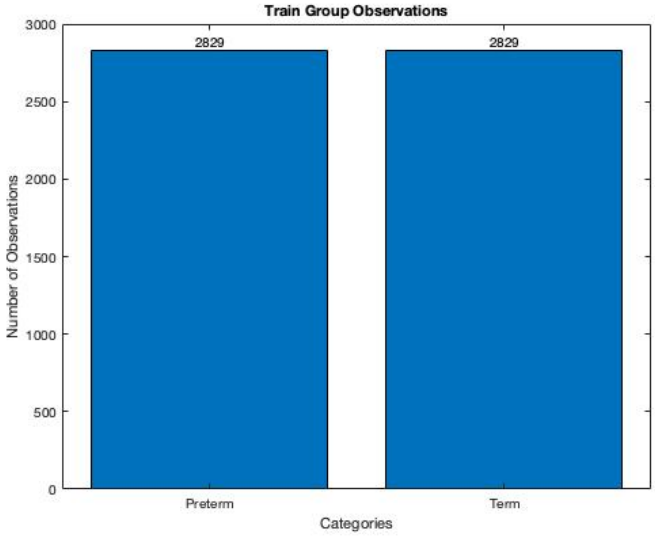


Figure 37: Number of observations for each class (Term or Preterm) in the training dataset after applying SMOTE

The previous classifiers were applied including the RF, the RUSBoosted Tree, the SVM with Gaussian kernel and Shallow Neural Network. For the validation set the same stratified 10-fold cross

validation as before was applied. Figure 37 displays the classification metrics for the training set, based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network, shown in **Table 8** to **Table 11,** respectively and in the ROC curves shown in the attachment.

Table 8: RF classifier confusion matrix for the SMOTE training dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2468	361
	Preterm Labor	287	2542

Table 9: RUSBoosted Tree Trees classifier confusion matrix for the SMOTE training dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1030	1799
	Preterm Labor	285	2544

Table 10: SVM with the Gaussian kernel classifier confusion matrix for the SMOTE training dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2440	389
	Preterm Labor	1180	1649

Table 11: NN classifier confusion matrix for the SMOTE training dataset

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2336	493
	Preterm Labor	163	2666

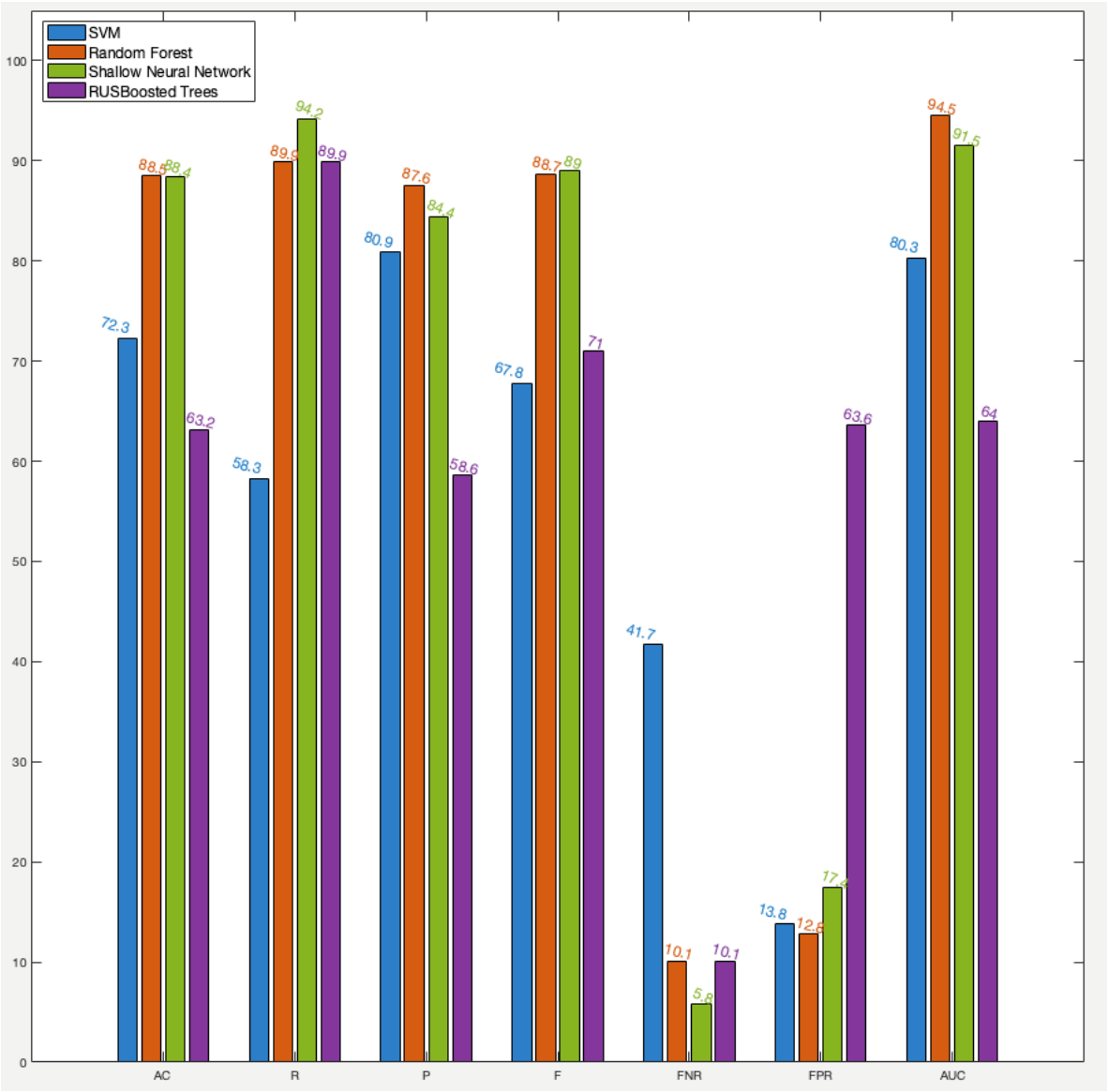


Figure 38: Machine learning performance results for the validation dataset (SMOTE) for comparison

As seen in **Figure 38** and in the confusion matrix of all classifiers, the results improved significantly from applying an oversampling technique. From the confusion matrix we can see that overall, all classifiers were able to predict almost all preterm labor contractions, expect for the SVM that performed slightly worse compared to the rest. Both the RF and NN performed extremely well compared to the performance in the initial dataset presented in **Chapter 5.1.1**, with an accuracy of 88.5% and 88.4%, respectively, and a high value for the F1-score of 88.7% and 89%, which is also a significant improvement from the F value of 0.4% and 16.8%. In terms of the FNR and FPR, the performance was similar for both the RF and the NN, showing a low rate of false negatives of 10.1% and 5.8% respectively and low rate of false positives, of less than 20%, which is once again a large improvement from the rates of 100% presented in the **Chapter 5.1.1**. Even though the SVM and RUSBoost did not present the highest scores they still performed well with 72.3% and 63.2% of accuracy, respectively and F1-score around 70%. However, these the RUSBoosted Trees showed a high false positive rate. These values were expected since the SMOTE method allowed to overcome the issue of the extremely imbalanced dataset present before.

In terms of the AUC values for the ROC curves, the classifiers also showed an improvement, with values above 60%. The RF and NN classifier stood out from the rest, with an AUC of 94,5% and 91.5%, respectively, and the SVM with 80,3% which means that the classifiers have an excellent predictive ability. The RUSBoosted Trees performed slightly lower, with an AUC of 64%, showing a good predictive ability.

5.1.3. Results for combining SMOTE and random undersampling for the majority class

In this step instead of oversampling the minority class to equal the number of observations corresponding to preterm and term, we decided to combine a process of oversampling the minority class and then undersample the majority class, to end with a 1:2 ratio in the train dataset.

First the minority class was oversampled, using the SMOTE algorithm, to have 40% of the number of examples of the majority class (e.g. about 1132). It is important to know that this value was agreed on after carefully searching for the percentage that would give better results, while still being fairly accurate. Then random undersampling was used to reduce the number of observations in the majority class to have 50% more than the minority class (e.g. 2264). After these steps, the final dataset had a total of 3396 observations, with 2264 term labor contractions and 1132 preterm labor contractions, as seen in **Figure 39.** This way the 1:7 ratio of the classes in the original dataset was transformed to a 1:2 ratio.

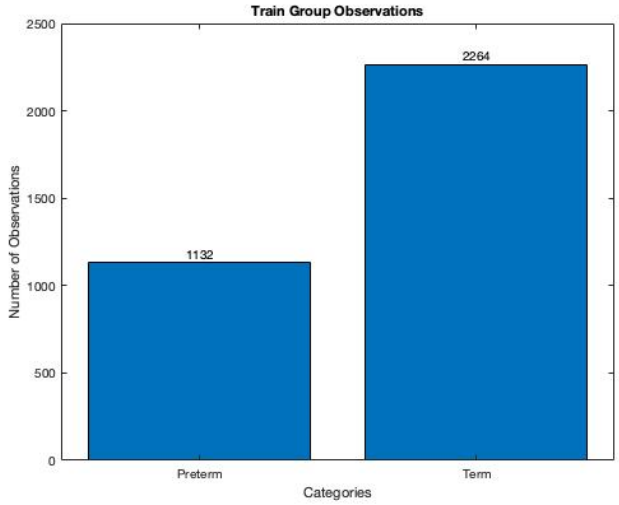


Figure 39: Number of observations for each class (Term or Preterm) in the training dataset after combining SMOTE with undersampling

The previous classifiers were applied including the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network. **Figure 43** displays the classification metrics for the training set, based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network, shown in **Table 12** to **Table 15**, respectively, and ROC curves in the attachment.

Table 12: RF classifier confusion matrix for the training dataset with SMOTE + undersampling

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2147	113
	Preterm Labor	462	668

Table 13: RUSBoosted Tree classifier confusion matrix for the training dataset with SMOTE + undersampling

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1286	974
	Preterm Labor	236	894

Table 14: SVM with the Gaussian kernel classifier confusion matrix for the training dataset with SMOTE + undersampling

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2259	1
	Preterm Labor	1095	35

Table 15: NN classifier confusion matrix for the training dataset with SMOTE + undersampling

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1870	394
	Preterm Labor	213	919

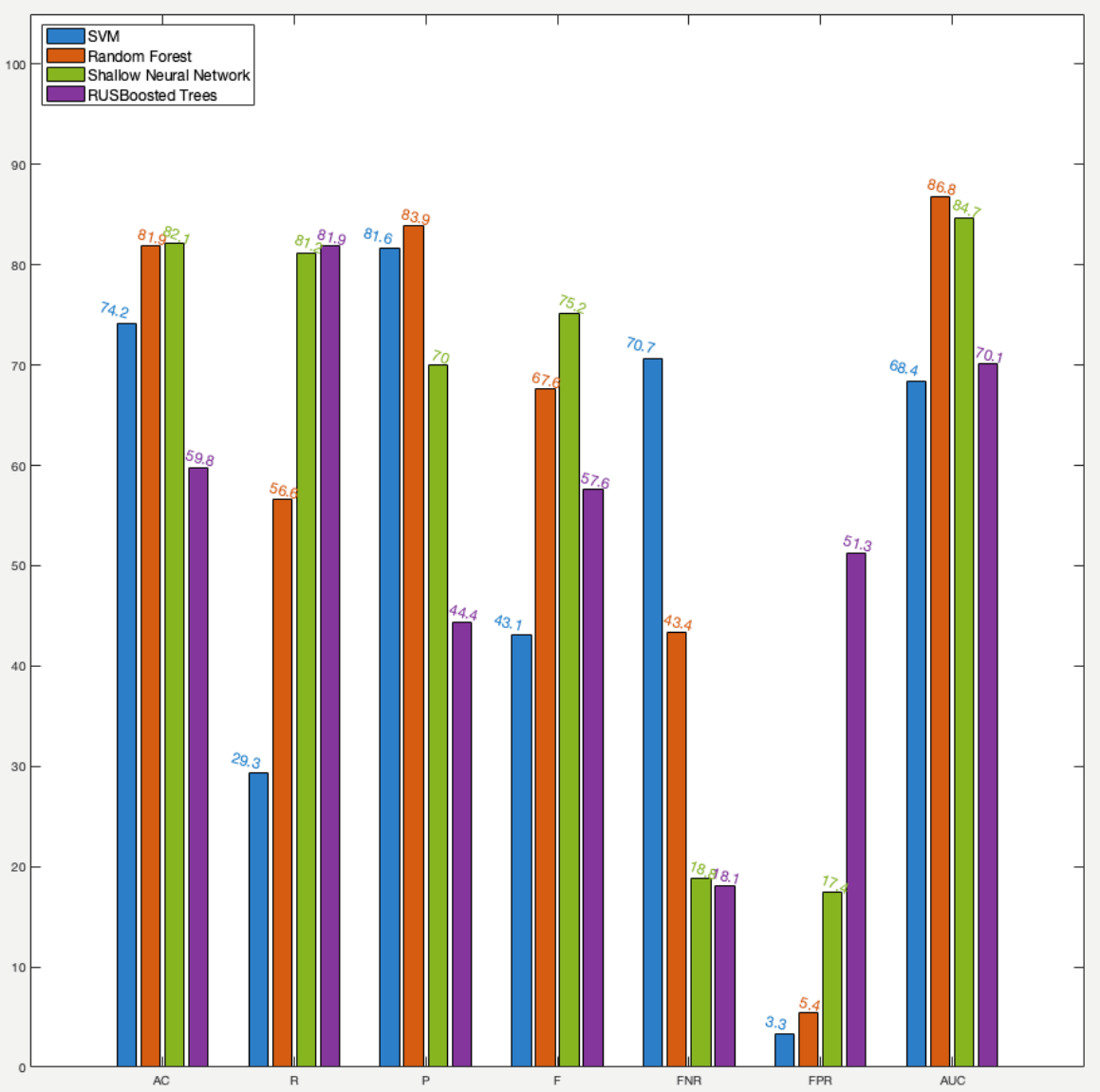


Figure 40: Machine learning performance results for the dataset using SMOTE with undersampling training dataset for comparison

As seen in **Figure 40** and in the confusion matrix of all classifiers, compared to the technique without the undersampling of the majority class, the classifiers performed worse overall, specifically in the recall score, meaning that the models started to produce more false negatives, which can also be seen with the increase of the FNR. This is usually associated with an imbalanced class or untuned model hyperparameters, which means that since we are currently working with a balanced dataset, the next step to improve our classifiers will be to tune the model hyperparameters.

The SVM, RF and NN showed the biggest change, with a significant drop in the F1-score value and the false negative rate increase. Even though these classifiers showed an increased capacity of identifying preterm labor, with a decrease on the the FPR, the change wasn't enough to construct a better model.

In terms of the AUC percentage values, the RUSBoosted Trees and SVM did not perform as well as before, with values around 70% which means that these classifiers have less predictive power, when comparing with the AUC value of 80,3% and 64% presented before. Finally, the Random Forest and

Shallow Neural Network algorithm continued to show the biggest value of the AUC of 86.5% and 84.7%, respectively, showing a good predictive capability.

This kind of results can be linked with the classifier having less observations from the minority class to learn from. Although the authors of the SMOTE paper showed that the results improved when combining random undersampling, the same did not happen with our work, and matching the number of observations in the minority class to the majority class, proved to be the best, with all algorithms showing better and more promising results for preterm l. In the next step we will use the SMOTE dataset used in 4.1.2, a) chapter to apply feature selection methods, dimensionality reduction and later a tuning of hyperparameters, to hopefully improve our results.

5.1.4. Results for the dataset combining PCA with the SMOTE dataset

After applying oversampling using SMOTE, our results increased significantly. With the purpose of making our classifier more robust and to avoid problems like overfitting, we chose to try the application of a PCA. This technique reduces the dimension of a dataset, so it is simpler and easier to work with, while preserving the information contained in it. This technique was applied keeping enough components to explain 95% of the variance.

The next step was to classify this new dataset with our different algorithms, to check for any improvements. The RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network were trained and validated using a 10-fold stratified cross-validation. The **Figure 43** displays the classification metrics for the training set, based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network, shown in **Table 16** to

Table 19, respectively.

Table 16: RF classifier confusion matrix for the training dataset after applying PCA

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2298	531
	Preterm Labor	343	2486

Table 17: RUSBoosted Tree classifier confusion matrix for the training dataset after applying PCA

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	956	1873
	Preterm Labor	539	2290

Table 18: SVM with the Gaussian kernel classifier confusion matrix for the training dataset after applying PCA

True Class \ Predicted Class	Term Labor	Preterm Labor
Term Labor	2218	611
Preterm Labor	1737	1092

Table 19: NN classifier confusion matrix for the training dataset after applying PCA

True Class \ Predicted Class	Term Labor	Preterm Labor
Term Labor	1995	834
Preterm Labor	493	2336

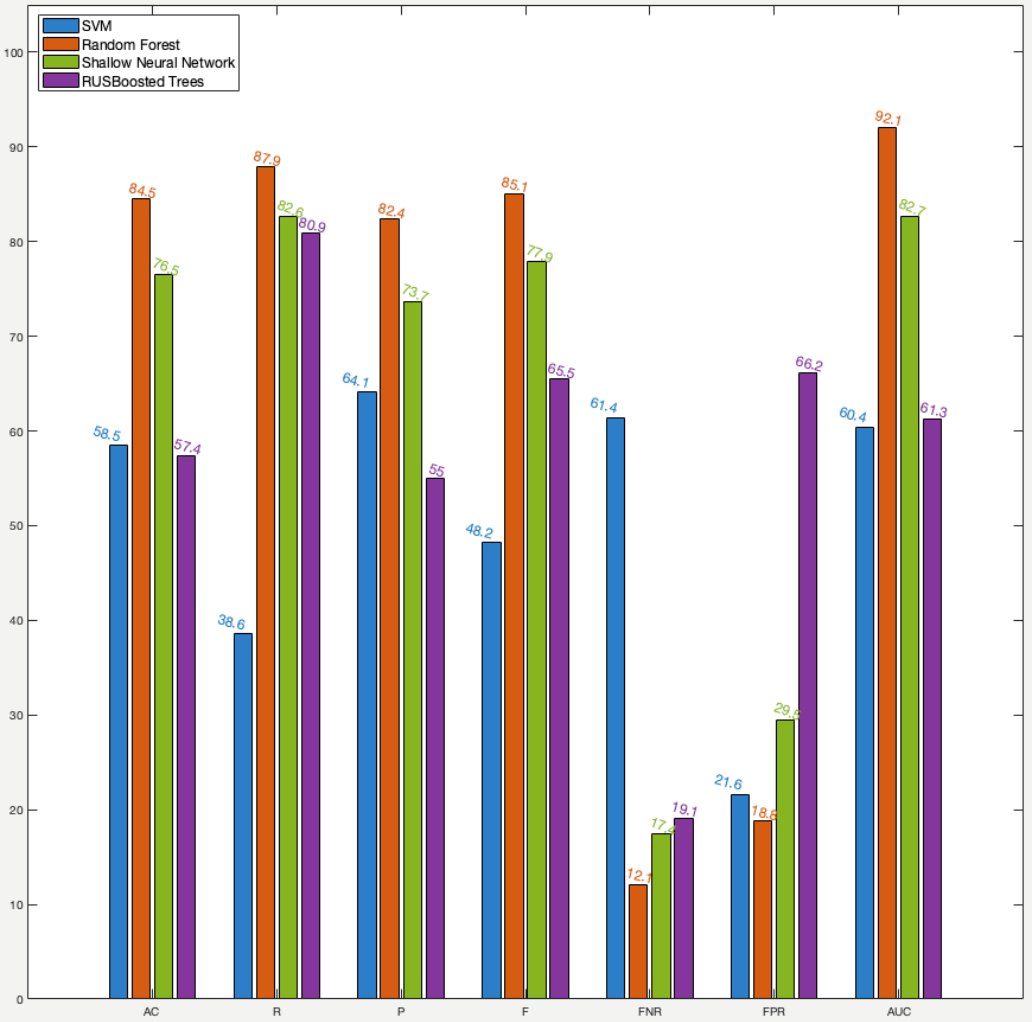


Figure 41: Machine learning performance results for the validation dataset for comparison after dimensionality reduction technique

In **Figure 43** it is visible that the PCA did not improve our results. Overall, the accuracy, recall, precision, F1-score and AUC got lower, while the FNR and FPR increased. The biggest noticeable difference is with the SVM classifier, in which you can see the biggest drop with the AC value dropping from 72,3% to 58,5%, the F value from the 67,8% to 48,2% and the AUC value from 80,3% to 60,4%.

By looking at these results, we conclude that PCA should not be used with our data, since it negatively influences the results for all classifiers. Although not expected, PCA can harm the results in classification problems, since as an unsupervised method, it will not take in consideration the labels of the observations. As a result, some features of a certain class, might not be considered since their labels are not correlated with the variance of the features, and for that reason worsening the results.

In the next step we will use the SMOTE dataset used in Chapter 50 to apply feature selection methods.

5.1.5. Results for the dataset with oversampling combining Feature Selection Methods with the SMOTE dataset

In this next phase, we continued to try to improve the performance of our classifiers. This step consisted in applying a feature selection method, explained in Chapter 4.9, to the training dataset with an

oversampled minority class. This technique was used since we are working with a high number of features, specifically 200, and some might be redundant, so by removing features with low predictive power, we can improve our models, so that the classifier is better at generalizing, avoiding overfitting. The feature ranking methods applied was the Minimum Redundancy Maximum Relevance (MRMR) Algorithm, and the correspondent feature importance ranking results are presented in Figure 42.

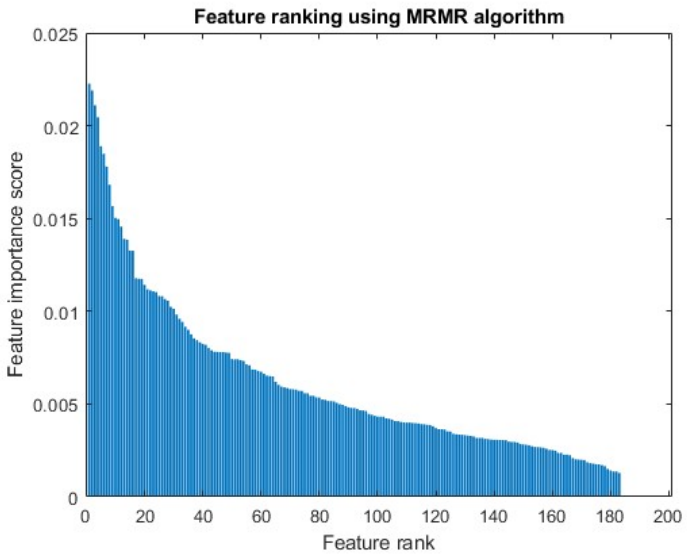


Figure 42: Feature ranking using Minimum Redundancy Maximum Relevance (MRMR) test rank

Looking at these results we can see in **Figure 42**, that only 183 features show any relevance, with 17 of the features scoring zero importance. With the goal of improving our dataset, we tried to establish a threshold within the ranking of **Figure 42,** to find the optimal number of features. Since there is not an evident drop in the score overall, three different number of features were selected: 16 (where the biggest drop of importance happens, from feature 16 to feature 17), 100 (considering a threshold equal to half of the number of features) and 183 (considering the only features that showed importance). It is important to mention that from feature 100 to feature 183 the feature importance0 score is approximately 0.005 and it keeps getting lower, therefore they are considered very low importance.

The next step was to classify these three datasets, to check for any improvements. RF, RUSBoosted Tree, SVM with Gaussian kernel, and Shallow Neural Network were trained and validated using 10-fold stratified cross-validation. **Figure 43** displays the classification metrics for the validation set using 183 features, based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel, and Shallow Neural Network, shown in Table 20 to **Table 23**, respectively. The ROC curves are represented in the attachment. The results for the other two datasets using 16 and 100 features are shown in the attachment since they performed poorer than the 183 features.

Table 20: RF classifier confusion matrix for the training dataset after feature selection

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2433	396
	Preterm Labor	321	2508

Table 21: RUSBoosted Tree classifier confusion matrix for the training dataset after feature selection

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1120	1709
	Preterm Labor	357	2472

Table 22: SVM with the Gaussian kernel classifier confusion matrix for the training dataset after feature selection

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2224	605
	Preterm Labor	1232	1597

Table 23: NN classifier confusion matrix for the training dataset after feature selection

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2295	534
	Preterm Labor	175	2654

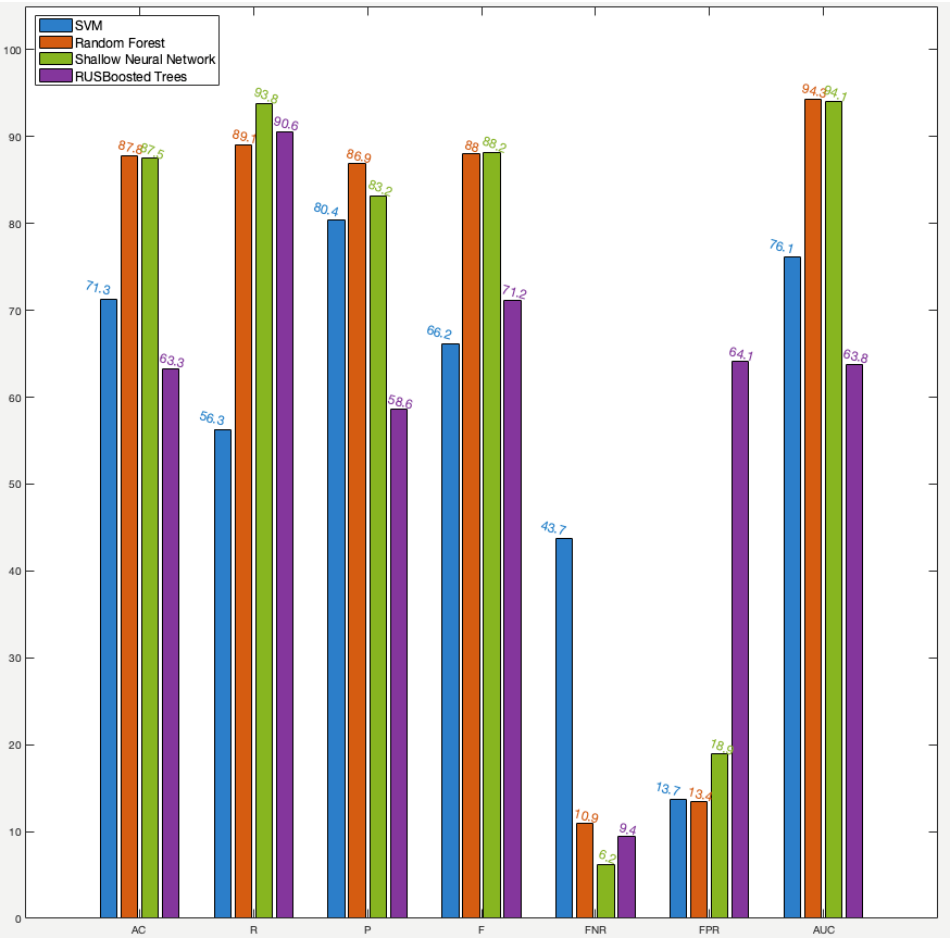


Figure 43: Machine learning performance results for the validation dataset with 183 features for comparison after feature selection

In **Figure 43** we can see that by removing some of the features that were poorly ranked, the results did not suffer any significant change, expect for the SVM that showed the biggest difference in the AUC value, with its value decreasing from 80 to 76. This means that the predictive ability of this classifier decreased slightly, but not enough to change the classification power. In terms of the F1-Score, in general the values stayed approximately the same. By deleting seventeen features with zero importance, the classifiers achieved, approximately, the same performance but with less training time and less chances of overfitting the data.

The next step in this thesis, was the last procedure for improvement, where the new dataset with only 183 features was used for hyperparameter optimization, with the Bayesian optimization method.

5.1.6. Results for the dataset applying SMOTE, Feature Selection and Hyperparameter Optimization

This next step consists of the final step in the machine learning classifier methods. Here we are tuning our model with our best performing dataset (oversampling of the minority class using the SMOTE technique with 183 features of the most relevant features) by selecting different advanced options, called hyperparameters, that strongly affect the performance of the algorithms. This technique is called Bayesian

optimization and was done by using the Classification Learner app from MatLab®, with 30 iterations each and using the "Expected improvement per second plus" method. This app tries different combinations of hyperparameter values and returns a final model with the optimized hyperparameters that minimized the model classification error and then trains it with the training data. The validation is then done with the same 10-fold stratified cross validation used previously. Finally, the test dataset is used in the trained model and the performance metrics are calculated. Only 30 iterations were chosen since a higher number was impossible to reach with the CPU used for this thesis.

For the Random Forest and RUSBoosted Tree model, the hyperparameters that were tuned were the maximum number of splits, the number of learners, learning rate and number of predictors to sample for the RF. The RF model was improved by 410 learners, with 3981 maximum number of splits and 2 predictors to sample. Comparing to the RUSBoosted Tree, the best tuned model had 12 learners, a learning rate of 0-814 and 500 maximum splits.

For the optimizable SVM model with the Gaussian kernel function, the Box constraint level and standardize data parameter were tuned. The model was improved by utilizing a a box constraint level of 590.7263 and the standardize data was set to false.

Finally, for the optimizable neural network, the number of fully connected layers, the activation function, standardize data parameter, regularization strength, first layer size, second layer size and the third layer size were tuned. The model was improved by utilizing three fully connected layers, the Tahn activation function, standardize data parameter was set to True, the first layer size was 221, the second layer size was 240 and the third layer size was 52.

The hyperparameter optimization results for the classification metrics for the training dataset are shown in **Figure 44.** Each of these results is based on the confusion matrix from the RF, RUSBoosted Tree, SVM with Gaussian kernel and Shallow Neural Network, shown in **Table 24** to **Table 27**, respectively, and the ROC curves shown in the attachment.

Table 24: RF classifier confusion matrix for the training dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2602	227
	Preterm Labor	188	2641

Table 25: RUSBoosted Tree Trees classifier confusion matrix for the training dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2013	816
	Preterm Labor	516	2313

Table 26: SVM with the Gaussian kernel classifier confusion matrix for the training dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2456	373
	Preterm Labor	315	2514

Table 27: NN classifier confusion matrix for the training dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2347	482
	Preterm Labor	156	2673

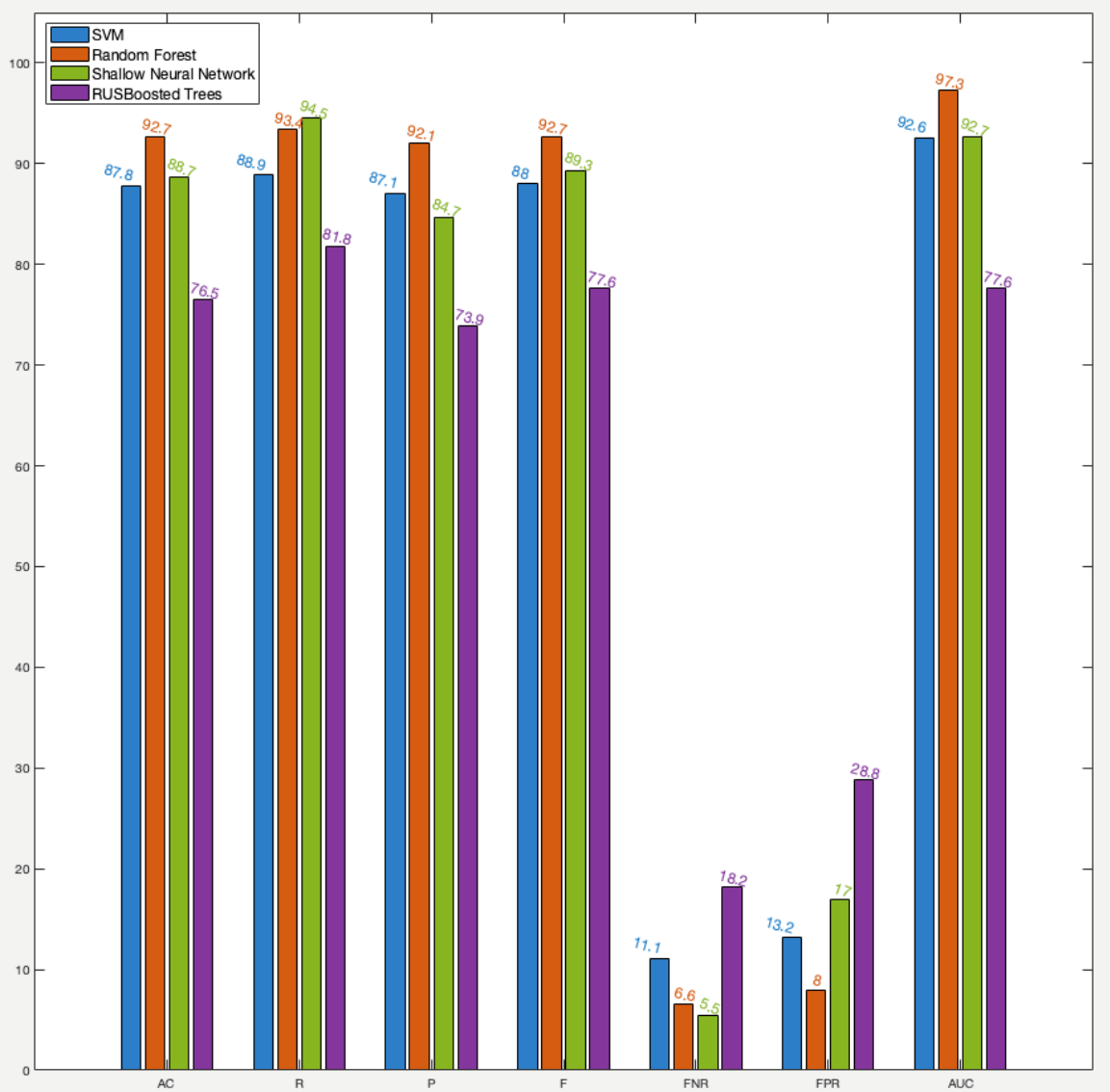


Figure 44: Machine learning performance results for the validation dataset for comparison after hyperparameter tuning

Analyzing the results in the **Figure 44** for the validation results, all classifiers performed better with the hyperparameter optimization. The accuracy, recall, precision, F1-score and AUC all improved. The SVM algorithm showed a significant increase in the recall value, from 58.3% to 88.9%, improving the ability of classifying the positive class, preterm labor. The optimized RF algorithm stood out from the rest, with a F of 92.7%, an AUC of 97.3%, a FNR 6.6% and FPR of 8%, while maintaining an accuracy of 92.7%. The RUSBoosted Trees performance in the precision, and false positive rate got better, increasing from 58.6% to 73.9% and decreasing from 63.6% to 28.8%.

After achieving such positive results, the models were tested with the test dataset, presented earlier in this chapter, with 1386 total observations. As mentioned before, although we are working with a dataset with the application of SMOTE and with feature selection, the test set will only include the original samples, so we have an unbiased estimate of how well the classifier works with new data. In **Figure 45** we can see all the results in a bar plot for comparison, and the confusion matrix for the RF, RUSBoosted Tree and SVM

with Gaussian kernel, from **Table 28** to **Table 31** respectively. The correspondent ROC curves are represented in the attachment.

Table 28: RF classifier confusion matrix for the test dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1106	106
	Preterm Labor	152	22

Table 29: RUSBoosted Tree classifier confusion matrix for the test dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	864	348
	Preterm Labor	121	53

Table 30: SVM with the Gaussian kernel classifier confusion matrix for the test dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	1041	171
	Preterm Labor	139	35

Table 31: NN classifier confusion matrix for the test dataset after hyperparameter tuning

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	992	220
	Preterm Labor	140	34

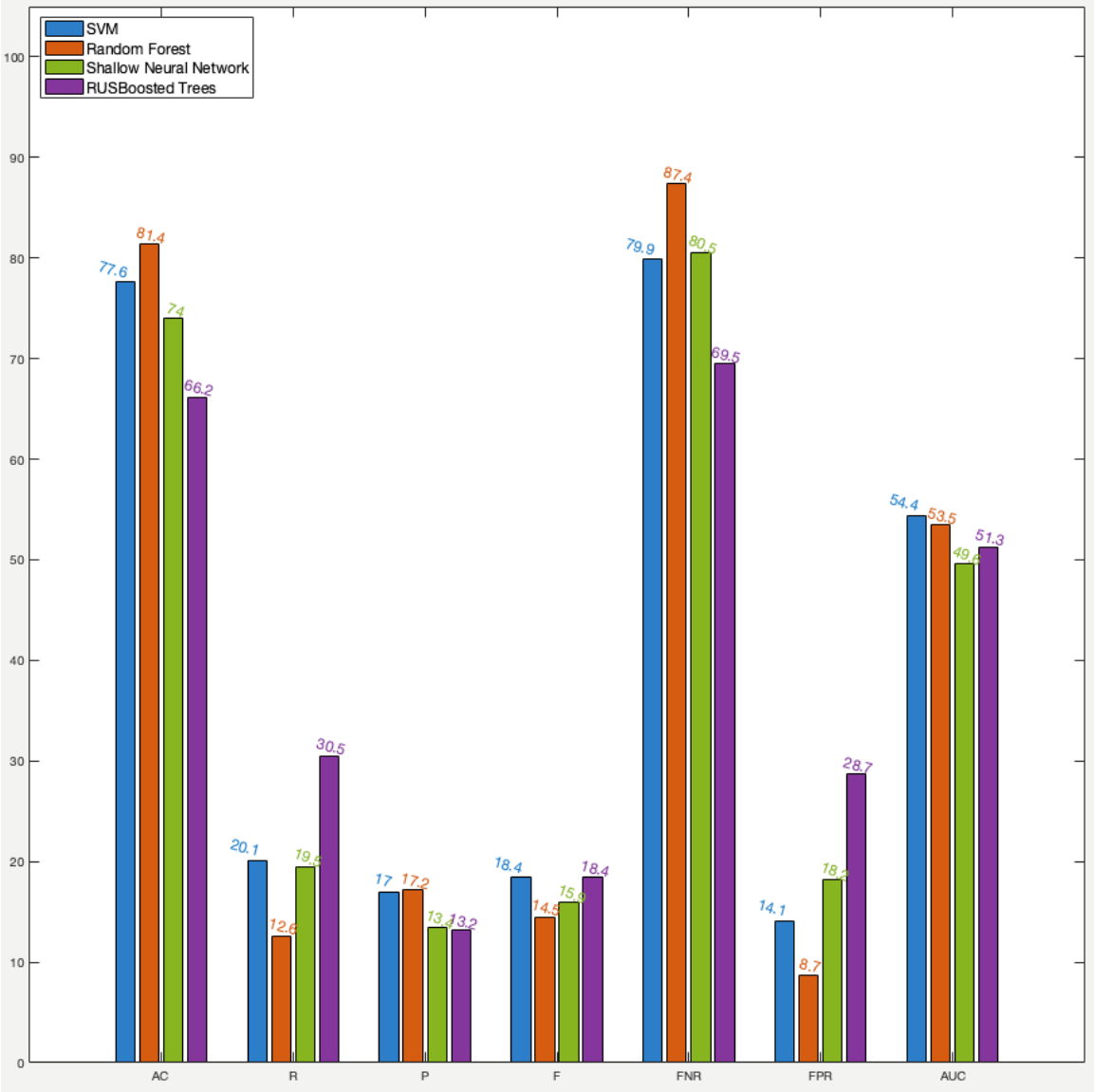


Figure 45: Machine learning performance results for the test dataset for comparison after hyperparameter tuning

For the test results we can see that the values were slightly lower than expected. Although the values for the accuracy remained similar for all classifiers, in the range of 70-80%, a good AC value, the precision and recall, as well as the F1-Score decreased at about 70%, overall. Additionally, the values for the FNR increased for all classifiers, at about 80%, while the FPR stayed approximately constant in the low range. Based on the high accuracy and the values presented in the confusion matrix from **Table 28** to **Table 31**, the classifiers are able to classify the term labors but struggle to predict preterm labors, justifying the high false negative rate. The AUC values sit between 50% and 60%, meaning that the predictions are just as good as random guesses.

It was expected for the results to be lower in the test set, since it is almost impossible to achieve the same performance on the test data as when validating on the training data. However, these values were way below expected. The main reason for this happening can be due to the change of the data quality in the train and test group, specifically the difference in the distribution of the data in the test set and training set. This

can be seen in **Table 32**, where the preterm term ratio drops from 1:1 to 1:7 in the test data, having almost 90% of term observations, thus being unrepresentative of the minority class. Apart from this it could also be the case of overfitting, and so the data is not able to generalize well to new unseen data. This is unlikely since from the beginning we set a robust model, to avoid any issues like this, by validating the results using stratified cross-validation, using feature selection to eliminate redundant features and complexity off the model, and adding more samples to the training data using the SMOTE technique.

Table 32: Comparison between Train and Test data distribution

	Train Data		Test Data	
	Count	**Percent**	**Count**	**Percent**
Term Labor	2829	50	1212	87.5
Preterm Labor	2829	50	174	12.6

In conclusion we can say that the RF had the best performance, even though all classifiers showed similar weak performances with the test set. This classifier showed the highest accuracy and precision, and lowest FPR, as well as one of the highest AUC value.

5.2. Deep Learning Methods

In this section we will present the performance evaluation results and discussion of the Deep Learning approaches. Firstly, we present the performance of the bidirectional LSTM with the original dataset, followed by the results obtained with the dataset treated with the oversampling technique, SMOTE. The original dataset was adapted to the raw structure, where we don't have the Welch variable organized into 200 features, but in a matrix of 1 by 200, demonstrating our sequential variable. This can be seen in **Figure A. 31**, shown in the attachment.

To process the observations (power spectra of the contractions), a bidirectional LSTM layer was chosen. This type of layer looks at the sequence in both forward and backward directions, which can help capture temporal dependencies. Since our observations have one dimension each, the chosen input size of the sequences is 1. The LSTM layer had 200 units, and the output was set to the last timestep to map the input into 200 features and prepare the output for the fully connected layer. Finally, a fully connected layer with a size of 2 was chosen to represent the two classes, term and pre-term, followed by a sigmoid layer (suited for binary classification problems) and a classification layer. **Figure 46** shows a code snippet of this model.

```
layers = [ ...
sequenceInputLayer(1)
bilstmLayer(200,'OutputMode','last')
fullyConnectedLayer(2)
sigmoidLayer
classificationLayer
]
```

Figure 46: Layer architecture for the LSTM algorithm

For the training options, represented in **Figure 47** we can see that "*MaxEpochs*" option was sent to 50 to allow the network to make 50 runs through the training dataset. A "*MiniBatchSize*" of 128 was chosen

so the network could look at 128 training observations at a time. The "*InitialLearnRate*" was 0.00001 to help speed up the training process, with a "*SequenceLength*" of 30 to help the algorithm to look for shorter pieces of the observations. The "*GradientThreshold*" was 1 to prevent gradients from getting too larger and to stabilize the training process.

```
options = trainingOptions('adam', ...
    'MaxEpochs',50, ...
    'MiniBatchSize',128, ...
    'InitialLearnRate', 0.00001, ...
    'SequenceLength', 30, ...
    'GradientThreshold', 1, ...
    'ExecutionEnvironment',"auto",...
    'plots','training-progress', ...
    'Verbose',false);
```

Figure 47: Training options for the LSTM algorithm

After specifying the layer architecture and training options the LSTM network was trained, using the "*trainNetwork*" function from MathWorks. The training process can be seen in **Figure 48** for the original dataset and in **Figure 49** for the dataset with SMOTE. On the top subplot we can see the training accuracy, that represents the classification accuracy on each mini-batch, and in the bottom we see the training loss, that corresponds to the cross-entropy loss on each mini-batch and ideally would decrease towards zero.

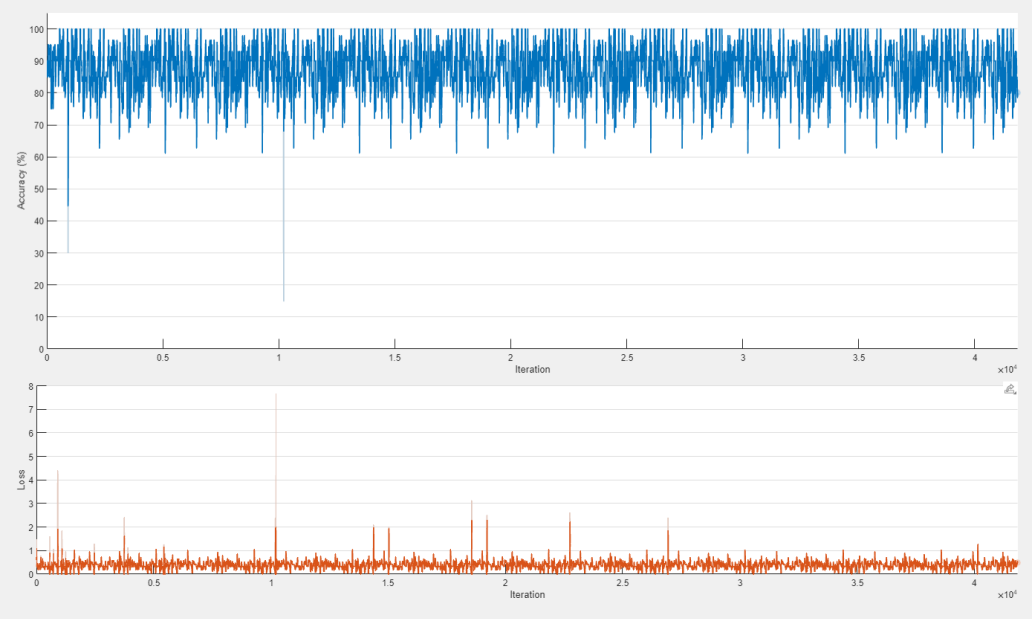


Figure 48: Training Progress for the LSTM network - Original Dataset

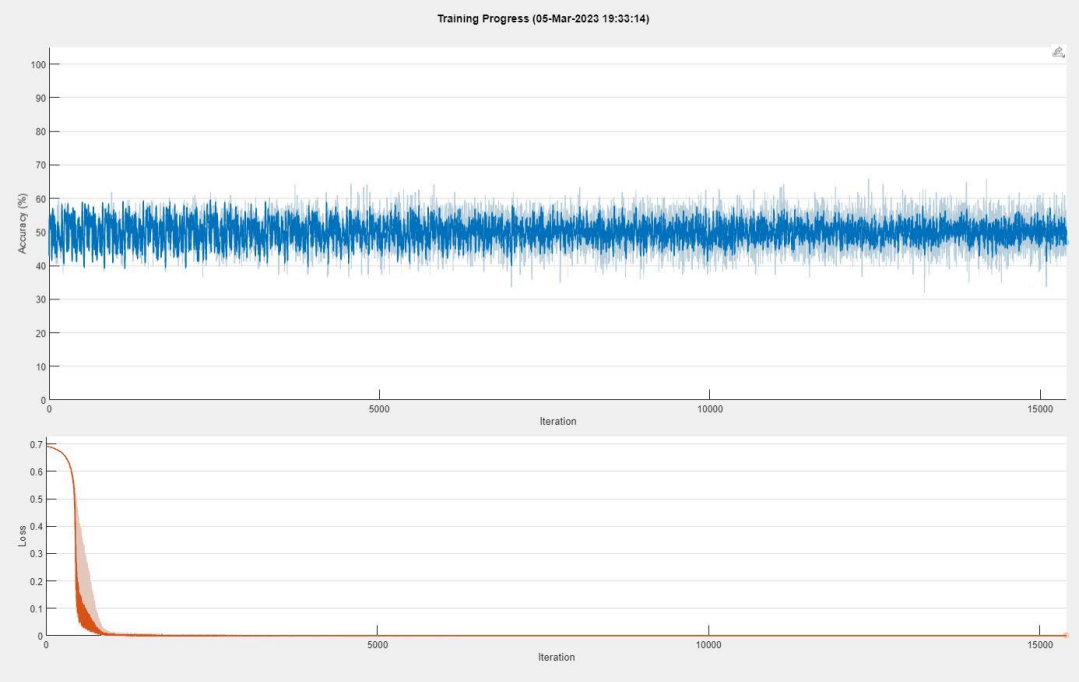


Figure 49: Training Progress for the LSTM network – Dataset with SMOTE application

In **Figure 48** we can see the training progress for the LSTM network in the original dataset. The accuracy oscilates between about 60% and 100%, and the loss function varies between zero and one. In **Table 33** the confusion matrix for the LSTM network is represented, where it is visible that the algorithm made no predictions for preterm labor but was able to successfully predict all term labor observations.

For the LSTM training with the SMOTE dataset, the training progress was recorded in **Figure 49,** where the accuracy sits between the range of 40-60%, while the loss function decreases to zero. In **Table 33** we can see the correspondent confusion matrix. For this dataset we would expect the results to improve, however that was not the case, and no predictions for preterm labor were visible, explaining the FNR value of 100% seen in **Figure 50**.

In **Figure 50** we see that for both datasets the deep learning classifier did not produce good results, showing a value of zero in all metrics and a 100% for the FNR, since all preterm labors were classified as term. Although it showed a high accuracy with 87,4% for the original dataset and a more modest value of 66,7% for the SMOTE+undersampling dataset, as previously mentioned in the ML classifiers results, a high accuracy, especially in this dataset, means very little, telling us that the algorithm just learned with the term observations.

Table 33: Confusion Matrix for the training dataset for the LSTM network – Original Dataset

Predicted Class

True Class		**Term Labor**	**Preterm Labor**
	Term Labor	2829	0
	Preterm Labor	407	0

Table 34: Confusion Matrix for the training dataset for the LSTM network – Dataset with SMOTE application

		Predicted Class	
		Term Labor	**Preterm Labor**
True Class	**Term Labor**	2829	0
	Preterm Labor	2829	0

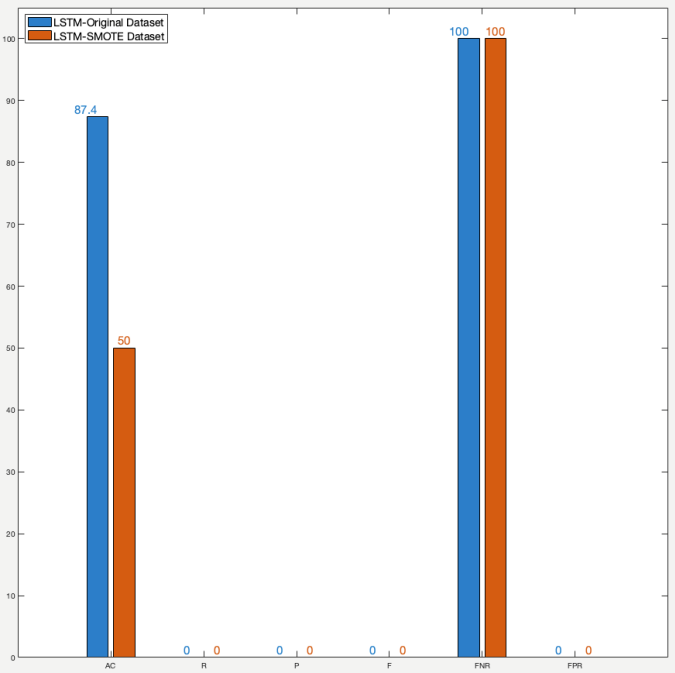


Figure 50: Deep learning performance results for the train datasets for comparison

After training the model we proceed to test the test dataset in the trained model. The results were again not successful and are shown in **Table 35** and **Table 36** for the for the confusion matrix, for the original dataset and for the dataset with SMOTE+undersampling, respectively.

Table 35: Confusion Matrix for the test dataset for LSTM network – Dataset with SMOTE application

	Predicted Class	
	Term Labor	**Preterm Labor**
Term Labor	1212	0
Preterm Labor	174	0

Table 36: Confusion Matrix for the test dataset for LSTM network – Dataset with SMOTE application

True Class	Predicted Class: Term Labor	Predicted Class: Preterm Labor
Term Labor	1212	0
Preterm Labor	174	0

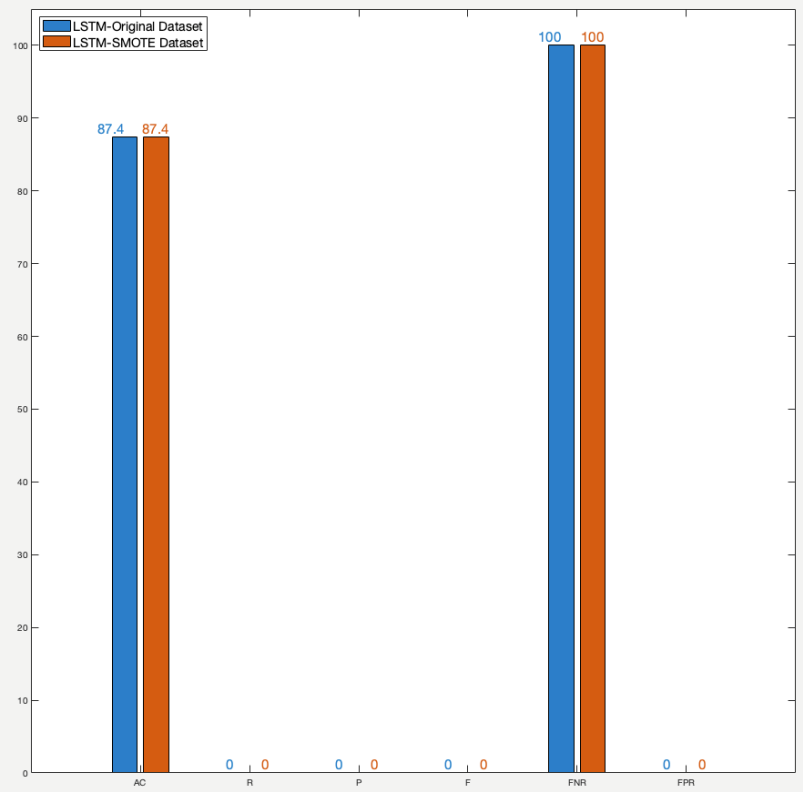


Figure 51: Deep learning performance results for the test datasets for comparison

In **Figure 51** we can see that once again the algorithm shows a high accuracy, of 87.4% and for both the original dataset and for the dataset after applying SMOTE. Additionally, the only metric that shows results different from 0 is the FNR, with a value of 100%. This value is extremely high, especially for this metric, where the value should be the lowest possible. Once again, this is a result of the network only learning with the majority class observations, visible on the confusion matrix in **Table 35** and **Table 36**.

Overall, the LSTM model did not perform well in predicting preterm labor, possibly due to overfitting. This highlights the importance of having a large and diverse dataset when using deep learning algorithms. The results obtained with the oversampled dataset using SMOTE were not satisfactory, indicating that oversampling techniques alone may not be sufficient to improve the performance of deep learning models. Even though the power of the LSTM algorithm is very enticing, these results show that with a smaller dataset, like ours, will not produce the expected results.

5.3. Results comparison with the literature review studies

We will compare our best performing classifiers in the ML and DL chapter, the Random Forest classifier and LSTM, with the classifiers from the studies in the literature review, in **Table 37.**

Table 37: Summary of the ML related work studies

Algoritm	AC (%)	R (%)	P (%)	F (%)	FNR (%)	FPR (%)	AUC (%)
Proposed RF classifier	**81.4**	**12.6**	**17.2**	**14.5**	**87.4**	**8.7**	**53.5**
Proposed LSTM classifier	**87.4**	**0**	**0**	**0**	**100**	**0**	**x**
RF proposed by Peng et al. [3]	93	89	x	x	x	x	80
DT proposed by Oliver et al. [16]	90	x	x	x	x	x	x
RF proposed by Allahem et al. [5]	95.7	96.3	92.9	94.5	3.6	4.6	99
DT proposed by Allahem et al. [5]	95.3	94.1	93.7	93.9	5.8	3.8	98
RF proposed by SAĞLAM et al. [21]	87.28	x	x	x	x	x	x
ANN proposed by Allahem et al. [5]	98	98	97	98	10	10	99

We can see in **Table 37** that most studies do not have all the evaluation criteria used in this work, except for the work of Allahem et al [5]. This makes our work more challenging since a lot of important measures like the F1-Score and the AUC are missing in literature, like in the DT proposed model by Xu et al. [20] or the RF proposed by SAĞLAM et al. [21], which only shows the value for the AC, which can be misleading when working with imbalanced test sets, as mentioned before. All the models proposed by Allahem et al. [5], show more promising results than our own, both in the ML and DL approaches. As mentioned in **Chapter 2**, these authors choose to use several different EHG datasets over oversampling, to make sure the quality of the data was not an issue, even though there is no evidence of the accuracy of that when comparing to real data.

Taking all of this into consideration, we can say that in terms of accuracy our proposed RF model classifies well in relation to the rest of the models, however in terms of the other metrics, except for the FPR, our proposed model scores significantly lower than the rest. Looking at the high value for the false negative rate, compared with the low FNR presented by Allahem et al. [5], it might also agree with the issue of having very proportionally different train and test sets, and an excess of synthetic samples in the training set, that decreased the classifiers' ability in identifying the majority sample with the test set. By comparing our study with the one from Allahem et al. [5], it shows that the use of synthetic samples on the training set might have a higher impact on the results than expected.

6. Conclusions

The prediction of preterm birth using machine learning (ML) and deep learning (DL) algorithms with electrohysterogram (EHG) signals is an ever-evolving field, but the lack of preterm data for classification remains a significant hurdle to further progress. It is important to note that preterm labor is a complex problem since there is no specific diagnostic for two thirds of these labours and there is very limited data for preterm births. Additionally, the ML field is vast, encompassing a variety of algorithms, data balancing techniques, feature selection algorithms, and different hyperparameter tuning methods. Therefore, finding the most robust dataset with the right features and the best performing classifier involves a trial-and-error strategy.

The results with the imbalanced TPEHG dataset were insignificant, indicating the considerable difference between the number of preterm records (minority classs) and term records (majority class). To address the skewed dataset problem, an oversampling technique called SMOTE was applied after partitioning the dataset, ensuring the test set remained untouched, achieving better and more significant results on the training dataset. However, combining oversampling of the minority class with SMOTE and randomly undersampling the majority class did not improve the results when compared to applying a more "common" SMOTE technique without undersampling. Applying PCA to the dataset also did not improve the results as expected. Furthermore, the F1-Score and AUC classification metrics appeared to be the most trustworthy metrics, correctly representing the performance of each classifier, while Accuracy, one of the most used in the literature, could be misleading with our imbalanced dataset.

This study proposes a novel approach to the classification problem by using the Welch power spectra of each contraction found in the EHG signal as features to predict preterm labor. Although promising results were obtained for all classifiers in the training dataset, for ML, using SMOTE with feature selection and hyperparameter tuning, with high values of F1-score, Accuracy, and AUC, the test data produced disappointing results with low AUC and F1-score, but good accuracy. These results can be linked to the test dataset, which is very imbalanced, causing the classifiers to perform badly to this unseen data.

In the DL network, the LSTM could not predict any preterm labor, possibly due to overfitting. This methodology problem is associated with the lack of data, severely impacting the performance of DL algorithms, which require big data to produce excellent and reliable results. In future studies, alternative deep learning models and architectures can be explored to improve the accuracy of preterm labor prediction.

While this study opens up various innovative investigation possibilities, including the possibility of using more complex features like the power spectra of each contraction, it is associated with some methodological problems, including the need to overcome the imbalanced data problem. Using synthetic samples to address this problem can decrease the algorithms' predictive ability for the majority class, especially not knowing about the quality of these samples in terms of similarity to the actual dataset. Moreover, different oversampling alternatives must be further investigated. As of now, we can say that these features were not successful and should not be use under these conditions, since they don't produce better results than the literature.

This study highlights the importance of collecting more preterm EHG recordings to improve the results without the use of SMOTE and to successfully use LSTM and other ML algorithms as a classification approach. In the future, this work may contribute to the discovery of a successful classification tool for preterm labor using EHG signals, preventing preterm birth.

7. Appendix

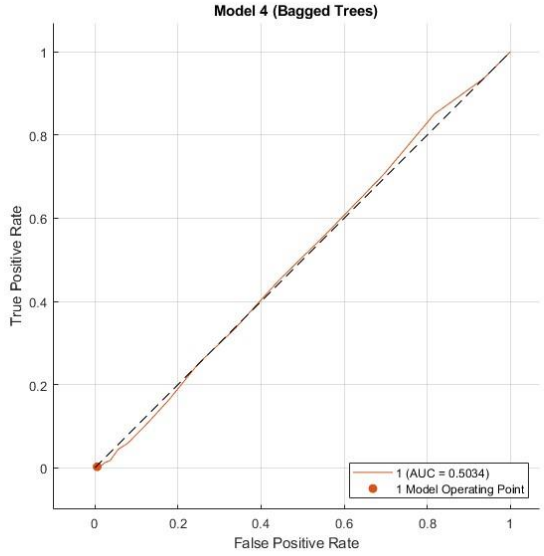


Figure A. 1: ROC curve for the validation set for the original dataset using RF as the classifier

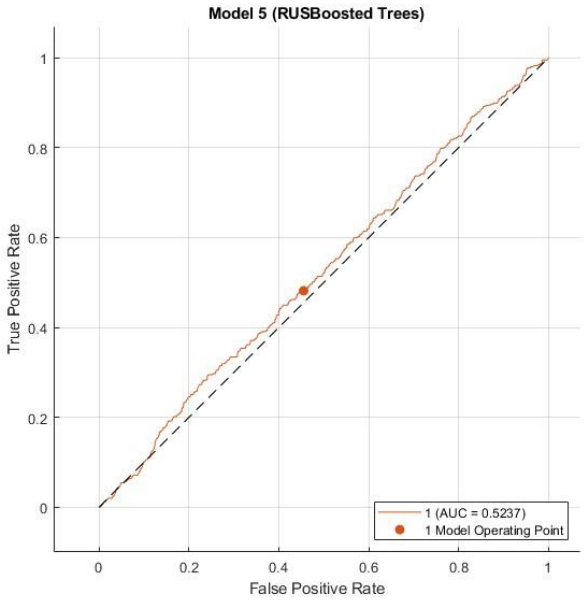


Figure A. 2: ROC curve for the validation set for the original dataset using RUSBoosted Tree as the classifier

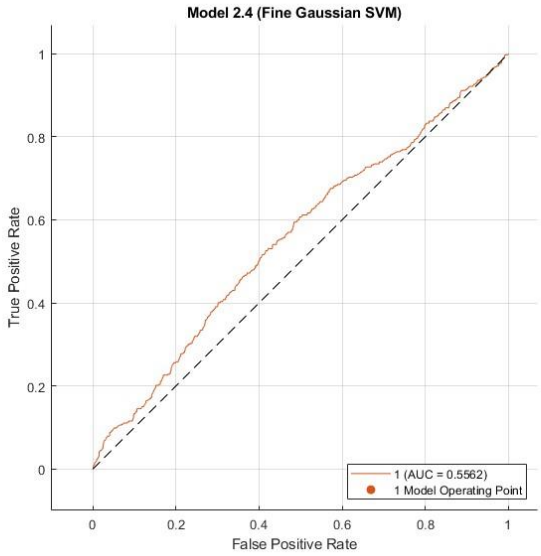


Figure A. 3:ROC curve for the validation set for the original dataset using SVM with Gaussian Kernel as the classifier

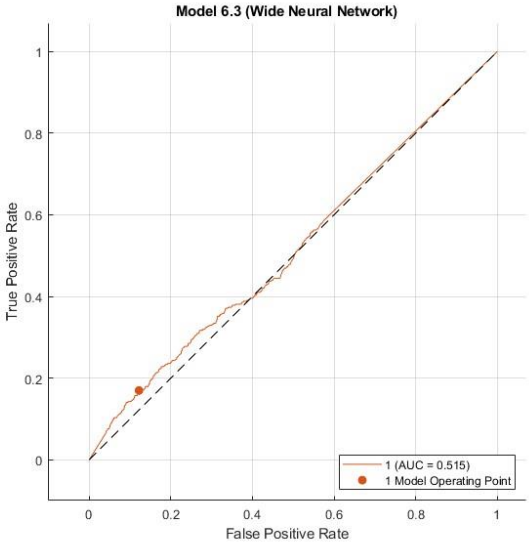


Figure A. 4: ROC curve for the validation set for the original dataset using a Shallow Neural Network

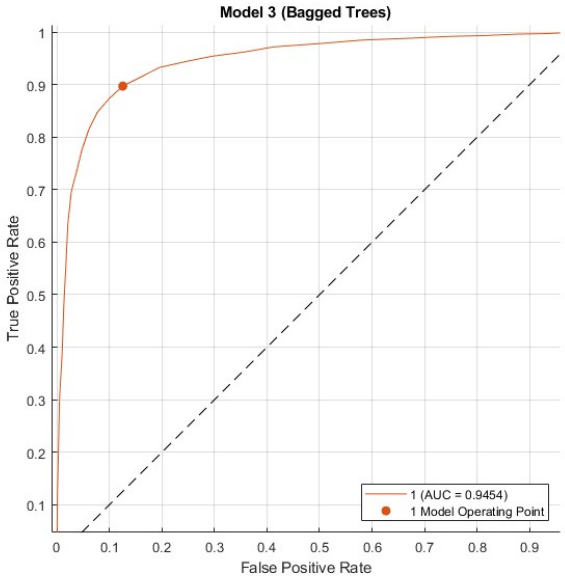
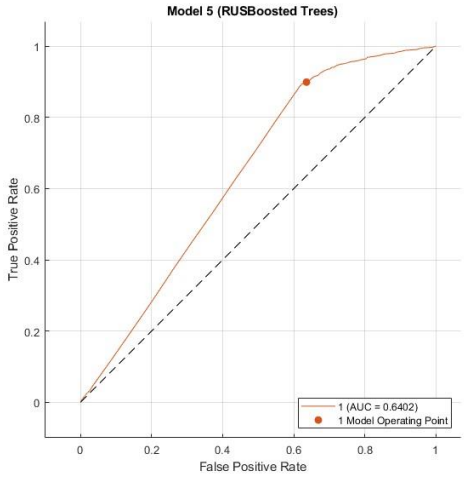


Figure A. 5: ROC curve for validation set after applying the SMOTE using RF as the classifier

Figure A. 6: ROC curve for validation set after applying the SMOTE using RUSBoosted Tree as the classifier



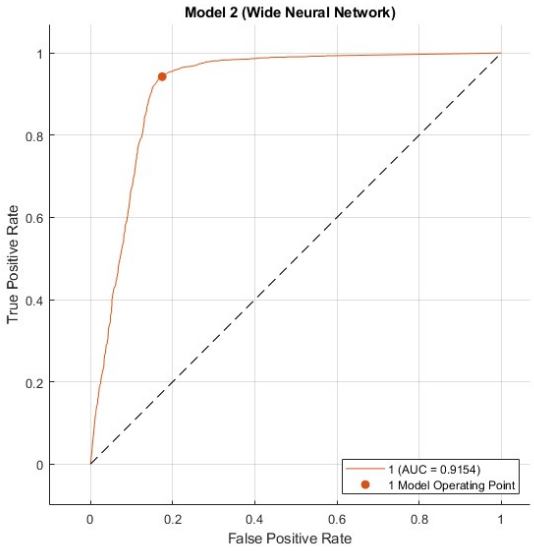


Figure A. 7: ROC curve for validation set after applying the SMOTE using a Shallow Neural Network

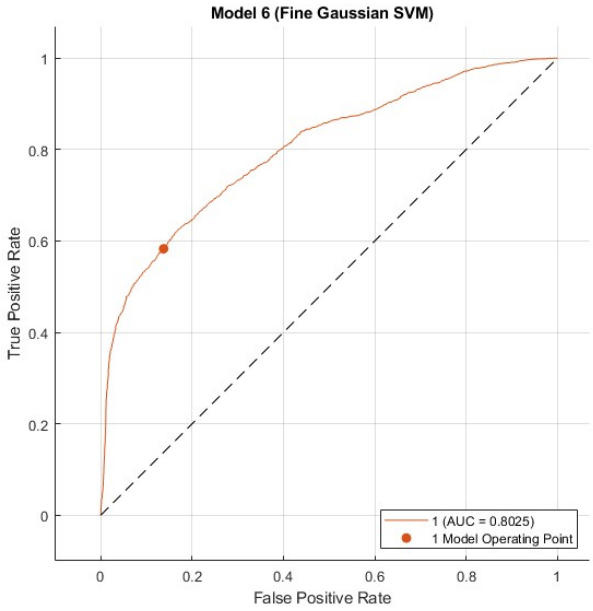


Figure A. 8: ROC curve for validation set after applying the SMOTE using SVM with the Gaussian kernel as the classifier

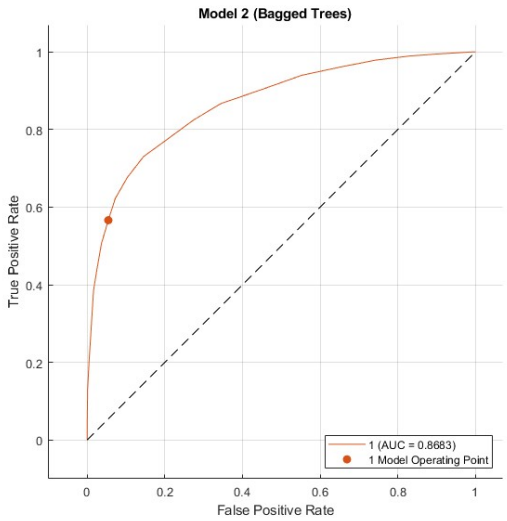


Figure A. 9: ROC curve for the validation set after applying SMOTE + undersampling using RF as the classifier

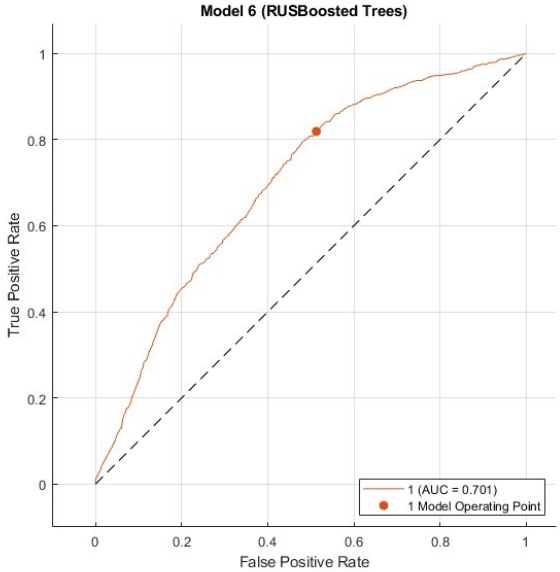


Figure A. 10: ROC curve for the validation set after applying SMOTE + undersampling using RUSBoosted Tree as the classifier

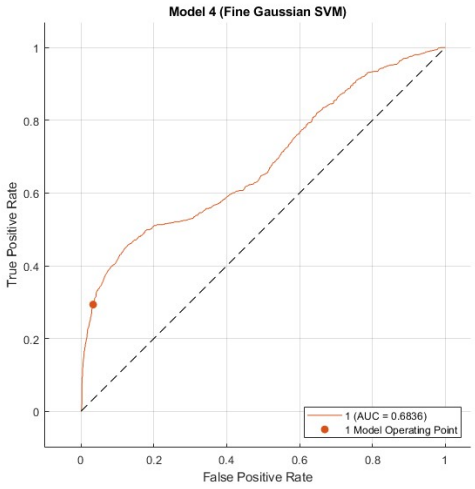


Figure A. 11:ROC curve for the validation set after applying SMOTE + undersampling using SVM with Gaussian Kernel as the classifier

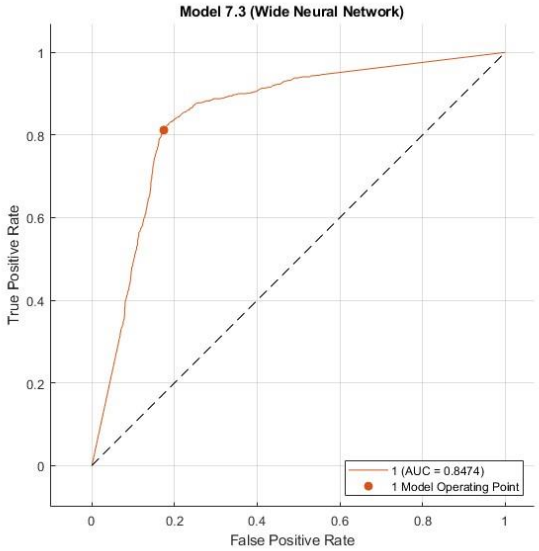
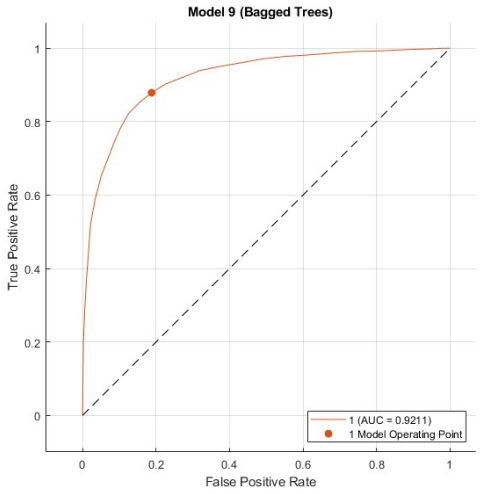


Figure A. 12: ROC curve for the validation set after applying SMOTE + undersampling using Shallow Neural Network as the classifier

Figure A. 13: ROC curve for the validation set after applying PCA using RF as the classifier



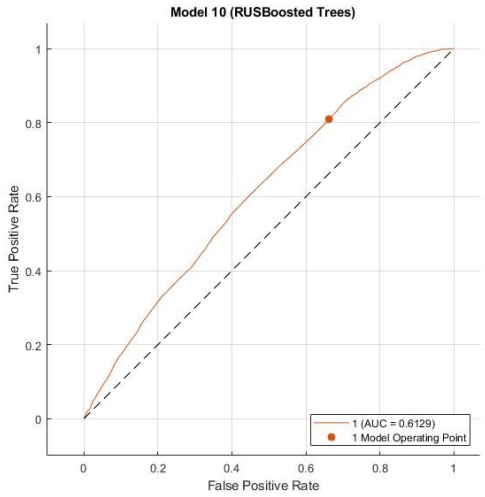


Figure A. 14: ROC curve for the validation set after applying PCA using RUSBoosted Tree as the classifier

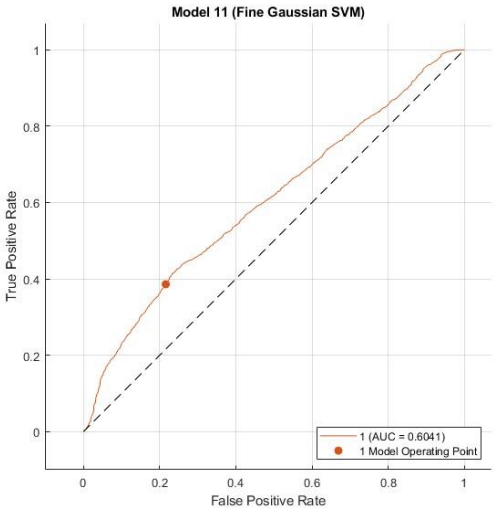


Figure A. 15:ROC curve for the validation set after applying PCA using SVM with a Gaussian Kernel as the classifier

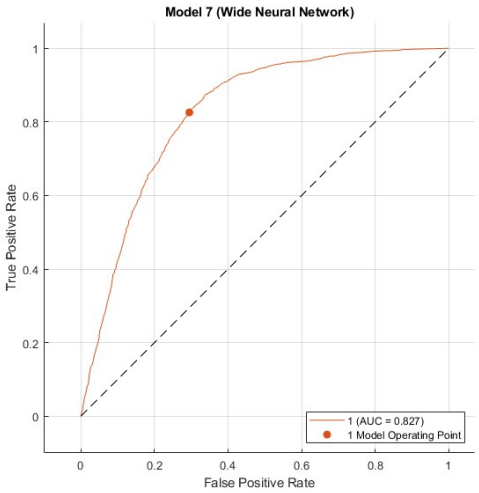
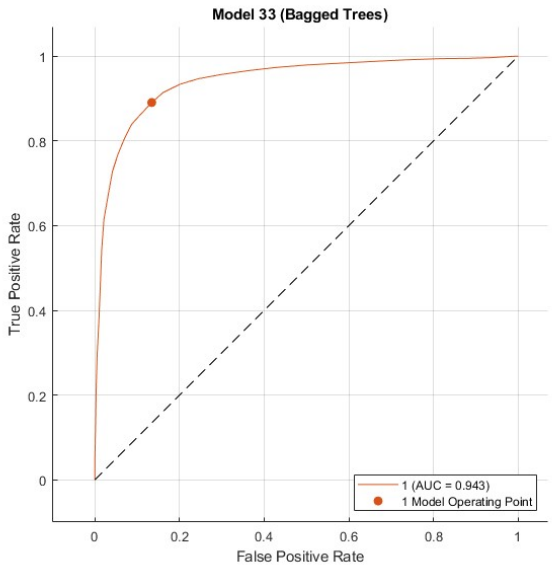


Figure A. 16: ROC curve for the validation set after applying PCA using a Shallow Neural Network as the classifier

Figure A. 17: ROC curve for the validation set after feature selection using RF as the classifier



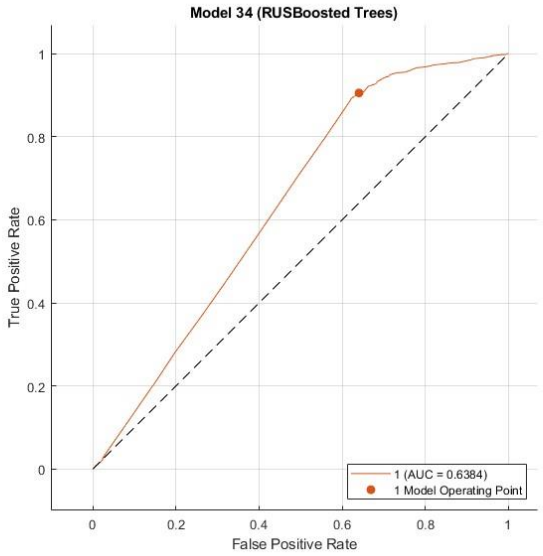


Figure A. 18: ROC curve for the validation set after feature selection using RUSBoosted Tree as the classifier

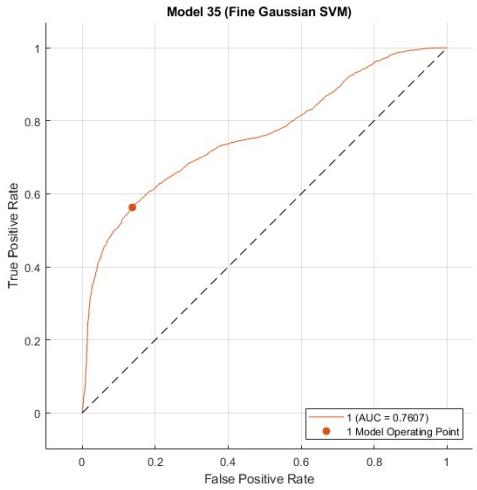


Figure A. 19: ROC curve for the validation set after feature selection using SVM with a Gaussian Kernel as the classifier

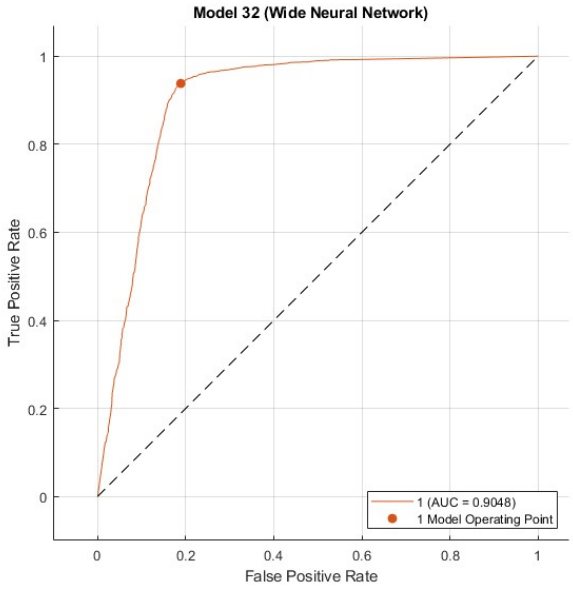
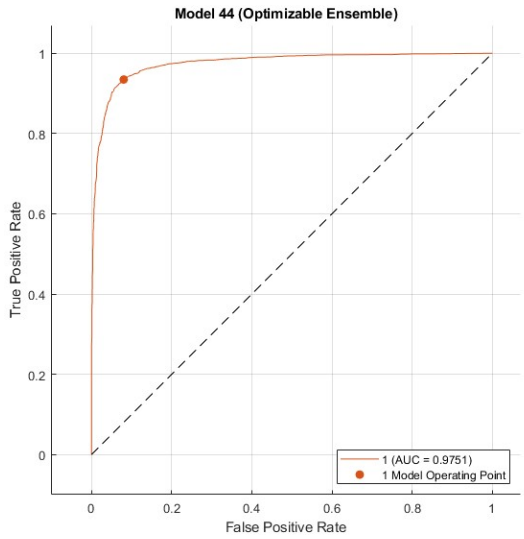


Figure A. 20: ROC curve for the validation set after feature selection using a Shallow Neural Network as the classifier

Figure A. 21: ROC curve for the validation set after hyperparameter tuning using RF as the classifier



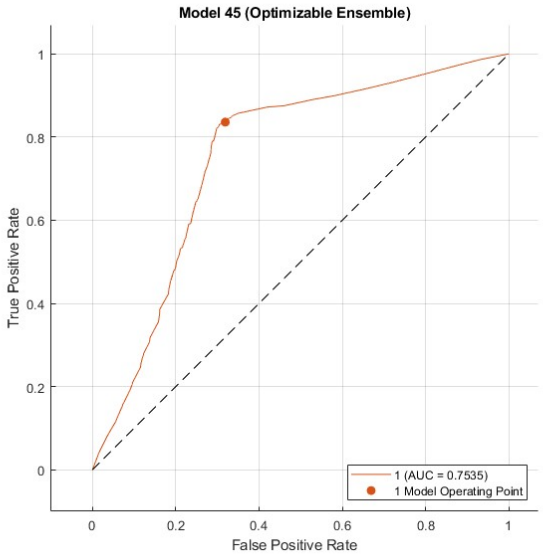


Figure A. 22: ROC curve for the validation set after hyperparameter tuning using RUSBoosted Tree as the classifier

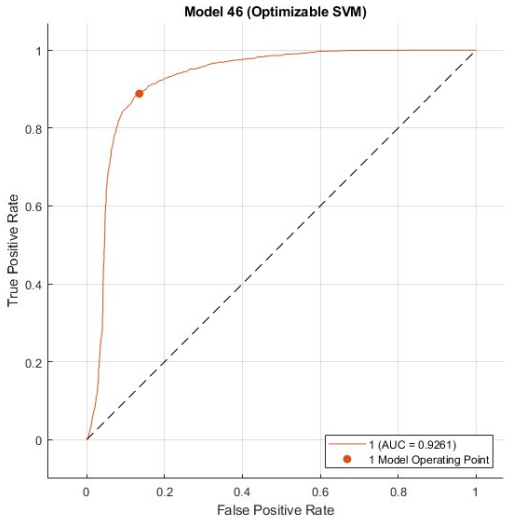


Figure A. 23: ROC curve for the validation set after hyperparameter tuning using SVM with a Gaussian Kernel as the classifier

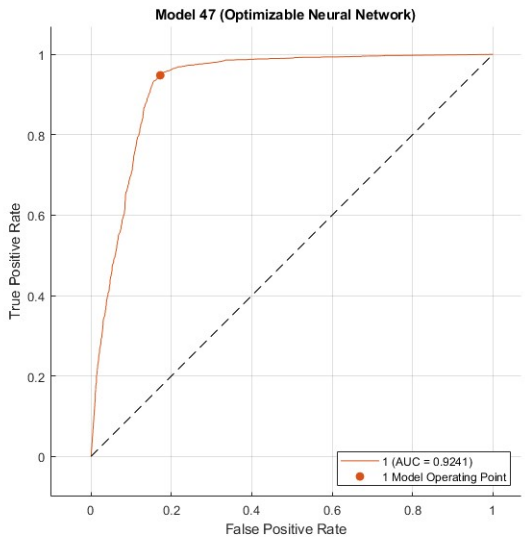
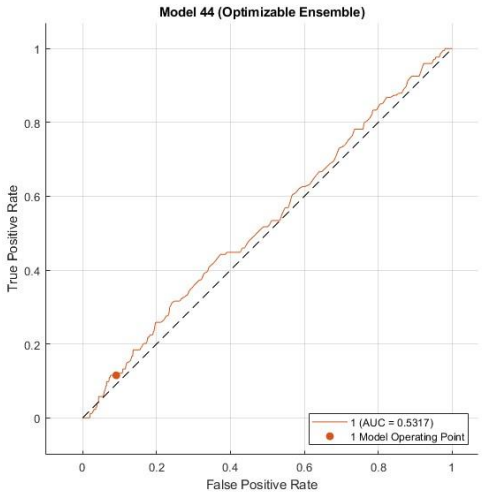


Figure A. 24: ROC curve for the validation set after hyperparameter tuning using a Shallow Neural Network as the classifier

Figure A. 25: ROC curve for the test set using RF as the classifier



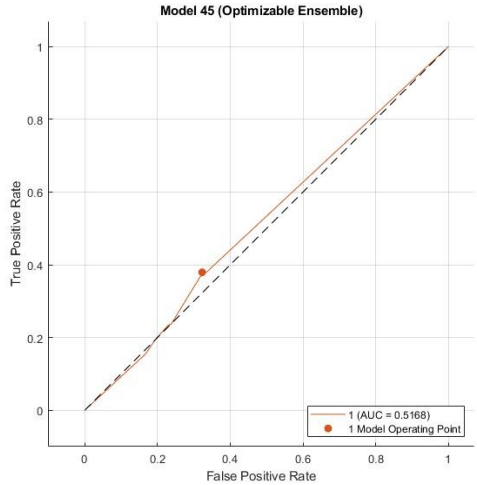


Figure A. 26: ROC curve for the test set using RUSBoosted Tree as the classifier

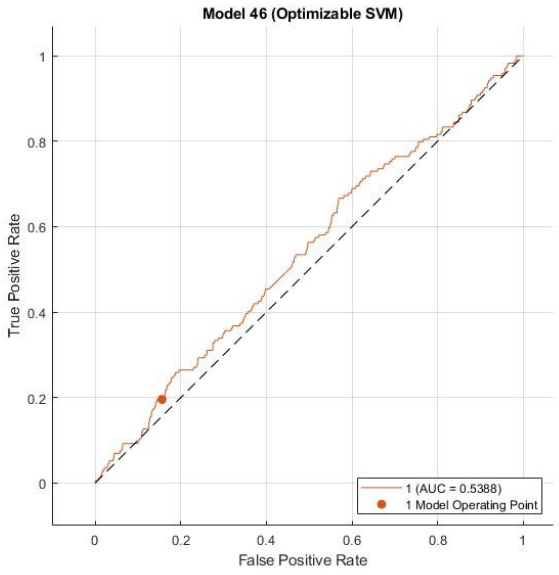


Figure A. 27: ROC curve for the test set using SVM with a Gaussian Kernel as the classifier

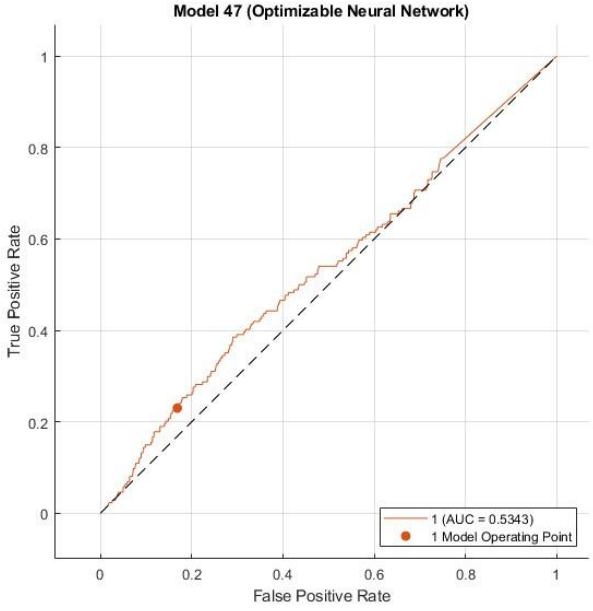


Figure A. 28: ROC curve for the test set using a Shallow Neural Network as the classifier

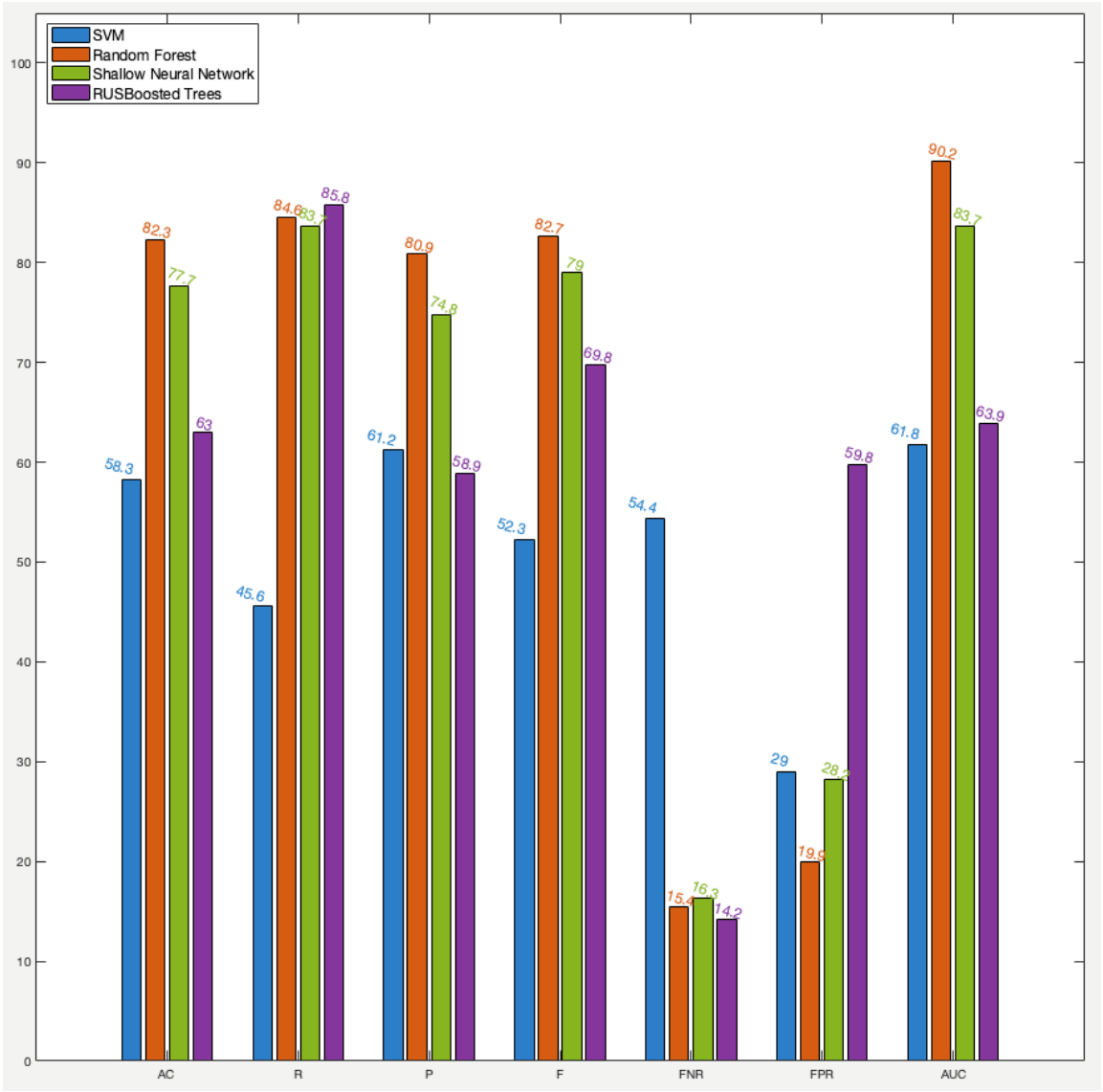


Figure A. 29: Machine learning performance results for the validation dataset with 16 features for comparison after feature selection

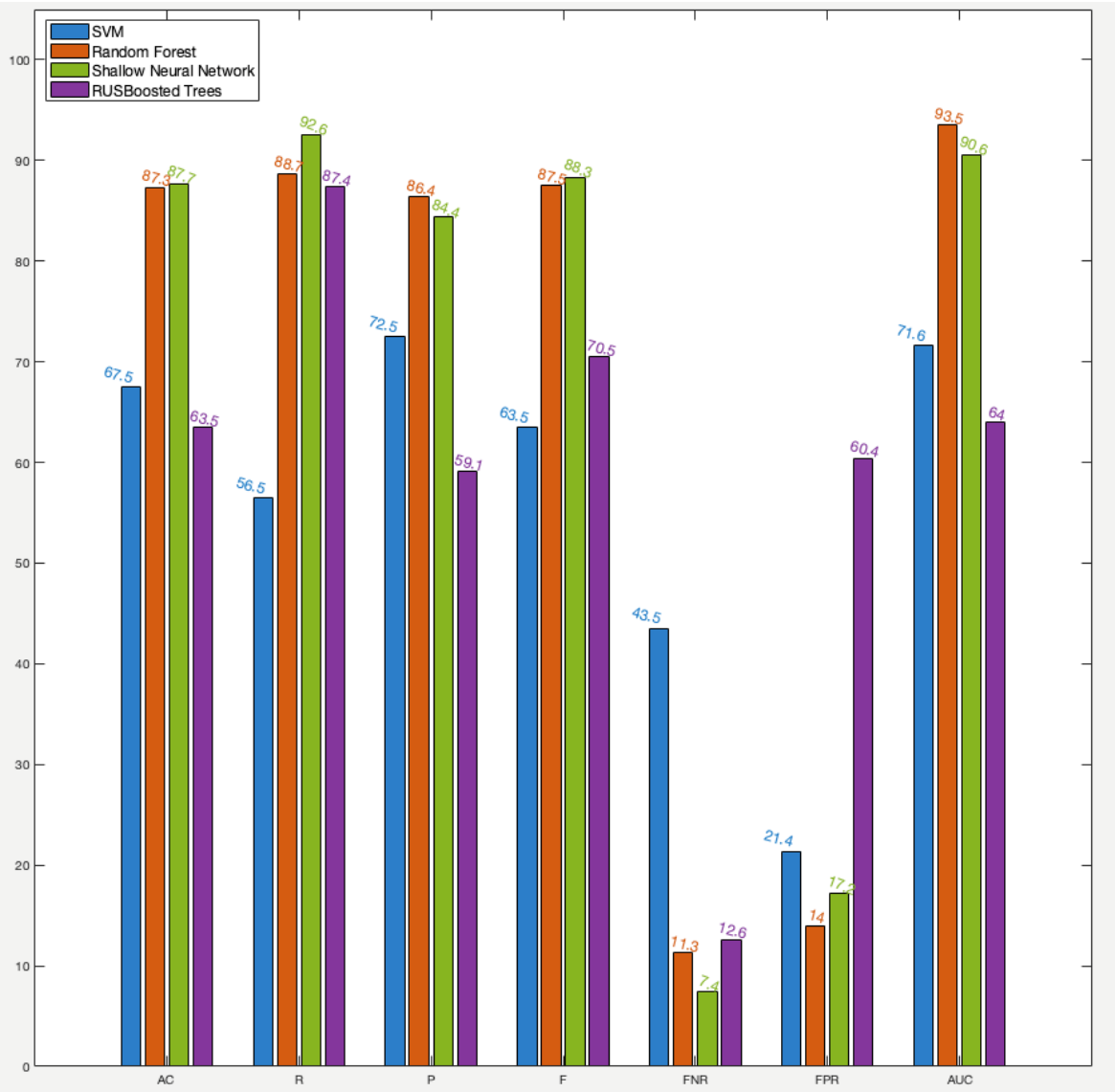


Figure A. 30: Machine learning performance results for the validation dataset with 100

(a)

	1 Classification	2 Welch
1	1	*1x200 double*
2	1	*1x200 double*
3	1	*1x200 double*
4	1	*1x200 double*
5	1	*1x200 double*
6	1	*1x200 double*
7	1	*1x200 double*
8	1	*1x200 double*
9	1	*1x200 double*
10	1	*1x200 double*
11	1	*1x200 double*
12	1	*1x200 double*

(b)

1	2	3	4	5	6	7	8	9
5.4583e-...	1.0895e-...	1.0822e-...	1.0673e-...	1.0419e-...	1.0036e-...	9.5313e-...	8.9678e-...	8.502

Figure A. 31: (a) LSTM Dataset (b) Observation Structure

8. Bibliography

[1] World Health Organization, "Preterm birth," *World Health Organization website*, 2022. https://www.who.int/news-room/fact-sheets/detail/preterm-birth (accessed Jul. 15, 2022).

[2] J. Peng *et al.*, "Evaluation of electrohysterogram measured from different gestational weeks for recognizing preterm delivery: a preliminary study using random Forest," *Biocybern Biomed Eng*, vol. 40, no. 1, pp. 352–362, Jan. 2020, doi: 10.1016/j.bbe.2019.12.003.

[3] J. Xu, Z. Chen, H. Lou, G. Shen, and A. Pumir, "Review on EHG signal analysis and its application in preterm diagnosis," *Biomedical Signal Processing and Control*, vol. 71. Elsevier Ltd, Jan. 01, 2022. doi: 10.1016/j.bspc.2021.103231.

[4] J. Xu, Z. Chen, J. Zhang, Y. Lu, X. Yang, and A. Pumir, "Realistic preterm prediction based on optimized synthetic sampling of EHG signal," *Comput Biol Med*, vol. 136, Sep. 2021, doi: 10.1016/j.compbiomed.2021.104644.

[5] H. Allahem and S. Sampalli, "Automated labor detection framework to monitor pregnant women with a high risk of premature labor using machine learning and deep learning," *Inform Med Unlocked*, vol. 28, Jan. 2022, doi: 10.1016/j.imu.2021.100771.

[6] I. O. Idowu, P. Fergus, A. Hussain, C. Dobbins, and H. al Askar, "Advance Artificial Neural Network Classification Techniques Using EHG for Detecting Preterm Births," in *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, Jul. 2014, pp. 95–100. doi: 10.1109/CISIS.2014.14.

[7] J. Ryu and C. Park, "Time-Frequency Analysis of Electrohysterogram for Classification of Term and Preterm Birth," *IEIE Transactions on Smart Processing and Computing*, vol. 4, no. 2, pp. 103–109, Apr. 2015, doi: 10.5573/IEIESPC.2015.4.2.103.

[8] A. J. Hussain, P. Fergus, D. Al-Jumeily, H. Alaskar, and N. Radi, "The Utilisation of Dynamic Neural Networks for Medical Data Classifications- Survey with Case Study," 2015, pp. 752–758. doi: 10.1007/978-3-319-22053-6_80.

[9] A. J. Hussain, P. Fergus, H. Al-Askar, D. Al-Jumeily, and F. Jager, "Dynamic neural network architecture inspired by the immune algorithm to predict preterm deliveries in pregnant women," *Neurocomputing*, vol. 151, no. P3, pp. 963–974, Mar. 2015, doi: 10.1016/j.neucom.2014.03.087.

[10] N. Sadi-Ahmed, B. Kacha, H. Taleb, and M. Kedir-Talha, "Relevant Features Selection for Automatic Prediction of Preterm Deliveries from Pregnancy ElectroHysterograhic (EHG) records," *J Med Syst*, vol. 41, no. 12, Dec. 2017, doi: 10.1007/s10916-017-0847-8.

[11] S. Hoseinzadeh and M. C. Amirani, "Use of Electro Hysterogram (EHG) Signal to Diagnose Preterm Birth," in *26th Iranian Conference on Electrical Engineering, ICEE 2018*, Sep. 2018, pp. 1477–1481. doi: 10.1109/ICEE.2018.8472416.

[12] F. Jager, S. Libenšek, and K. Geršak, "Characterization and automatic classification of preterm and term uterine records," *PLoS One*, vol. 13, no. 8, Aug. 2018, doi: 10.1371/journal.pone.0202125.

[13] M. Shahbakhti, M. Beiramvand, M. R. Bavi, and S. M. Far, *A New Efficient Algorithm for Prediction of Preterm Labor; A New Efficient Algorithm for Prediction of Preterm Labor*. 2019. doi: 10.0/Linux-x86_64.

[14] L. Chen, Y. Hao, and X. Hu, "Detection of preterm birth in electrohysterogram signals based on wavelet transform and stacked sparse autoencoder," *PLoS One*, vol. 14, no. 4, Apr. 2019, doi: 10.1371/journal.pone.0214712.

[15] S. Saleem *et al.*, "Granger causal analysis of electrohysterographic and tocographic recordings for classification of term vs. preterm births," *Biocybern Biomed Eng*, vol. 40, no. 1, pp. 454–467, Jan. 2020, doi: 10.1016/j.bbe.2020.01.007.

[16] JCT College of Engineering and Technology and Institute of Electrical and Electronics Engineers, *Proceedings of the 4th International Conference on Inventive Systems and Control (ICISC 2020) : 8-10 January, 2020*.

[17] D. K. Degbedzui and M. E. Yüksel, "Accurate diagnosis of term–preterm births by spectral analysis of electrohysterography signals," *Comput Biol Med*, vol. 119, Apr. 2020, doi: 10.1016/j.compbiomed.2020.103677.
[18] L. Chen and H. Xu, "Deep neural network for semi-automatic classification of term and preterm uterine recordings," *Artif Intell Med*, vol. 105, May 2020, doi: 10.1016/j.artmed.2020.101861.
[19] F. Esgalhado *et al.*, "Automatic contraction detection using uterine electromyography," *Applied Sciences (Switzerland)*, vol. 10, no. 20, pp. 1–14, Oct. 2020, doi: 10.3390/app10207014.
[20] A. SAĞLAM, Ü. ŞENTÜRK, and İ. YÜCEDAĞ, "Premature Birth Detection from EHG signals," *European Journal of Science and Technology*, Nov. 2021, doi: 10.31590/ejosat.1014179.
[21] S. Mohammadi Far, M. Beiramvand, M. Shahbakhti, and P. Augustyniak, "Prediction of Preterm Delivery from Unbalanced EHG Database," *Sensors (Basel)*, vol. 22, no. 4, Feb. 2022, doi: 10.3390/s22041507.
[22] G. Vandewiele *et al.*, "Overly optimistic prediction results on imbalanced data: a case study of flaws and benefits when applying over-sampling," *Artif Intell Med*, vol. 111, Jan. 2021, doi: 10.1016/j.artmed.2020.101987.
[23] N. v Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," 2002.
[24] E. B. da Fonseca, R. Damião, and D. A. Moreira, "Preterm birth prevention," *Best Pract Res Clin Obstet Gynaecol*, vol. 69, pp. 40–49, Nov. 2020, doi: 10.1016/j.bpobgyn.2020.09.003.
[25] Portuguese Pediatrics Society, "Dia Mundial da Prematuridade," *Portuguese Pediatrics Society website*. https://www.spp.pt/noticias/default.asp?IDN=372&op=2&ID=132 (accessed Nov. 15, 2022).
[26] J. Tucker and W. McGuire, "Epidemiology of preterm birth," *BMJ*, vol. 329, no. 7467, pp. 675–678, Sep. 2004, doi: 10.1136/bmj.329.7467.675.
[27] D. M. Ferrero *et al.*, "Cross-Country Individual Participant Analysis of 4.1 Million Singleton Births in 5 Countries with Very High Human Development Index Confirms Known Associations but Provides No Biologic Explanation for 2/3 of All Preterm Births," *PLoS One*, vol. 11, no. 9, p. e0162506, Sep. 2016, doi: 10.1371/journal.pone.0162506.
[28] J.-M. Moutquin, "Classification and heterogeneity of preterm birth," *BJOG*, vol. 110, pp. 30–33, Apr. 2003, doi: 10.1046/j.1471-0528.2003.00021.x.
[29] P. R and S. D. S, "Acquisition and Analysis of Electrohysterogram Signal," *J Med Syst*, vol. 44, no. 3, p. 66, Mar. 2020, doi: 10.1007/s10916-020-1523-y.
[30] R. C. YOUNG, "Myocytes, Myometrium, and Uterine Contractions," *Ann N Y Acad Sci*, vol. 1101, no. 1, pp. 72–84, Feb. 2007, doi: 10.1196/annals.1389.038.
[31] D. A. MACINTYRE, E.-C. CHAN, and R. SMITH, "MYOMETRIAL ACTIVATION – COORDINATION, CONNECTIVITY AND CONTRACTILITY," *Fetal Matern Med Rev*, vol. 18, no. 4, pp. 333–356, Nov. 2007, doi: 10.1017/S0965539507002033.
[32] R. E. Garfield and W. L. Maner, "Physiology and electrical activity of uterine contractions," *Semin Cell Dev Biol*, vol. 18, no. 3, pp. 289–295, Jun. 2007, doi: 10.1016/j.semcdb.2007.05.004.
[33] A. R. Mohan, J. A. Loudon, and P. R. Bennett, "Molecular and biochemical mechanisms of preterm labor," *Semin Fetal Neonatal Med*, vol. 9, no. 6, pp. 437–444, Dec. 2004, doi: 10.1016/j.siny.2004.08.001.
[34] F. Cardoso, "Uterine Contractions Clustering Based on Surface Electromyography: An Input for Pregnancy Monitoring," 2018.
[35] B. R. Hayes-Gill, "Monica Healthcare: From the research laboratory to commercial reality—A real-life case study," *Healthc Technol Lett*, vol. 8, no. 1, pp. 1–10, Feb. 2021, doi: 10.1049/htl2.12004.
[36] G. Li *et al.*, "Active Laplacian electrode for the data-acquisition system of EHG," *J Phys Conf Ser*, vol. 13, pp. 330–335, Jan. 2005, doi: 10.1088/1742-6596/13/1/077.
[37] L. Chen and Y. Hao, "Feature Extraction and Classification of EHG between Pregnancy and Labor Group Using Hilbert-Huang Transform and Extreme Learning Machine," *Comput Math Methods Med*, vol. 2017, pp. 1–9, 2017, doi: 10.1155/2017/7949507.

[38] B. Moslem, M. Diab, M. Khalil, and C. Marque, “Combining data fusion with multiresolution analysis for improving the classification accuracy of uterine EMG signals,” *EURASIP J Adv Signal Process*, vol. 2012, no. 1, p. 167, Dec. 2012, doi: 10.1186/1687-6180-2012-167.
[39] Y. Ye-Lin, G. Prats-Boluda, J. Alberola-Rubio, J.-M. Bueno Barrachina, A. Perales, and J. Garcia-Casado, “Prediction of labor using non-invasive laplacian EHG recordings,” in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2013, pp. 7428–7431. doi: 10.1109/EMBC.2013.6611275.
[40] A. Alexandersson, T. Steingrimsdottir, J. Terrien, C. Marque, and B. Karlsson, “The Icelandic 16-electrode electrohysterogram database,” *Sci Data*, vol. 2, no. 1, p. 150017, Apr. 2015, doi: 10.1038/sdata.2015.17.
[41] B. Hayes-Gill *et al.*, “Accuracy and Reliability of Uterine Contraction Identification Using Abdominal Surface Electrodes,” *Clin Med Insights Womens Health*, vol. 5, p. CMWH.S10444, Jan. 2012, doi: 10.4137/CMWH.S10444.
[42] R. E. Garfield and W. L. Maner, “Physiology and electrical activity of uterine contractions,” *Semin Cell Dev Biol*, vol. 18, no. 3, pp. 289–295, Jun. 2007, doi: 10.1016/j.semcdb.2007.05.004.
[43] Nemo HealthCare Website, “Nemo HealthCare.” https://nemohealthcare.com/en/ (accessed Jan. 18, 2023).
[44] GE Healthcare Website, “GE Healthcare,” 2017. https://www.gehealthcare.com/about/newsroom/press-releases/ge-healthcare-expands-digital-maternal-infant-care-offerings-acquisition-monica (accessed Jan. 18, 2023).
[45] Arnaldo Batista, “Uterine Electromyography Processing for Pregnancy Monitoring and Preterm Risk Evaluation.” https://sites.fct.unl.pt/sites/default/files/aspi/files/presentation_oxford_final_arnaldo.pdf (accessed Jan. 18, 2023).
[46] OB Tools Website, “OB Tools .” https://ob-tools.com/problems-with-external-monitors/ (accessed Jan. 18, 2023).
[47] H. Leman and C. Marque, “Rejection of the maternal electrocardiogram in the electrohysterogram signal,” *IEEE Trans Biomed Eng*, vol. 47, no. 8, pp. 1010–1017, 2000, doi: 10.1109/10.855927.
[48] M. Hassan, S. Boudaoud, J. Terrien, B. Karlsson, and C. Marque, “Combination of Canonical Correlation Analysis and Empirical Mode Decomposition Applied to Denoising the Labor Electrohysterogram,” *IEEE Trans Biomed Eng*, vol. 58, no. 9, pp. 2441–2447, Sep. 2011, doi: 10.1109/TBME.2011.2151861.
[49] G. Fele-Žorž, G. Kavšek, Ž. Novak-Antolič, and F. Jager, “A comparison of various linear and non-linear signal processing techniques to separate uterine EMG records of term and pre-term delivery groups,” *Med Biol Eng Comput*, vol. 46, no. 9, pp. 911–922, Sep. 2008, doi: 10.1007/s11517-008-0350-y.
[50] Z.-H. Zhou, *Machine Learning*. Singapore: Springer Singapore, 2021. doi: 10.1007/978-981-15-1967-3.
[51] IBM Cloud Education, “Machine Learning,” *IBM Website*, Jul. 15, 2020. https://www.ibm.com/cloud/learn/machine-learning (accessed Jul. 25, 2022).
[52] S Shalev-Shwartz and S Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. 2014.
[53] J. Alzubi, A. Nayyar, and A. Kumar, “Machine Learning from Theory to Algorithms: An Overview,” in *Journal of Physics: Conference Series*, Nov. 2018, vol. 1142, no. 1. doi: 10.1088/1742-6596/1142/1/012012.
[54] B. Liu, “Supervised Learning,” in *Web Data Mining*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 63–132. doi: 10.1007/978-3-642-19460-3_3.
[55] V. Nasteski, “An overview of the supervised machine learning methods,” *HORIZONS.B*, vol. 4, pp. 51–62, Dec. 2017, doi: 10.20544/HORIZONS.B.04.1.17.P05.
[56] S. Hanif, T. Ilyas, and M. Zeeshan, “Intrusion Detection In IoT Using Artificial Neural Networks On UNSW-15 Dataset,” in *2019 IEEE 16th International Conference on Smart Cities: Improving*

Quality of Life Using ICT & IoT and AI (HONET-ICT), Oct. 2019, pp. 152–156. doi: 10.1109/HONET.2019.8908122.
[57] Metehan Kozan, "Supervised and Unsupervised," *Medium Website (1 Sep, 2021)*. https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644 (accessed Sep. 12, 2022).
[58] O. Maimon and L. Rokach, Eds., *Data Mining and Knowledge Discovery Handbook*. New York: Springer-Verlag, 2005. doi: 10.1007/b107408.
[59] IBM Cloud Education, "What is a Decision Tree," *IBM Website, (2020)*. https://www.ibm.com/topics/decision-trees (accessed Sep. 12, 2022).
[60] R. STINE, "An Introduction to Bootstrap Methods," *Sociol Methods Res*, vol. 18, no. 2–3, pp. 243–291, Nov. 1989, doi: 10.1177/0049124189018002003.
[61] A. Verikas, E. Vaiciukynas, A. Gelzinis, J. Parker, and M. Olsson, "Electromyographic Patterns during Golf Swing: Activation Sequence Profiling and Prediction of Shot Effectiveness," *Sensors*, vol. 16, no. 4, p. 592, Apr. 2016, doi: 10.3390/s16040592.
[62] William S Noble, "What is a support vector machine?," vol. 24, no. 12. NATURE BIOTECHNOLOGY , 2006.
[63] IBM Documentation, "About SVM," *IBM Website,* 2021. https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-about-svm (accessed Sep. 15, 2022).
[64] Harry Zhang, "The Optimality of Naive Bayes."
[65] Scikit-learn, "Naive Bayes ," *Scikit-learn Website*, 2022. https://scikit-learn.org/stable/modules/naive_bayes.html (accessed Sep. 15, 2022).
[66] P. Cunningham and S. J. Delany, "k-Nearest Neighbour Classifiers - A Tutorial," *ACM Comput Surv*, vol. 54, no. 6, pp. 1–25, Jul. 2022, doi: 10.1145/3459665.
[67] N. Bhatia and Vandana, "Survey of Nearest Neighbor Techniques," Jul. 2010.
[68] Zoubin Ghahramani, "Unsupervised Learning," in *Advanced Lectures on Machine Learning LNAI 3176*, 2004.
[69] IBM Cloud Education, "What is Unsupervised Learning," *IBM Website*. https://www.ibm.com/topics/unsupervised-learning (accessed Sep. 19, 2022).
[70] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, Apr. 2016, doi: 10.1098/rsta.2015.0202.
[71] J. Alzubi, A. Nayyar, and A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," *J Phys Conf Ser*, vol. 1142, p. 012012, Nov. 2018, doi: 10.1088/1742-6596/1142/1/012012.
[72] C. Seiffert, T. M. Khoshgoftaar, J. van Hulse, and A. Napolitano, "RUSBoost: Improving classification performance when training data is skewed," in *2008 19th International Conference on Pattern Recognition*, Dec. 2008, pp. 1–4. doi: 10.1109/ICPR.2008.4761297.
[73] Geoffrey E. Hinton, Yee-Whye Teh, and Simon Osindero, "A Fast Learning Algorithm for Deep Belief Nets," in *Neural Computation* , vol. 18, 206AD, pp. 1527–1554.
[74] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput Sci*, vol. 2, no. 6, p. 420, Nov. 2021, doi: 10.1007/s42979-021-00815-1.
[75] R. E. Neapolitan and X. Jiang, *Artificial Intelligence*. Chapman and Hall/CRC, 2018. doi: 10.1201/b22400.
[76] A. Dertat, "Applied deep learning - part 1: Artificial neural networks," 2017. https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networksd7834f67a4f6 (accessed Oct. 10, 2022).
[77] M. Mahmud, M. S. Kaiser, T. M. McGinnity, and A. Hussain, "Deep Learning in Mining Biological Data," *Cognit Comput*, vol. 13, no. 1, pp. 1–33, Jan. 2021, doi: 10.1007/s12559-020-09773-x.
[78] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Single Layer & Multi-layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting," *Procedia Comput Sci*, vol. 135, pp. 89–98, 2018, doi: 10.1016/j.procs.2018.08.153.

[79] A. Eliasy and J. Przychodzen, “The role of AI in capital structure to enhance corporate funding strategies,” *Array*, vol. 6, p. 100017, Jul. 2020, doi: 10.1016/j.array.2020.100017.
[80] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” *Neural Comput*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/neco_a_01199.
[81] A. Baldominos, A. Cervantes, Y. Saez, and P. Isasi, “A Comparison of Machine Learning and Deep Learning Techniques for Activity Recognition using Mobile Devices,” *Sensors*, vol. 19, no. 3, p. 521, Jan. 2019, doi: 10.3390/s19030521.
[82] Q. Ma, M. Wang, L. Hu, L. Zhang, and Z. Hua, “A Novel Recurrent Neural Network to Classify EEG Signals for Customers’ Decision-Making Behavior Prediction in Brand Extension Scenario,” *Front Hum Neurosci*, vol. 15, Mar. 2021, doi: 10.3389/fnhum.2021.610890.
[83] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Trans Knowl Data Eng*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.
[84] Foster Provost, “Machine Learning from Imbalanced Data Sets 101.”
[85] M. Renz, C. Shahabi, X. Zhou, and M. A. Cheema, Eds., *Database Systems for Advanced Applications*, vol. 9050. Cham: Springer International Publishing, 2015. doi: 10.1007/978-3-319-18123-3.
[86] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, Jun. 2008, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.
[87] MathWorks, “Feature Extraction.” https://www.mathworks.com/discovery/feature-extraction.html (accessed Sep. 17, 2022).
[88] Ervin Sejdic, Igor Djurovic, and Jin Jiang, “Time-Frequency Feature Representation Using Energy Concentration: An Overview of Recent Advances,” in *Digital Signal Processing*, vol. 19, 2019.
[89] J. O. M. Solomon, “PSD computations using Welch’s method. [Power Spectral Density (PSD)],” Albuquerque, NM, and Livermore, CA (United States), Dec. 1991. doi: 10.2172/5688766.
[90] Hangfang Zhao and Lin Gui, “Nonparametric and parametric methods of spectral analysis,” *MATEC Web of Conferences*, 2019.
[91] P. Welch, “The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms,” *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, Jun. 1967, doi: 10.1109/TAU.1967.1161901.
[92] K. K. Parhi and M. Ayinala, “Low-Complexity Welch Power Spectral Density Computation,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 1, pp. 172–182, Jan. 2014, doi: 10.1109/TCSI.2013.2264711.
[93] Haibo He and Yunqian Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*. 2013.
[94] MathWorks, “fscmrmr,” *MathWorks Website*. https://www.mathworks.com/help/stats/fscmrmr.html#mw_733b9b36-11f2-4aa2-85fc-0988c425cd95_head (accessed Nov. 20, 2022).
[95] Yoshua Bengio and James Bergstra, “Random Search for Hyper-Parameter Optimization,” *Journal of Machine Learning Research* , vol. 13, pp. 281–305, 2012.
[96] MathWorks, “Bayesian Optimization Algorithm,” *MathWorks website*, 2022. https://www.mathworks.com/help/stats/bayesian-optimization-algorithm.html (accessed Nov. 01, 2022).
[97] E. Brochu, V. M. Cora, and N. de Freitas, “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning,” Dec. 2010.