EARLY DIAGNOSIS AND PREDICTION OF FETAL ABNORMALITIES USING MACHINE LEARNING

[1]P. Sumanth, [2]A. Vijay, [3]D. Nithin Aravindh, [4]Dr. G. Rosline Nesakumari

[1,2,3]student, [4]Assistant Professor,
Department of Computer Science and Engineering,
Bharath Institute of Higher Education and Research,

Abstract- Normal fetal growth is a critical component of a healthy pregnancy and influences the long-term health of the offspring. However, defining normal and abnormal fetal growth has been a long-standing challenge in clinical practice and research. The authors review various references and standards that are widely used to evaluate fetal growth, and discuss common pitfalls of current definitions of abnormal fetal growth. Pros and cons of different approaches to customize fetal growth standards are described. The authors further discuss recent advances towards an integrated definition for fetal growth restriction. Such a definition may incorporate fetal size with the status of placental health measured by maternal and fetal Doppler velocimetry and biomarkers, biophysical findings and genetics. Although the concept of an integrated definition appears promising, further development and testing are required. An improved definition of abnormal fetal growth should benefit both research and clinical practice.

1.INTRODUCTION

Normal fetal growth is a critical component of a healthy pregnancy and influences the long-term health of the offspring. Common adult diseases such as type 2 diabetes and cardiovascular conditions have been linked to abnormal fetal growth, particularly fetal growth restriction (FGR).However, the latter has not been clearly defined. Therefore, an objective assessment of normal and abnormal fetal growth has enormous utility in prenatal and neonatal care and outcome-based research. The purpose of this review is to summarize literature on the definition of abnormal fetal growth that go beyond simple fetal size

1.1 PROPOSED ALGORITHMS

RANDOM FOREST ALGORITHM

Random forest algorithm can use both for classification and the regression kind of problems. In this you are going to learn, how the **random forest algorithm** works in machine learning for the classification task.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The below diagram explains the working of the Random Forest algorithm:

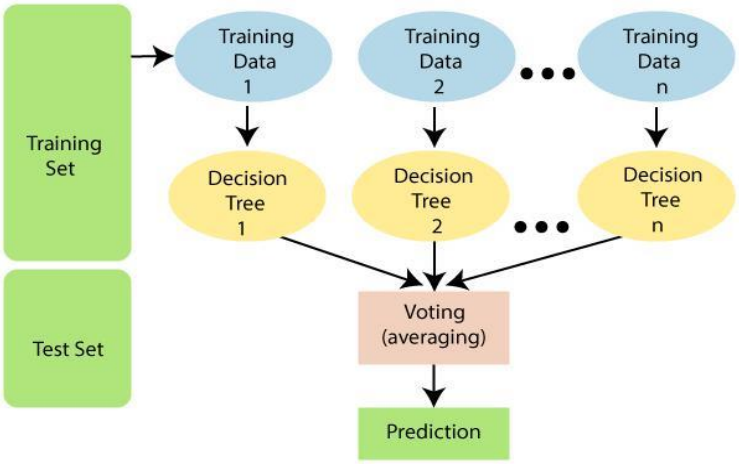


Fig 2: Explaining the working algorithm of the

Random Forest algorithm

Below are some points that explain why we should use the Random Forest algorithm:

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Features of a Random Forest Algorithm

- It's more accurate than the decision tree algorithm.
- It provides an effective way of handling missing data.
- It can produce a reasonable prediction without hyper-parameter tuning.
- It solves the issue of overfitting in decision trees.
- In every random forest tree, a subset of features is selected randomly at the node's splitting point.

Classification in random forests

Classification in random forests employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes.

A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. The diagram below shows a simple random forest classifier.

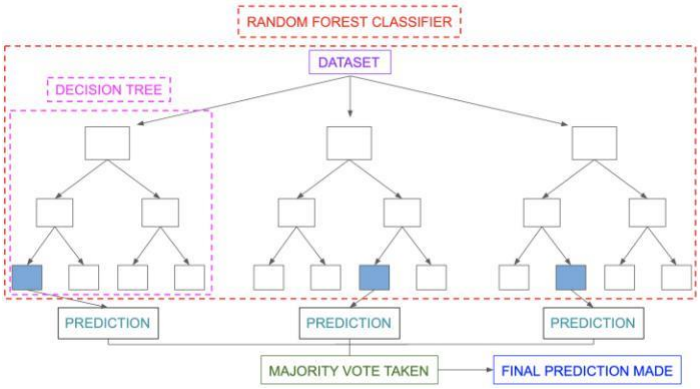


Fig 3: Explaining the Random Forest Classifier

Random Forest Steps

1. Randomly select "k" features from total "m" features.
2. Where k << m
3. Among the "k" features, calculate the node "d" using the best split point.
4. Split the node into daughter nodes using the best split.
5. Repeat 1 to 3 steps until "l" number of nodes has been reached.
6. Build forest by repeating steps 1 to 4 for "n" number times to create "n" number of trees.

The beginning of random forest algorithm starts with randomly selecting "k" features out of total "m" features. In the image, you can observe that we are randomly taking features and observations.
The working of the algorithm can be better understood by the below example:

Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:

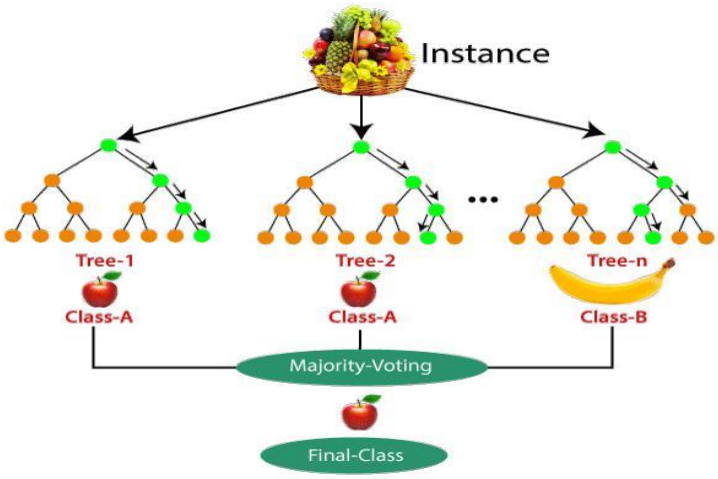


Fig 4: Explaining the Random Forest Classifier

algorithm with example

APPLICATIONS OF RANDOM FOREST

There are mainly four sectors where Random forest mostly used:

1. **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
2. **Medicine:** With the help of this algorithm, disease trends and risks of the disease can be identified.
3. **Land Use:** We can identify the areas of similar land use by this algorithm.
4. **Marketing:** Marketing trends can be identified using this algorithm.

Advantages of Random Forest

Random Forest is capable of performing both Classification and Regression tasks.

- It is capable of handling large datasets with high dimensionality.
- It enhances the accuracy of the model and prevents the overfitting issue.

Disadvantages of Random Forest

- Although random forest can be used for both classification and regression tasks, it is not more suitable for Regression tasks.

KNN ALGORITHMS

o K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

o K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

o K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

o K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

o K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

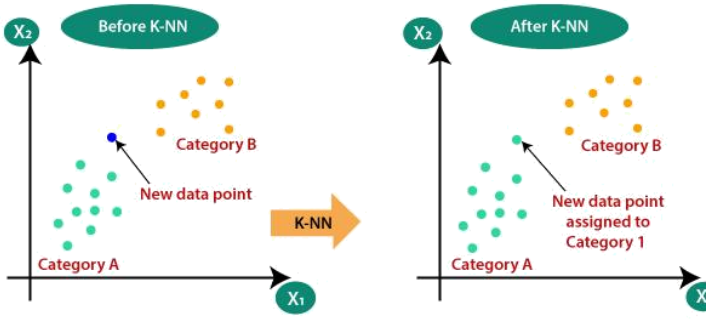
o It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

o KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

o **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

Why do we need a K-NN Algorithm?

Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

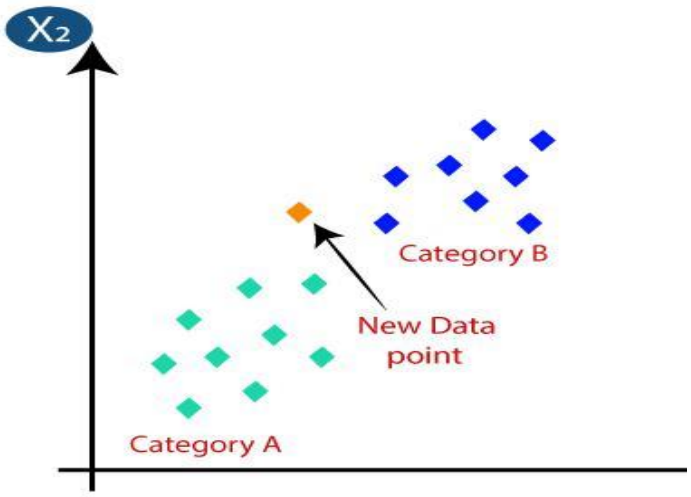


How does K-NN work?

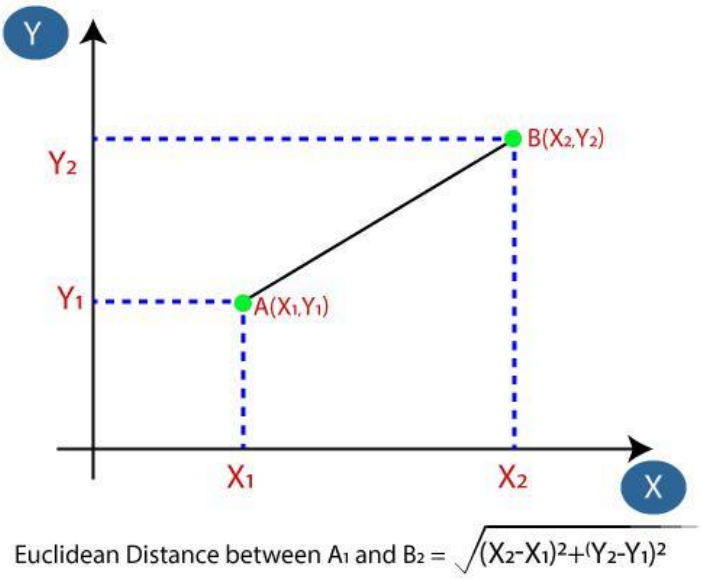
The K-NN working can be explained on the basis of the below algorithm:

- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:



- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



o By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

How to select the value of K in the K-NN Algorithm?

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

Advantages of KNN Algorithm:

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

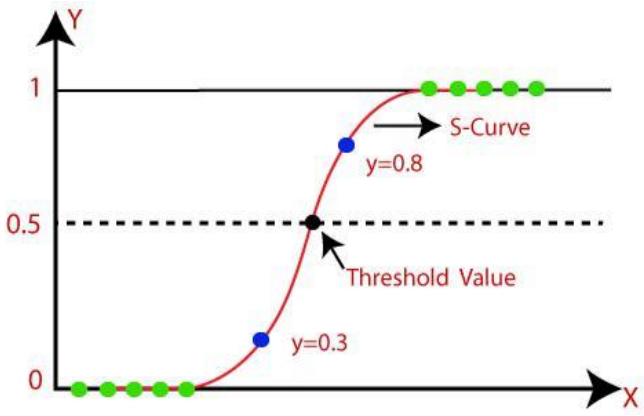
Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

LOGISTIC REGRESSION

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Note: ***Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.***

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \cdots + b_n x_n$$

Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + \cdots$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

- **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"

- **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

CHAPTER 2

LITERATURE REVIEW

Evidence-based national guidelines for the management of suspected fetal growth restriction

The aim of this review is to: summarize areas of consensus and controversy between recently published national guidelines on small for gestational age or fetal growth restriction; highlight any recent evidence that should be incorporated into existing guidelines; and identify future research priorities in this field. A search of MEDLINE, Google, and the International Guideline Library identified 6 national guidelines on management of pregnancies complicated by fetal growth restriction/small for gestational age published from 2010 onwards. There is general consensus between guidelines (at least 4 of 6 guidelines in agreement) in early pregnancy risk selection, and use of low-dose aspirin for women with major risk factors for placental insufficiency. All highlight the importance of smoking cessation to prevent small for gestational age. While there is consensus in recommending fundal height measurement in the third trimester, 3 specify the use of a customized growth chart, while 2 recommend McDonald rule. Routine third-trimester scanning is not recommended for small-for-gestational-age screening, while women with major risk factors should have serial scanning in the third trimester. Umbilical artery Doppler studies in suspected small-for-gestational-age pregnancies are

universally advised, however there is inconsistency in the recommended frequency for growth scans after diagnosis of small for gestational age/fetal growth restriction (2-4 weekly). In late-onset fetal growth restriction (≥32 weeks) general consensus is to use cerebral Doppler studies to influence surveillance and/or delivery timing. Fetal surveillance methods

(most recommend cardiotocography) and recommended timing of delivery vary. There is universal agreement on the use of corticosteroids before birth at <34 weeks, and general consensus on the use of magnesium sulfate for neuroprotection in early-onset fetal growth restriction (<32 weeks). Most

guidelines advise using cardiotocography surveillance to plan delivery in fetal growth restriction <32 weeks. The recommended gestation at delivery for fetal growth restriction with absent and reversed end-diastolic velocity varies from 32 to ≥34 weeks and 30 to ≥34 weeks, respectively. Overall, where there is high-quality evidence from randomized controlled trials and meta-analyses, eg, use of umbilical artery Doppler and corticosteroids for delivery <34 weeks, there is a high degree of consistency between national small-for-gestational-age guidelines. This review discusses areas where there is potential for convergence between small-for-gestational-age guidelines based on existing randomized controlled trials of management of small-for-gestational-age pregnancies, and areas of controversy. Research priorities include assessing the utility of late third-trimester scanning to prevent major morbidity and mortality and to investigate the optimum timing of delivery in fetuses with late-onset fetal growth restriction and abnormal Doppler parameters. Prospective studies are needed to compare new international population ultrasound standards with those in current use.

Diagnosis and surveillance of late-onset fetal growth restriction

By consensus, late fetal growth restriction is that diagnosed >32 weeks. This condition is mildly associated with a higher risk of perinatal hypoxic events and suboptimal neurodevelopment. Histologically, it is characterized by the presence of uteroplacental vascular lesions (especially infarcts), although the incidence of such lesions is lower than in preterm fetal growth restriction. Screening procedures for fetal growth restriction need to identify small babies and then differentiate between those who are healthy and those who are pathologically small. First- or second-trimester screening strategies provide detection rates for late smallness for gestational age <50% for 10% of false positives. Compared to clinically indicated ultrasonography in the third trimester, universal screening triples the detection rate of late smallness for gestational age. As opposed to early third-trimester ultrasound, scanning late in pregnancy (around 37 weeks) increases the detection rate for birthweight <3rd centile. Contrary to early fetal growth restriction, umbilical artery Doppler velocimetry alone does not provide good differentiation between late smallness for gestational age and fetal growth restriction. A combination of biometric parameters (with severe smallness usually defined as estimated fetal weight or abdominal circumference <3rd centile) with Doppler criteria of placental insufficiency (either in the maternal [uterine Doppler] or fetal [cerebroplacental ratio] compartments) offers a classification tool that correlates with the risk for adverse perinatal outcome. There is no evidence that induction of late fetal growth restriction at term improves perinatal outcomes nor is it a cost-effective strategy, and it may increase neonatal admission when performed <38 weeks.

Diagnosis and management of fetal growth restriction

The purpose of this Consult is to outline an evidence-based, standardized approach for the prenatal diagnosis and management of fetal growth restriction. The recommendations of the Society for Maternal-Fetal Medicine are as follows: (1) we recommend that fetal growth restriction be defined as an ultrasonographic estimated fetal weight or abdominal circumference below the 10th percentile for gestational age (GRADE 1B);

> (2) we recommend the use of population-based fetal growth references (such as Hadlock) in determining fetal weight percentiles (GRADE 1B); (3) we recommend against the use of low-molecular-weight heparin for the sole indication of prevention of recurrent fetal growth restriction (GRADE 1B); (4) we recommend against the use of sildenafil or activity restriction for in utero treatment of fetal growth restriction (GRADE 1B); (5) we recommend that a detailed obstetrical ultrasound examination (current procedural terminology code 76811) be performed with early-onset fetal growth restriction (<32 weeks of gestation) (GRADE 1B).

Growth charts and prediction of abnormal growth

— what is known, what is not known and what is misunderstood

Assessment of fetal growth has an important effect on perinatal morbidity and mortality. To understand what tool to choose best for a given population a basic knowledge of how growth charts are developed and used has to be acquired. For this reason, this literature review was performed. An extensive literature review aimed at identifying articles related to the development of growth assessment in both spectrums of abnormal fetal growth — large and small. The analyzed articles were chosen and presented to show both the historical aspects of growth assessment, current trends and future considerations.Identification of both large and small fetuses and neonates is equally crucial. Definitions and methodology vary worldwide and there is an ongoing discussion on the best tool to choose for a given population. An important part of the debate is how to differentiate between thephysiologically small fetus and the truly growth restricted fetus who is at risk of perinatal complication. Similarly, the diagnosis of a large fetus is important in prevention of perinatal complications and surgical deliveries. Many clinical settings still lack growth standards.Birthweight for gestational age charts are biased for weight in preterm birth. Prediction and management of outcome cannot be based solely on fetal size. Small is not the only problem, we have to think large as well. A common misunderstanding in clinical practice is not using uniform charts in defining growth.

An outcome-based approach for the creation of fetal growth standards: do singletons and twins need separate standards
Contemporary fetal growth standards are created by using theoretical properties (percentiles) of birth weight (for gestational age) distributions. The authors used a clinically relevant, outcome-based methodology to determine if separate fetal growth standards are required for singletons and twins. All singleton and twin livebirths between 36 and 42 weeks' gestation in the United States (1995-2002) were included, after exclusions for missing information and other factors (n = 17,811,922). A birth weight range was identified, at each gestational age, over which serious neonatal morbidity and neonatal mortality rates were lowest. Among singleton males at 40 weeks, serious neonatal morbidity/mortality rates were lowest between 3,012 g (95% confidence interval (CI): 3,008, 3,018) and 3,978 g (95% CI: 3,976, 3,980). The low end of this optimal birth weight range for females was 37 g (95% CI: 21, 53) less. The low optimal birth weight was 152 g (95% CI: 121, 183) less for twins compared with singletons. No differences were observed in low optimal birth weight by period (1999-2002 vs. 1995-1998), but small differences were observed for maternal education, race, parity, age, and smoking status. Patterns of birth weight-specific serious neonatal morbidity/neonatal mortality support the need for plurality-specific fetal growth standards.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

System : Pentium i3 Processor
Hard Disk : 500 GB.
Monitor : 15'' LED
Input Devices : Keyboard, Mouse
Ram :2GB

3.2 SOFTWARE REQUIREMENTS

Operating system : Windows 10
Coding Language : Python

3.3 LANGUAGE SPECIFICATION

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This **tutorial** gives enough understanding on **Python programming** language. 3.4. HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.5. APPLICATION OF PYTHON

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **ExteZdable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.

- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

3.6 FEATURES OF PYTHON

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

3.7 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

3.7.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.7.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.7.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

CHAPTER 4

SYSTEM ANALYSIS

4.1 PURPOSE

The purpose of this document is approach of early diagnosis and prediction of fetal abnormalities using machine learning algorithms. In detail, this document will provide a general description of our project, including user requirements, product perspective, and overview of requirements, general constraints. In addition, it will also provide the specific requirements and functionality needed for this project - such as interface, functional requirements and performance requirements.

4.2 SCOPE

The scope of this SRSdocument persists for the entire life cycle of the project. This document defines the final state of the software requirements agreed upon by the customers and designers. Finally at the end of the project execution all the functionalities may be traceable from the SRSto the product. The document describes the functionality, performance, constraints, interface and reliability for the entire life cycle of the project.

4.3 EXISTING SYSTEM

In Existing,a method using customized birthweight norms that incorporated information about fetal growth potential. Based on the premise that birthweight varies with maternal and fetal physiological parameters (e.g., race/ethnicity, parity, sex, prepregnancy or early pregnancy body mass), they defined a new methodology to calculate optimal fetal weight at each gestational week customized by individual profile. The authors combined birthweight data at 40 weeks of gestation with estimated fetal weight based on the Hadlock estimated fetal weight curve.The ultrasound EFW curves were proportionately adjusted upwards or downwards according to the birthweight at 40 weeks for specific maternal and fetal profiles.

4.4 PROPOSED SYSTEM

In proposed system, discuss with recent advances towards an integrated definition for fetal growth restriction. Such a definition may incorporate fetal size with the status of placental health measured by maternal and fetal Doppler velocimetry and biomarkers, biophysical findings and genetics. Although the concept of an integrated definition appears promising, further development and testing are required. An improved definition of abnormal fetal growth should benefit both research and clinical practice.

CHAPTER 5

SYSTEM DESIGN

5.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

5.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action

5.3 DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity thatinteracts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

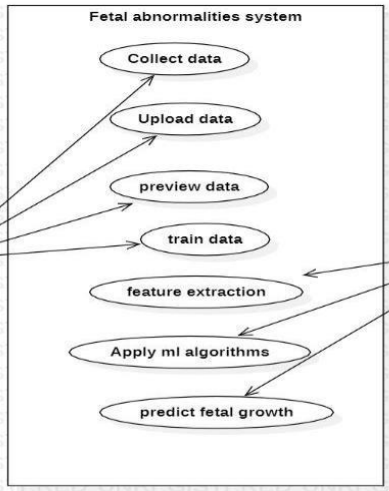
The UML represents a collection of best engineering practices that have proven successful int he modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

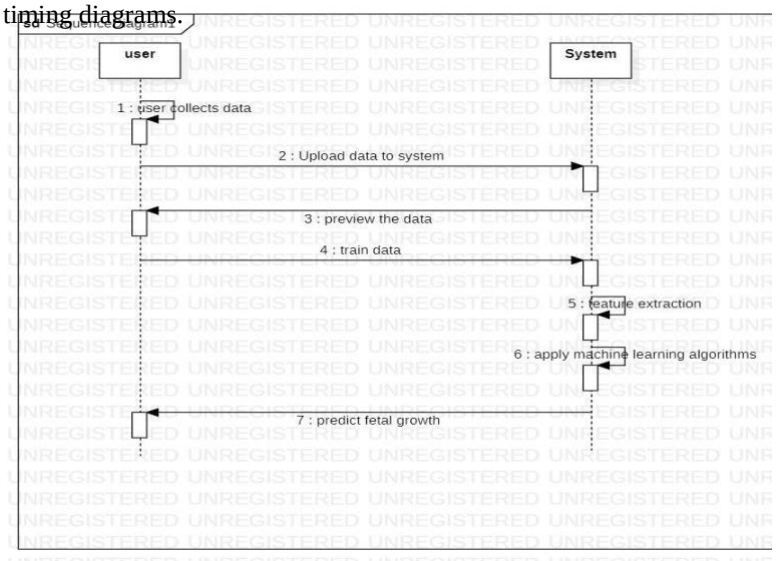
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Intzgrate best practices.



SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

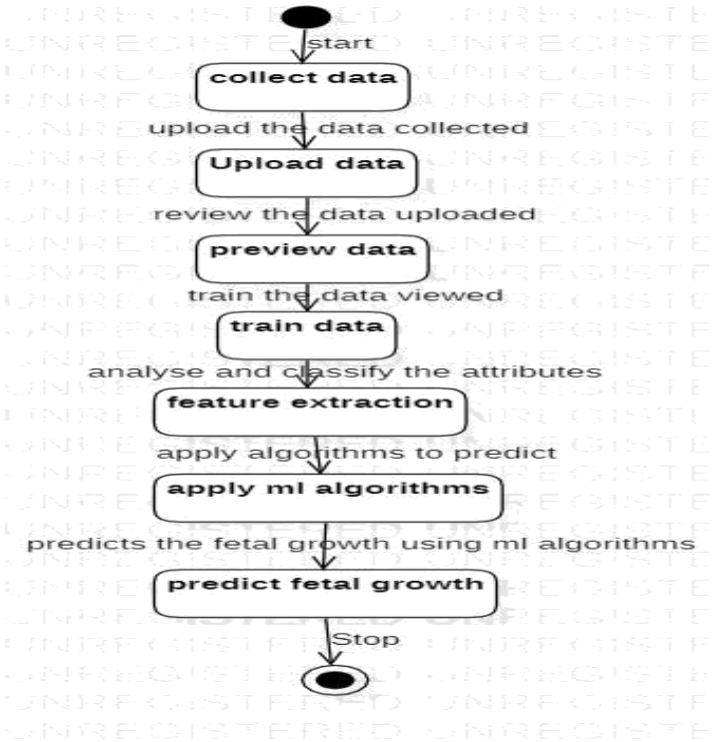


USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



CHAPTER 6
MODULES

MODULES

- Data Collection Module
- Analysis Module
- Feature Extraction
- Predict Fetal Growth

6.1 Data Collection Module

These guidelines represent a desirable standard for thecollectionofdataonavailability following immunization to allow for comparability of data, and are recommended as an addition to data collected for the specific study question and setting. The guidelines are not intended to guide the primary reporting of FGR to a surveillance system or study monitor. Investigators developing a data collection tool based on these data collection guidelines also need to refer to the criteria in the case definition, which are not repeated in these guidelines.

6.2 Analysis Module

In this module, we can analyse the fetal growth level and give patient feedbacks if it is life threatening or not.

6.3 Feature Extraction

This paper presents a method for feature extraction and classification of Fetal Growth between pregnancies based on learning machine Technique.

6.4 Predict Fetal Growth

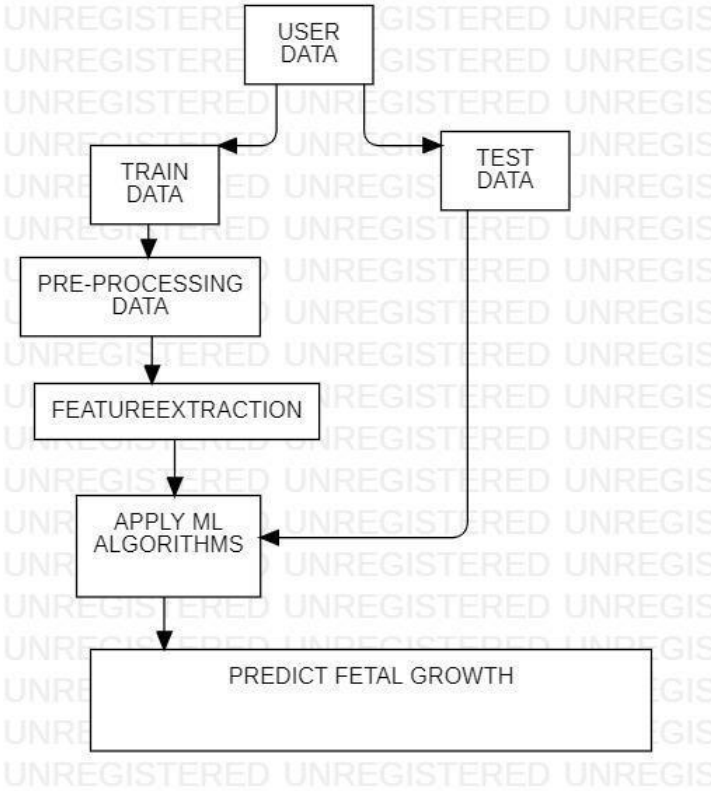
In this Module, Normal fetal growth is a critical component of a healthy pregnancy and influences the long-term health of the offspring. However, defining normal and abnormal fetal growth has been a long- standing challenge in clinical practice and research.

An improved definition of abnormal fetal growth should benefit both research and clinical practice.

CHAPTER 7

SYSTEM ARCHITECTURE

The system architectural design is the design process for identifying the subsystems making up the system and framework for subsystem control and communication. The goal of the architectural design is to establish the overall structure of software system.



CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test
Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid

input must be rejected.

Functions : identified functions must be

exercised.

Output : identified classes of application

outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test
System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing
White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing
Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:
Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach
Field testing will be performed manually and functional tests will be written in detail.

Test objectives
- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing
Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

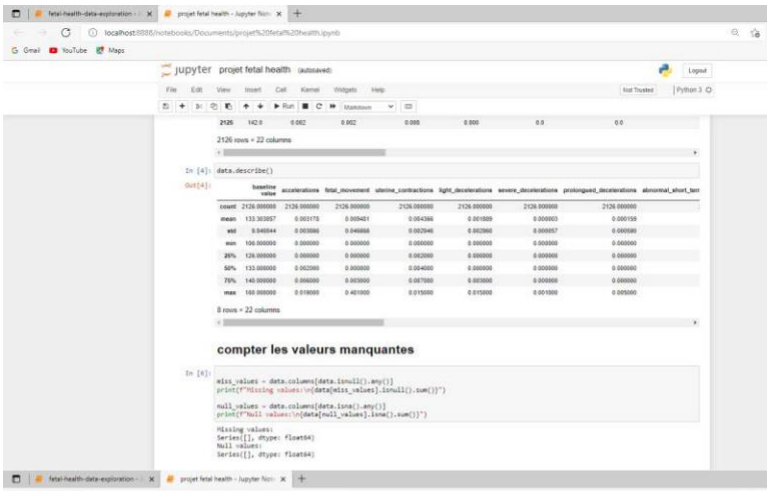
The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

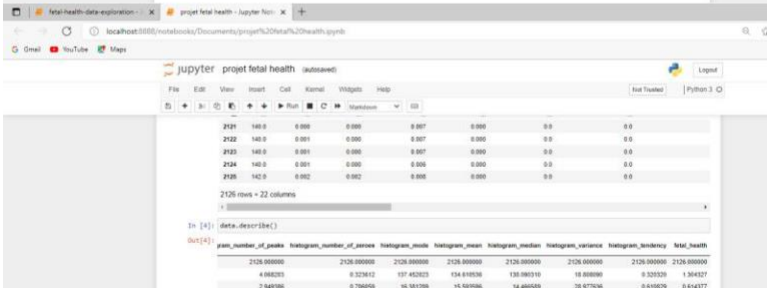
Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

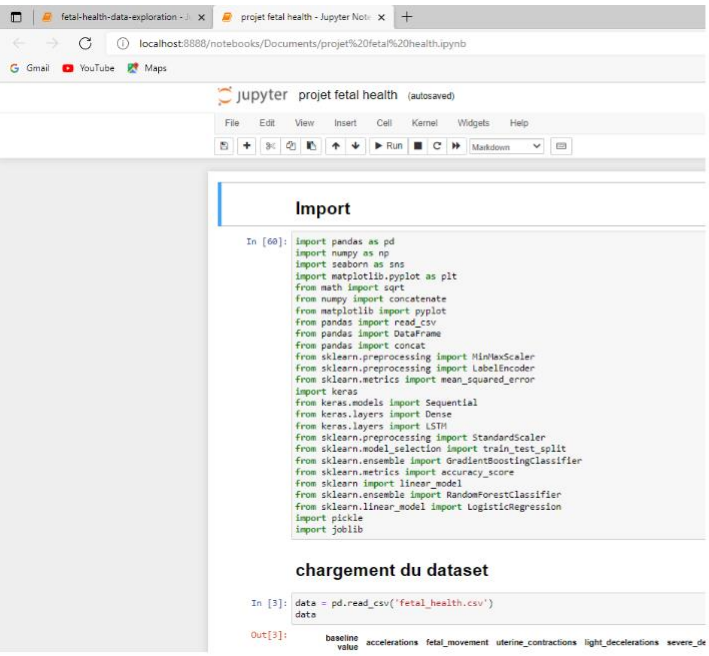
CHAPTER 9
CONCLUSION

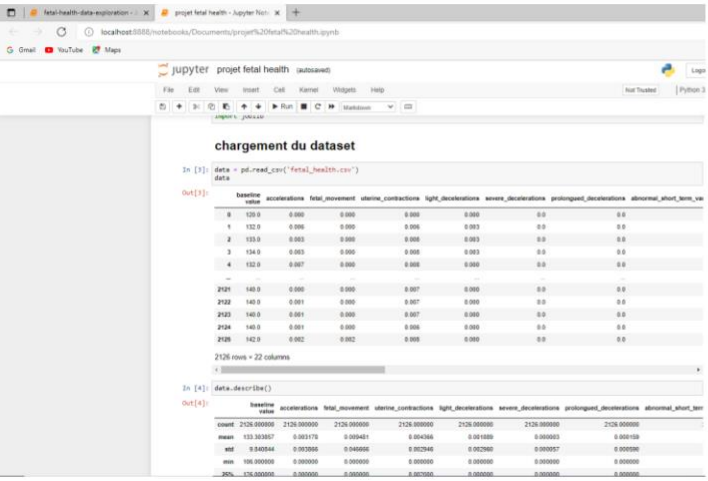
Fetal Health denotes the health and growth of the fetal and frequent contacts in the uterus of the pregnant women during pregnancy. Maximum pregnancy period complexities leads fetal to a severe difficulty which limits right growth that causes deficiency or death. Harmless pregnancy period by predicting the risk levels before the occasion of difficulties boost right fetal growth. This paper presented the various approaches and researches done so far for forecasting the fetal health and growth state from a set of pre-classified patterns knowledge with its accuracy. Predication of fetal health is vital in developing a predictive classifier model using Machine Learning Algorithms.

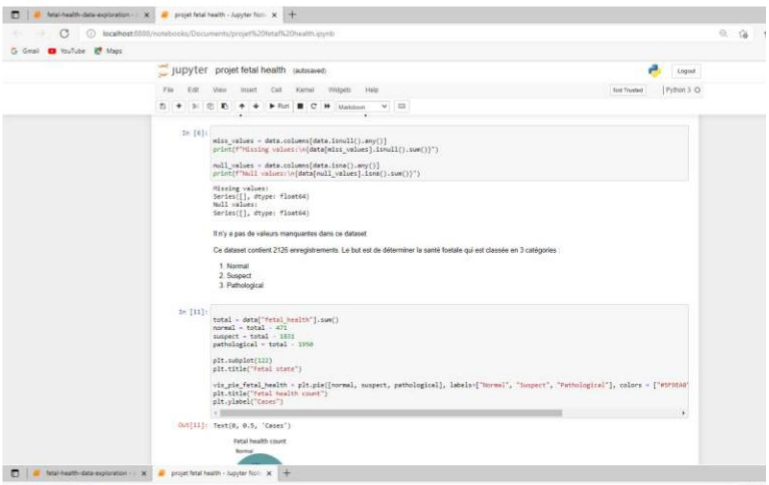


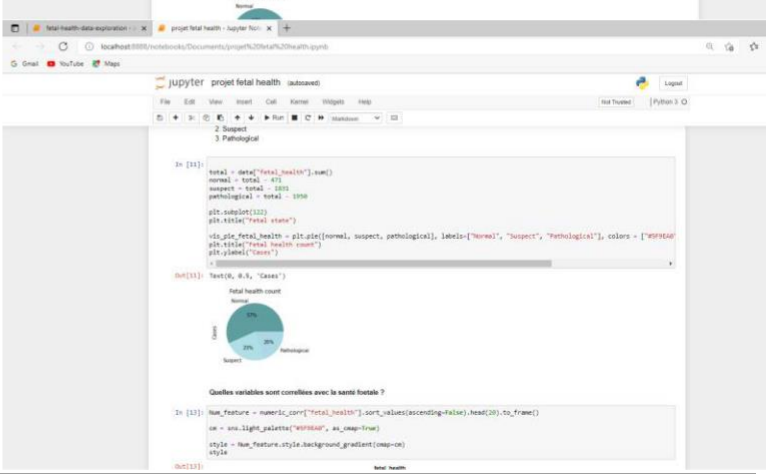


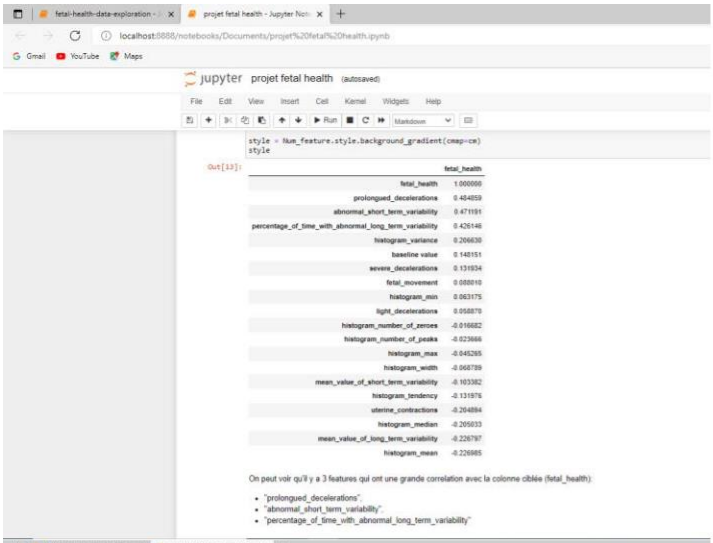
SCREENSHOTS

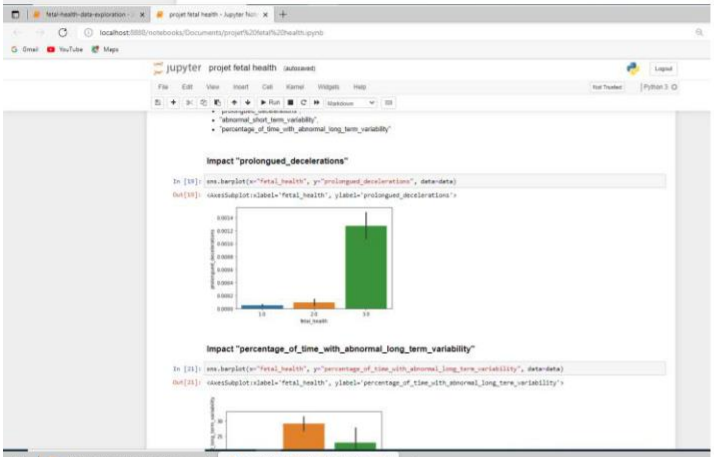


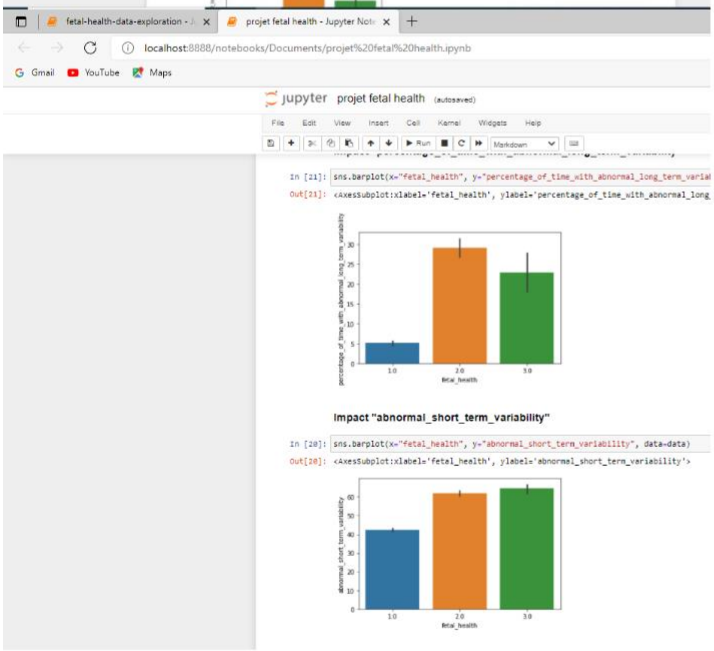


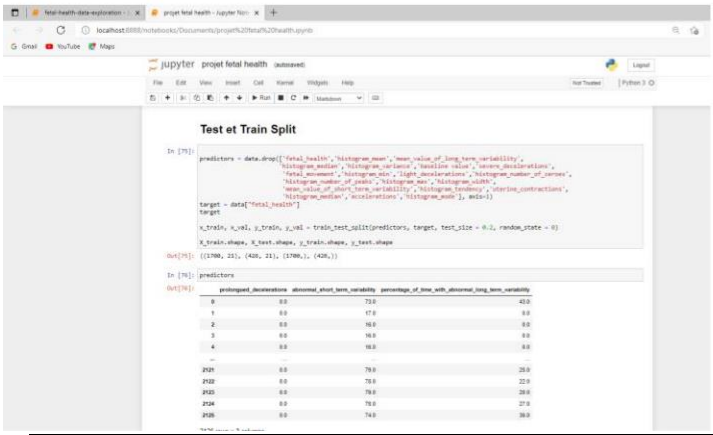


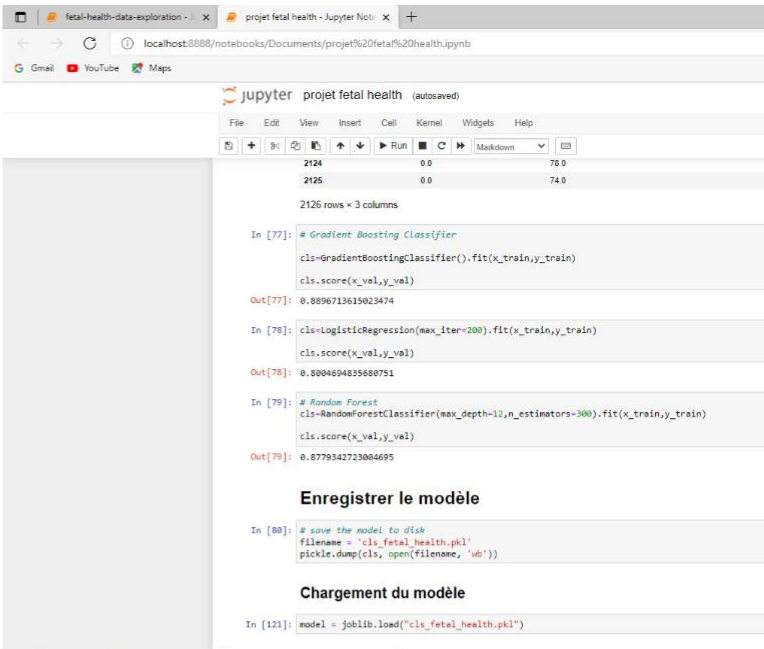


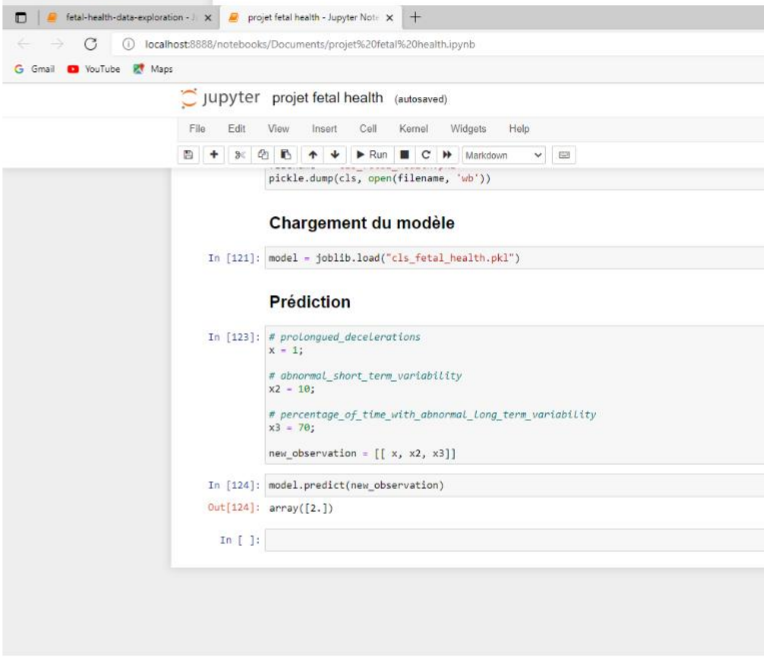

jupyter projet fetal health (autosaved)
style = Num_feature.style.background_gradient(cmap=cm)
style
Out[13]:
fetal_health
fetal_health 1.000000
prolongued_decelerations 0.484059
abnormal_short_term_variability 0.471191
percentage_of_time_with_abnormal_long_term_variability 0.426146
histogram_variance 0.266630
baseline value 0.148151
severe_decelerations 0.131934
fetal_movement 0.088910
histogram_min 0.063175
light_decelerations 0.058870
histogram_number_of_zeroes -0.016682
histogram_number_of_peaks -0.023666
histogram_max -0.045265
histogram_width -0.068789
mean_value_of_short_term_variability -0.103382
histogram_tendency -0.131976
uterine_contractions -0.204894
histogram_median -0.205033
mean_value_of_long_term_variability -0.226797
histogram_mean -0.226985
On peut voir qu'il y a 3 features qui ont une grande correlation avec la colonne ciblée (fetal_health):
"prolongued_decelerations",
"abnormal_short_term_variability",
"percentage_of_time_with_abnormal_long_term_variability"

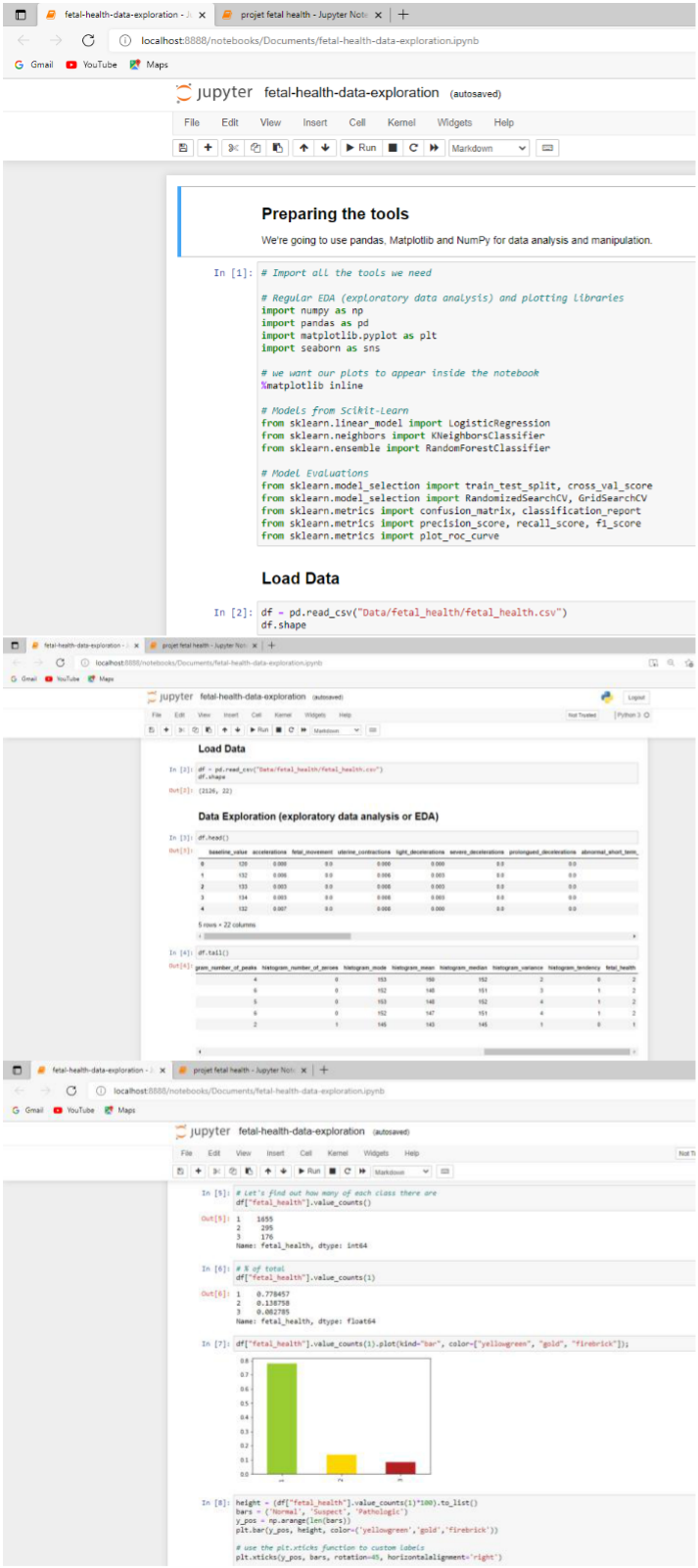

jupyter projet fetal health (autosaved)
Impact "prolongued_decelerations"
In [19]: sns.barplot(x="fetal_health", y="prolongued_decelerations", data=data)
Out[19]: <AxesSubplot:xlabel='fetal_health', ylabel='prolongued_decelerations'>
Impact "percentage_of_time_with_abnormal_long_term_variability"
In [21]: sns.barplot(x="fetal_health", y="percentage_of_time_with_abnormal_long_term_variability", data=data)
Out[21]: <AxesSubplot:xlabel='fetal_health', ylabel='percentage_of_time_with_abnormal_long_term_variability'>

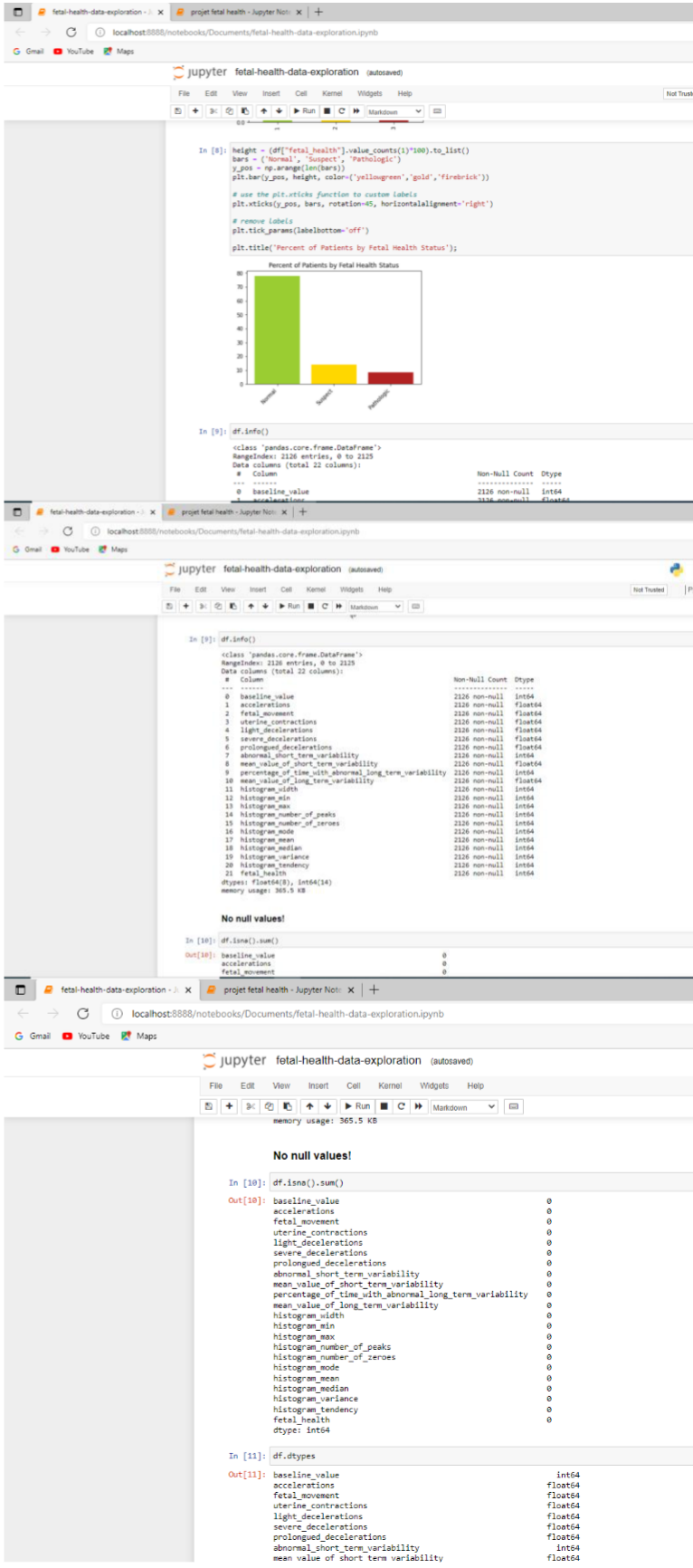

jupyter projet fetal health (autosaved)
In [21]: sns.barplot(x="fetal_health", y="percentage_of_time_with_abnormal_long_term_variability", data=data)
Out[21]: <AxesSubplot:xlabel='fetal_health', ylabel='percentage_of_time_with_abnormal_long
Impact "abnormal_short_term_variability"
In [20]: sns.barplot(x="fetal_health", y="abnormal_short_term_variability", data=data)
Out[20]: <AxesSubplot:xlabel='fetal_health', ylabel='abnormal_short_term_variability'>

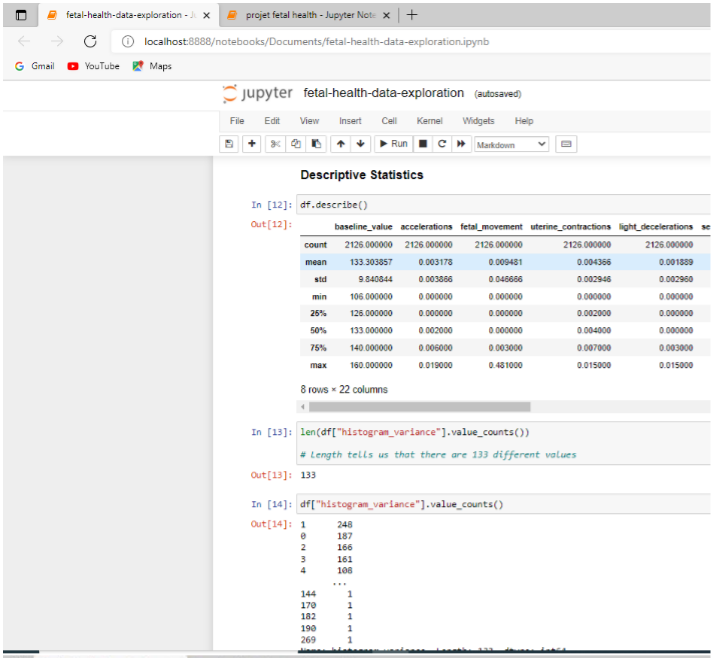

jupyter projet fetal health (autosaved)
Test et Train Split
In [75]:
predictors = data.drop(['fetal_health','histogram_mean','mean_value_of_long_term_variability',
'histogram_median','histogram_variance','baseline value','severe_decelerations',
'fetal_movement','histogram_min','light_decelerations','histogram_number_of_zeroes',
'histogram_number_of_peaks','histogram_max','histogram_width',
'mean_value_of_short_term_variability','histogram_tendency','uterine_contractions',
'histogram_median','accelerations','histogram_mode'], axis=1)
target = data["fetal_health"]
target
x_train, x_val, y_train, y_val = train_test_split(predictors, target, test_size = 0.2, random_state = 0)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
Out[75]: ((1700, 21), (426, 21), (1700,), (426,))
In [76]: predictors
Out[76]:
prolongued_decelerations abnormal_short_term_variability percentage_of_time_with_abnormal_long_term_variability
0 0.0 73.0 43.0
1 0.0 17.0 0.0
2 0.0 16.0 0.0
3 0.0 16.0 0.0
4 0.0 16.0 0.0
2121 0.0 79.0 25.0
2122 0.0 78.0 22.0
2123 0.0 78.0 26.0
2124 0.0 78.0 27.0
2125 0.0 74.0 36.0

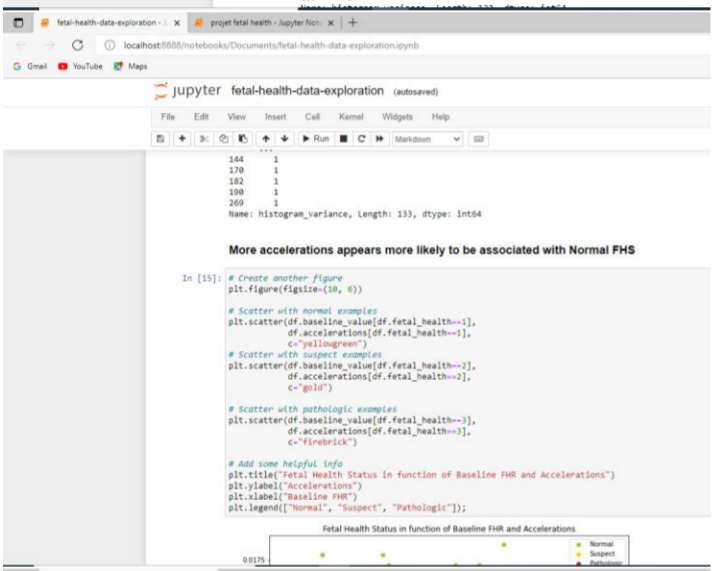

jupyter projet fetal health (autosaved)
2124 0.0 78.0
2125 0.0 74.0
2126 rows × 3 columns
In [77]: # Gradient Boosting Classifier
cls=GradientBoostingClassifier().fit(x_train,y_train)
cls.score(x_val,y_val)
Out[77]: 0.8896713615023474
In [78]: cls=LogisticRegression(max_iter=200).fit(x_train,y_train)
cls.score(x_val,y_val)
Out[78]: 0.8004694835680751
In [79]: # Random Forest
cls=RandomForestClassifier(max_depth=12,n_estimators=300).fit(x_train,y_train)
cls.score(x_val,y_val)
Out[79]: 0.8779342723004695
Enregistrer le modèle
In [80]: # save the model to disk
filename = 'cls_fetal_health.pkl'
pickle.dump(cls, open(filename, 'wb'))
Chargement du modèle
In [121]: model = joblib.load("cls_fetal_health.pkl")

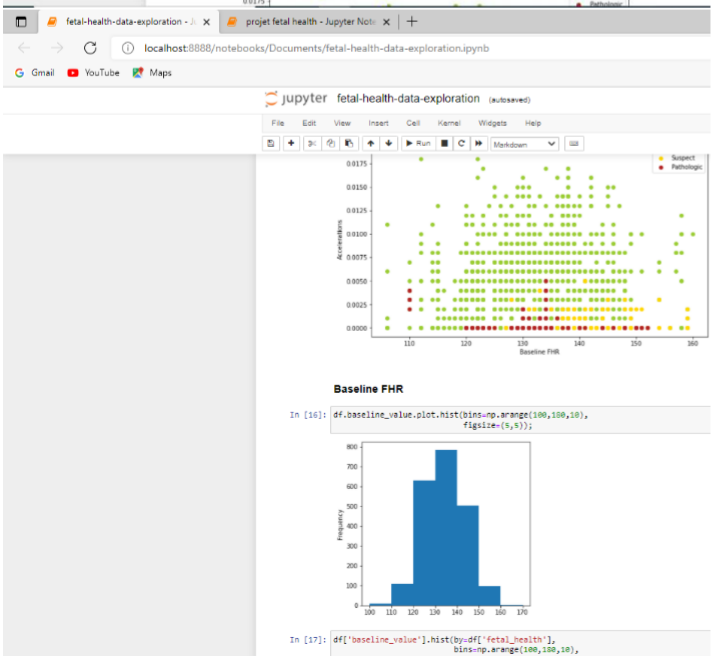

jupyter projet fetal health (autosaved)
pickle.dump(cls, open(filename, 'wb'))
Chargement du modèle
In [121]: model = joblib.load("cls_fetal_health.pkl")
Prédiction
In [123]: # prolongued_decelerations
x = 1;
# abnormal_short_term_variability
x2 = 10;
# percentage_of_time_with_abnormal_long_term_variability
x3 = 70;
new_observation = [[x, x2, x3]]
In [124]: model.predict(new_observation)
Out[124]: array([2.])
In [ ]:

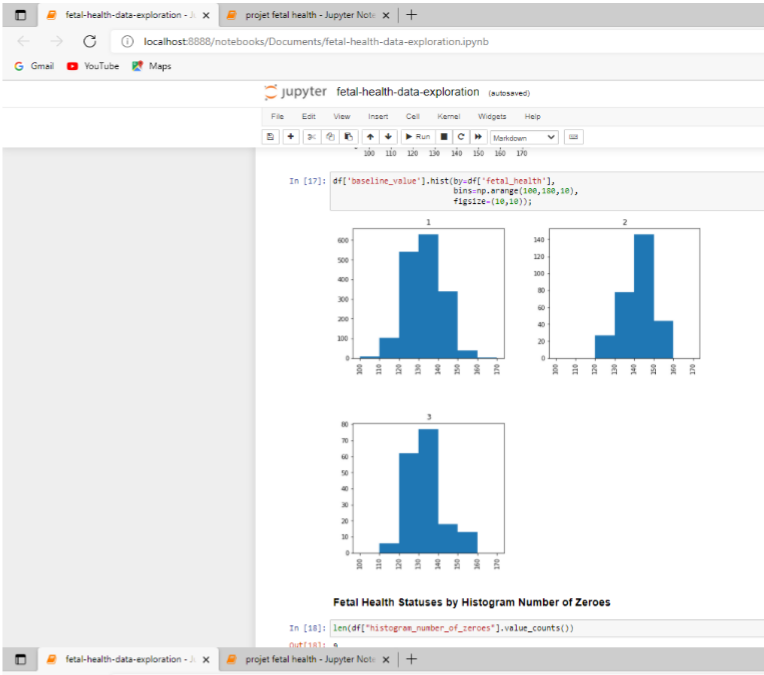

jupyter fetal-health-data-exploration (autosaved)
Preparing the tools
We're going to use pandas, Matplotlib and NumPy for data analysis and manipulation.
Load Data
Data Exploration (exploratory data analysis or EDA)

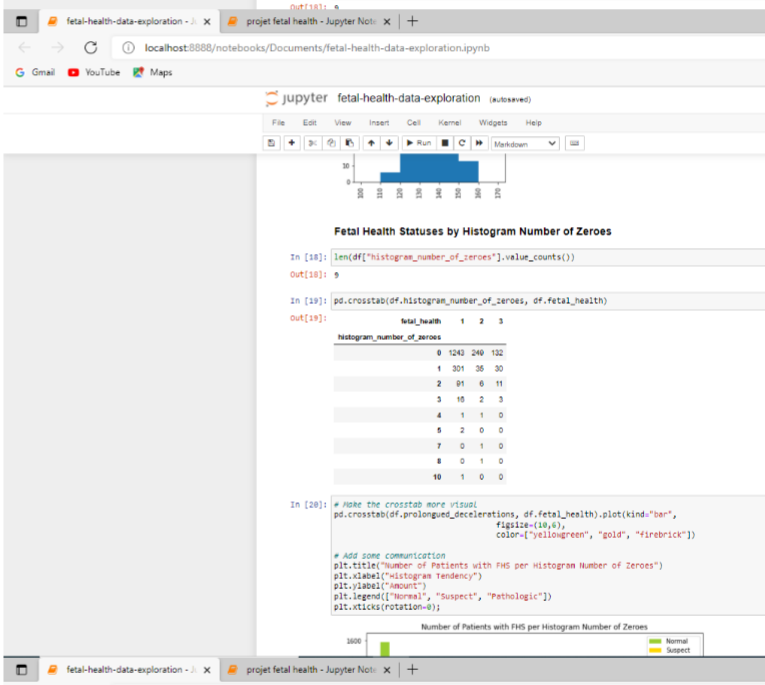

jupyter fetal-health-data-exploration (autosaved)
No null values!

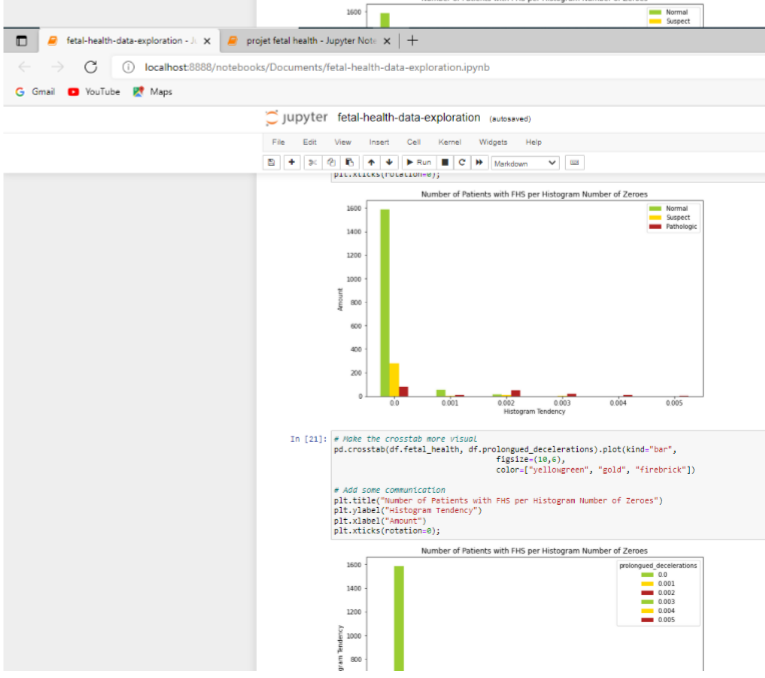

jupyter fetal-health-data-exploration (autosaved)
Descriptive Statistics
In [12]: df.describe()
Out[12]:
baseline_value accelerations fetal_movement uterine_contractions light_decelerations
count 2126.000000 2126.000000 2126.000000 2126.000000 2126.000000
mean 133.303857 0.003178 0.009481 0.004366 0.001889
std 9.840844 0.003866 0.046666 0.002946 0.002960
min 106.000000 0.000000 0.000000 0.000000 0.000000
25% 126.000000 0.000000 0.000000 0.002000 0.000000
50% 133.000000 0.002000 0.000000 0.004000 0.000000
75% 140.000000 0.006000 0.003000 0.007000 0.003000
max 160.000000 0.019000 0.481000 0.015000 0.015000
8 rows × 22 columns
In [13]: len(df["histogram_variance"].value_counts())
Length tells us that there are 133 different values
Out[13]: 133
In [14]: df["histogram_variance"].value_counts()
Out[14]: 1 248
0 187
2 166
3 161
4 108
144 1
170 1
182 1
190 1
269 1

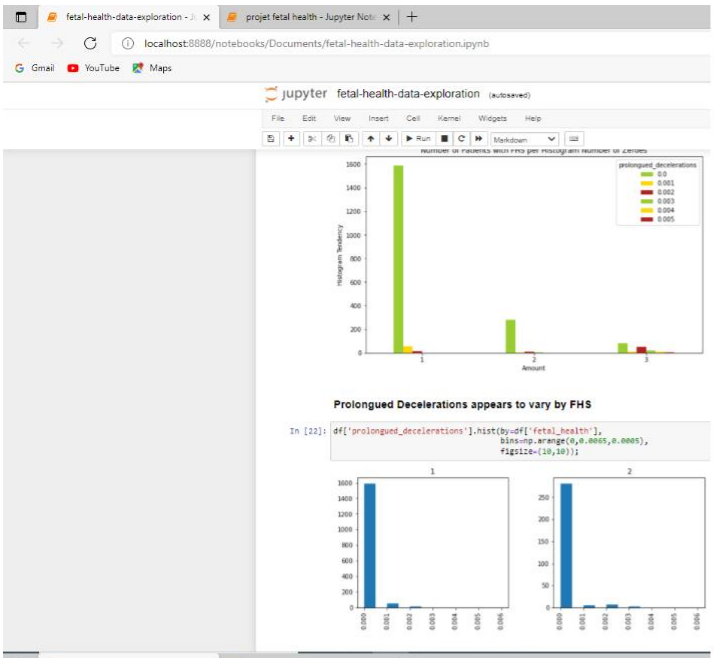

jupyter fetal-health-data-exploration (autosaved)
144 1
170 1
182 1
190 1
269 1
Name: histogram_variance, Length: 133, dtype: int64
More accelerations appears more likely to be associated with Normal FHS
In [15]: # Create another figure
plt.figure(figsize=(10, 6))
# Scatter with normal examples
plt.scatter(df.baseline_value[df.fetal_health==1],
df.accelerations[df.fetal_health==1],
c="yellowgreen")
# Scatter with suspect examples
plt.scatter(df.baseline_value[df.fetal_health==2],
df.accelerations[df.fetal_health==2],
c="gold")
# Scatter with pathologic examples
plt.scatter(df.baseline_value[df.fetal_health==3],
df.accelerations[df.fetal_health==3],
c="firebrick")
# Add some helpful info
plt.title("Fetal Health Status in function of Baseline FHR and Accelerations")
plt.ylabel("Accelerations")
plt.xlabel("Baseline FHR")
plt.legend(["Normal", "Suspect", "Pathologic"]);
Fetal Health Status in function of Baseline FHR and Accelerations
Normal
Suspect
Pathologic

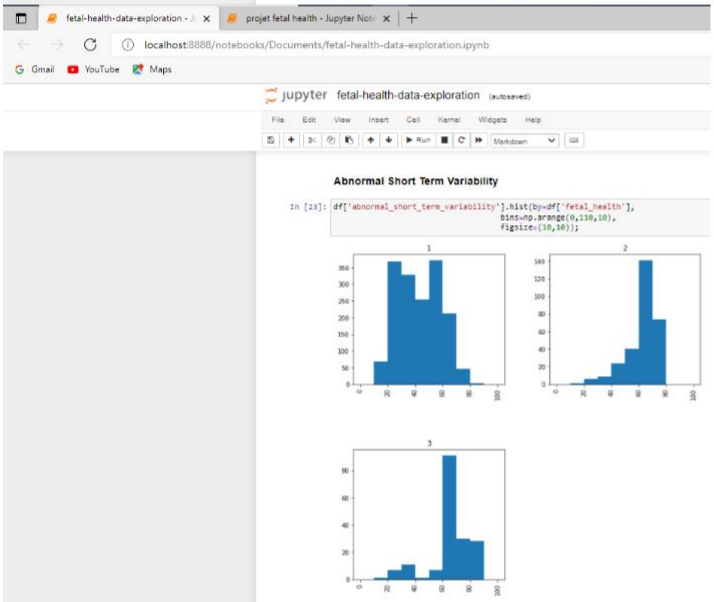

jupyter fetal-health-data-exploration (autosaved)
Suspect
Pathologic
Accelerations
Baseline FHR
Baseline FHR
In [16]: df.baseline_value.plot.hist(bins=np.arange(100,180,10),
figsize=(5,5));
Frequency
In [17]: df['baseline_value'].hist(by=df['fetal_health'],
bins=np.arange(100,180,10),

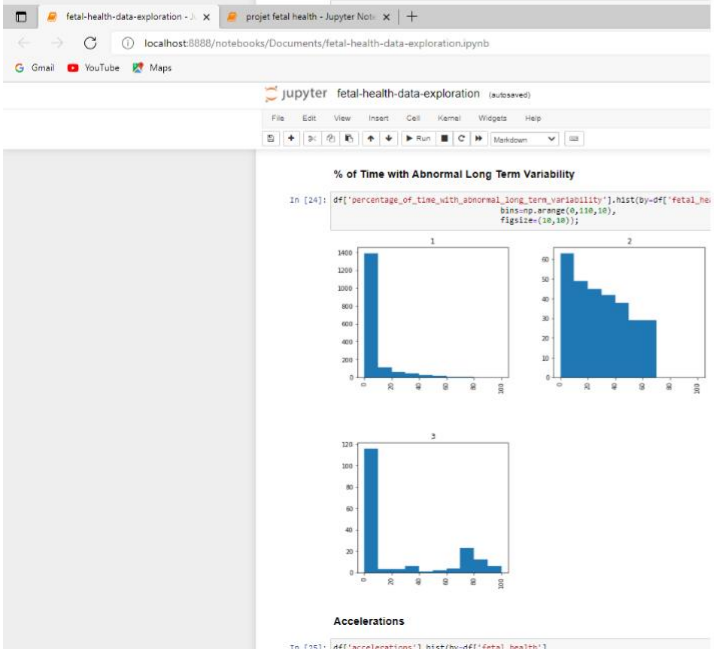

jupyter fetal-health-data-exploration (autosaved)
In [17]: df['baseline_value'].hist(by=df['fetal_health'],
bins=np.arange(100,180,10),
figsize=(10,10));
Fetal Health Statuses by Histogram Number of Zeroes
In [18]: len(df["histogram_number_of_zeroes"].value_counts())

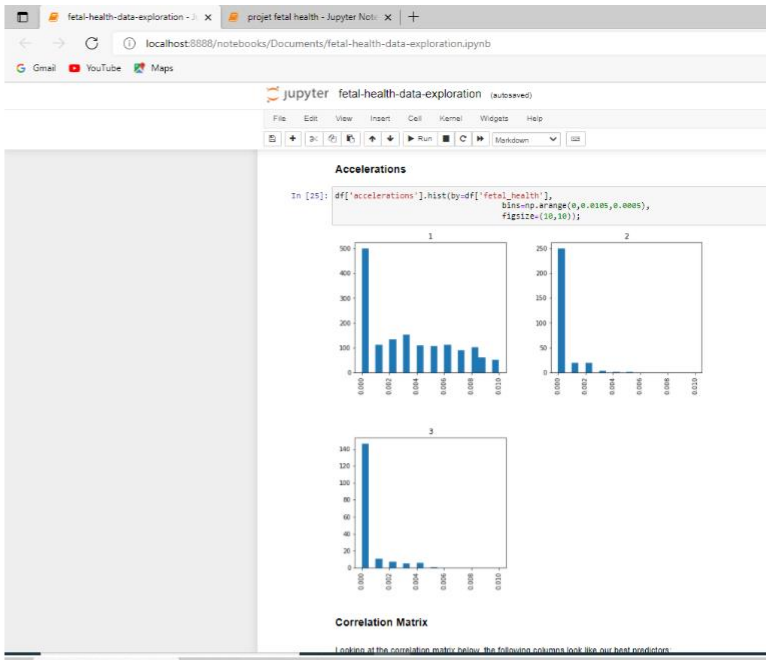

jupyter fetal-health-data-exploration (autosaved)
Fetal Health Statuses by Histogram Number of Zeroes
In [18]: len(df["histogram_number_of_zeroes"].value_counts())
Out[18]: 9
In [19]: pd.crosstab(df.histogram_number_of_zeroes, df.fetal_health)
Out[19]:
fetal_health 1 2 3
histogram_number_of_zeroes
0 1243 249 132
1 301 35 30
2 91 6 11
3 15 2 3
4 1 1 0
5 2 0 0
7 0 1 0
8 0 1 0
10 1 0 0
In [20]: # Make the crosstab more visual
pd.crosstab(df.prolongued_decelerations, df.fetal_health).plot(kind="bar",
figsize=(10,6),
color=["yellowgreen", "gold", "firebrick"])
# Add some communication
plt.title("Number of Patients with FHS per Histogram Number of Zeroes")
plt.xlabel("Histogram Tendency")
plt.ylabel("Amount")
plt.legend(["Normal", "Suspect", "Pathologic"])
plt.xticks(rotation=0);
Number of Patients with FHS per Histogram Number of Zeroes
Normal
Suspect

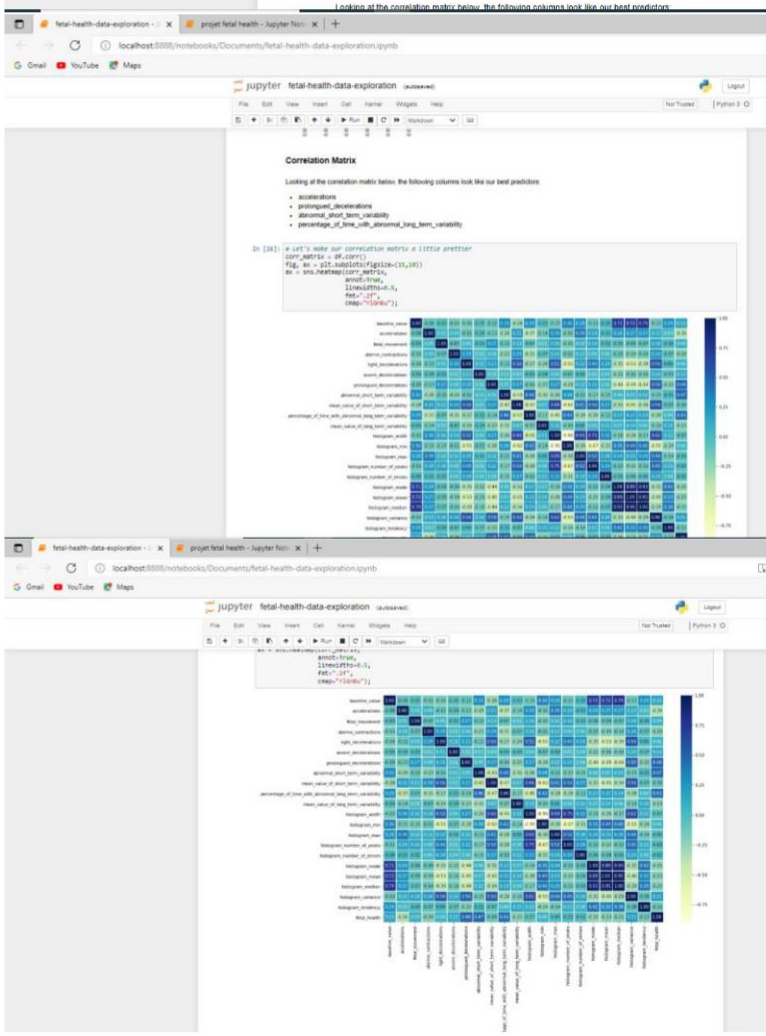

jupyter fetal-health-data-exploration (autosaved)
Number of Patients with FHS per Histogram Number of Zeroes
Normal
Suspect
Pathologic
Amount
Histogram Tendency
In [21]: # Make the crosstab more visual
pd.crosstab(df.fetal_health, df.prolongued_decelerations).plot(kind="bar",
figsize=(10,6),
color=["yellowgreen", "gold", "firebrick"])
# Add some communication
plt.title("Number of Patients with FHS per Histogram Number of Zeroes")
plt.ylabel("Histogram Tendency")
plt.xlabel("Amount")
plt.xticks(rotation=0);
Number of Patients with FHS per Histogram Number of Zeroes
prolongued_decelerations
0.0
0.001
0.002
0.003
0.004
0.005


fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)
prolongued_decelerations
Histogram Tendency
Amount
Prolongued Decelerations appears to vary by FHS
In [22]: df['prolongued_decelerations'].hist(by=df['fetal_health'],
bins=np.arange(0,0.0065,0.0005),
figsize=(10,10));

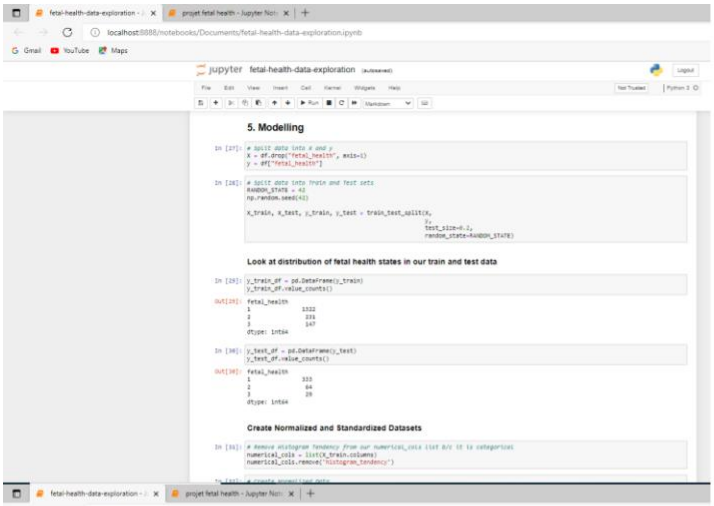

fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)
Abnormal Short Term Variability
In [23]: df['abnormal_short_term_variability'].hist(by=df['fetal_health'],
bins=np.arange(0,110,10),
figsize=(10,10));
% of Time with Abnormal Long Term Variability

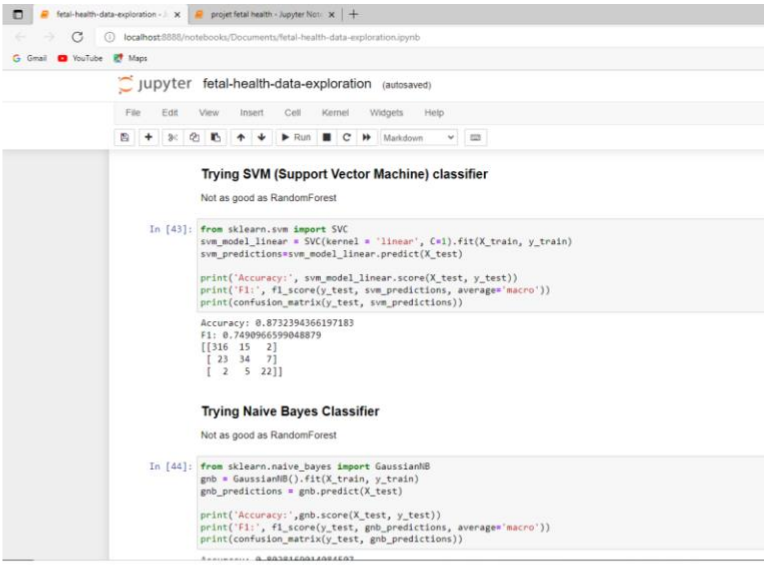

fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)
% of Time with Abnormal Long Term Variability
In [24]: df['percentage_of_time_with_abnormal_long_term_variability'].hist(by=df['fetal_he
bins=np.arange(0,110,10),
figsize=(10,10));
Accelerations

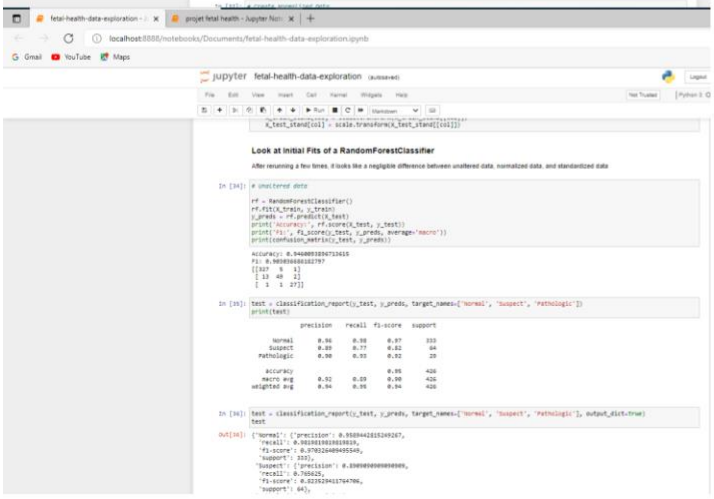

fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)
Accelerations
In [25]: df['accelerations'].hist(by=df['fetal_health'],
bins=np.arange(0,0.010,0.0005),
figsize=(10,10));
Correlation Matrix

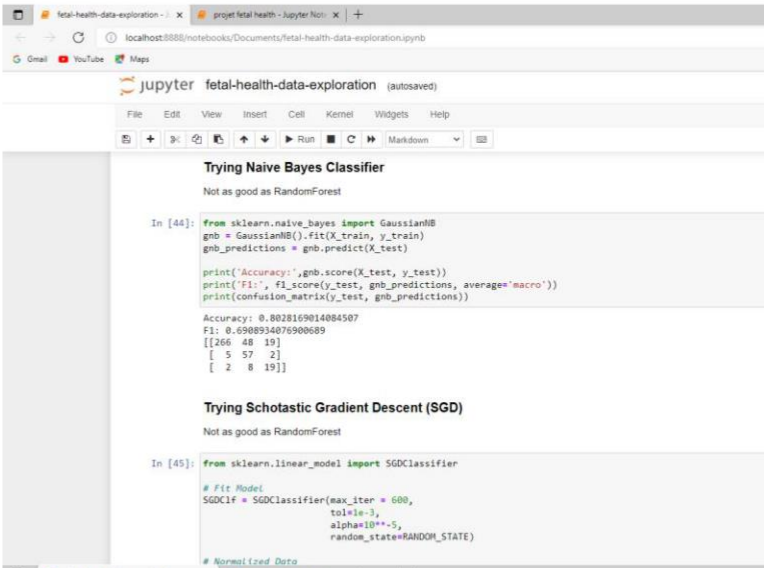

fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)
Correlation Matrix
Looking at the correlation matrix below, the following columns look like our best predictors:
accelerations
prolongued_decelerations
abnormal_short_term_variability
percentage_of_time_with_abnormal_long_term_variability
In [26]: # Let's make our correlation matrix a little prettier
corr_matrix = df.corr()
fig, ax = plt.subplots(figsize=(15,10))
ax = sns.heatmap(corr_matrix,
annot=True,
linewidths=0.5,
fmt=".2f",
cmap="YlGnBu");

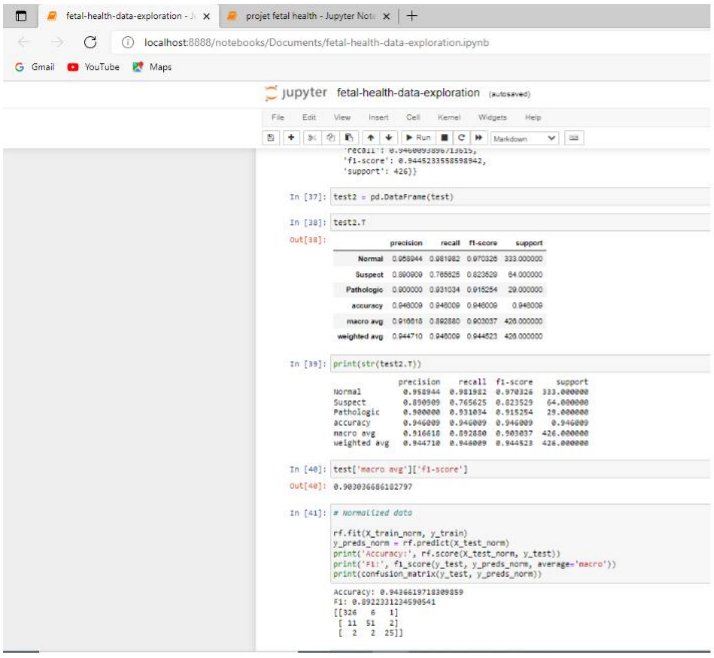
fetal-health-data-exploration - Jupyter Notebook
projet fetal health - Jupyter Notebook
localhost:8888/notebooks/Documents/fetal-health-data-exploration.ipynb
jupyter fetal-health-data-exploration (autosaved)

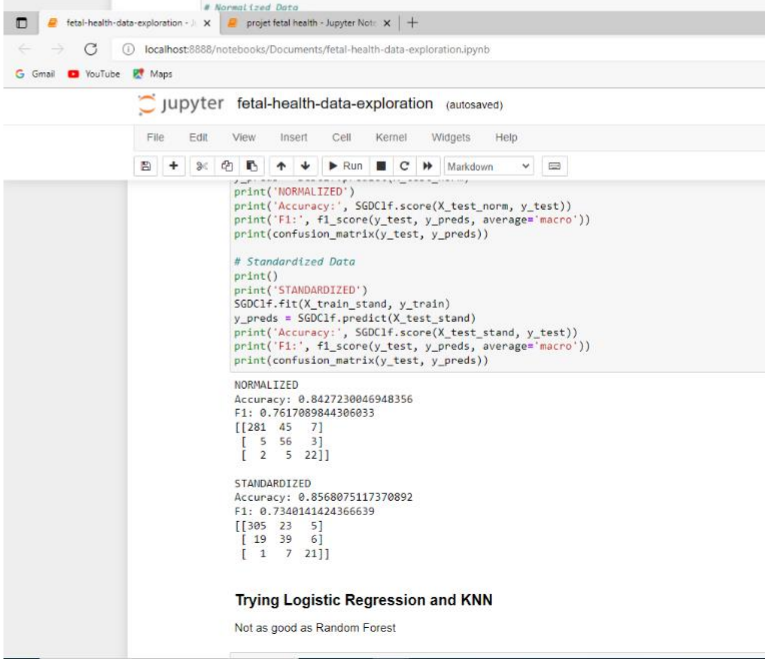

5. Modelling
Look at distribution of fetal health states in our train and test data
Create Normalized and Standardized Datasets
Look at Initial Fits of a RandomForestClassifier

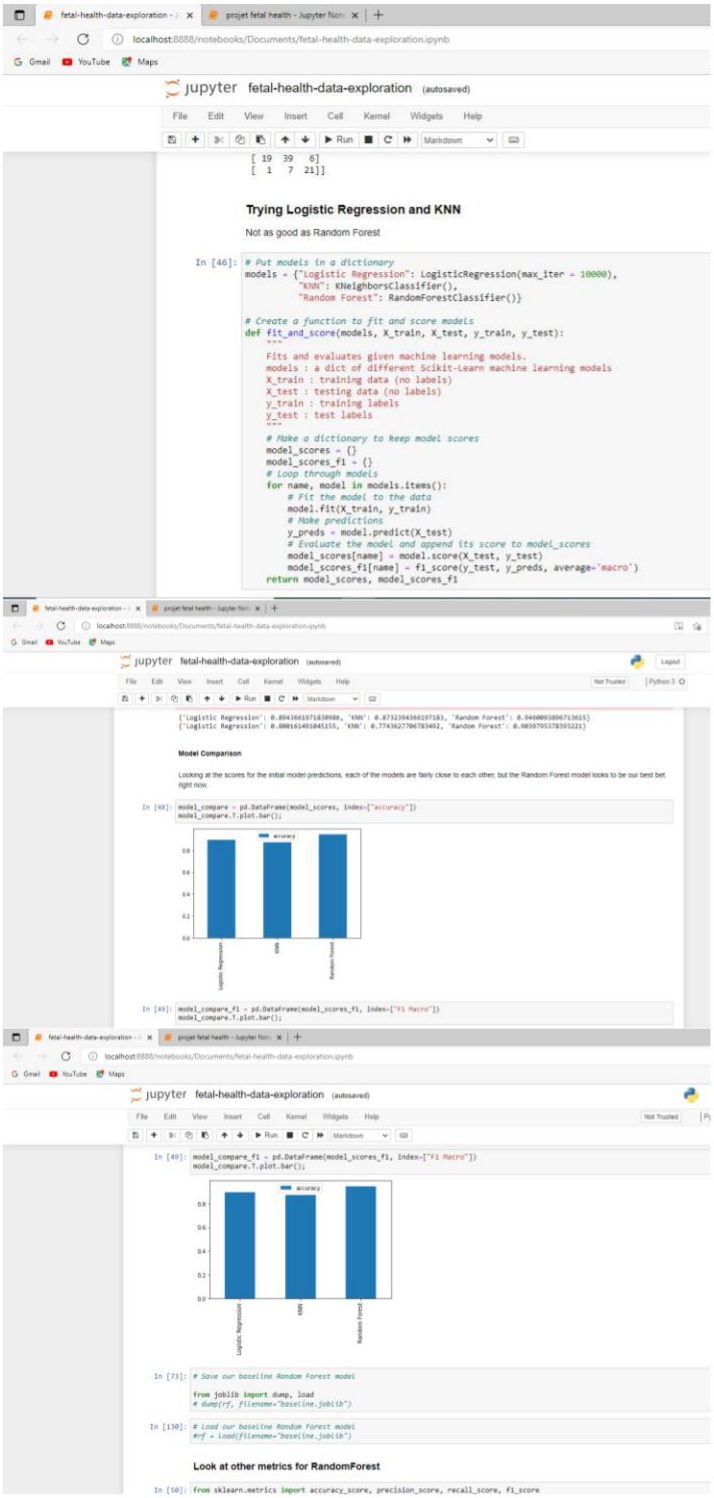

Trying SVM (Support Vector Machine) classifier
Not as good as RandomForest
Accuracy: 0.8732394366197183
F1: 0.7490966599048879
Trying Naive Bayes Classifier
Not as good as RandomForest
Accuracy: 0.8028169014084507
F1: 0.6908934075900689

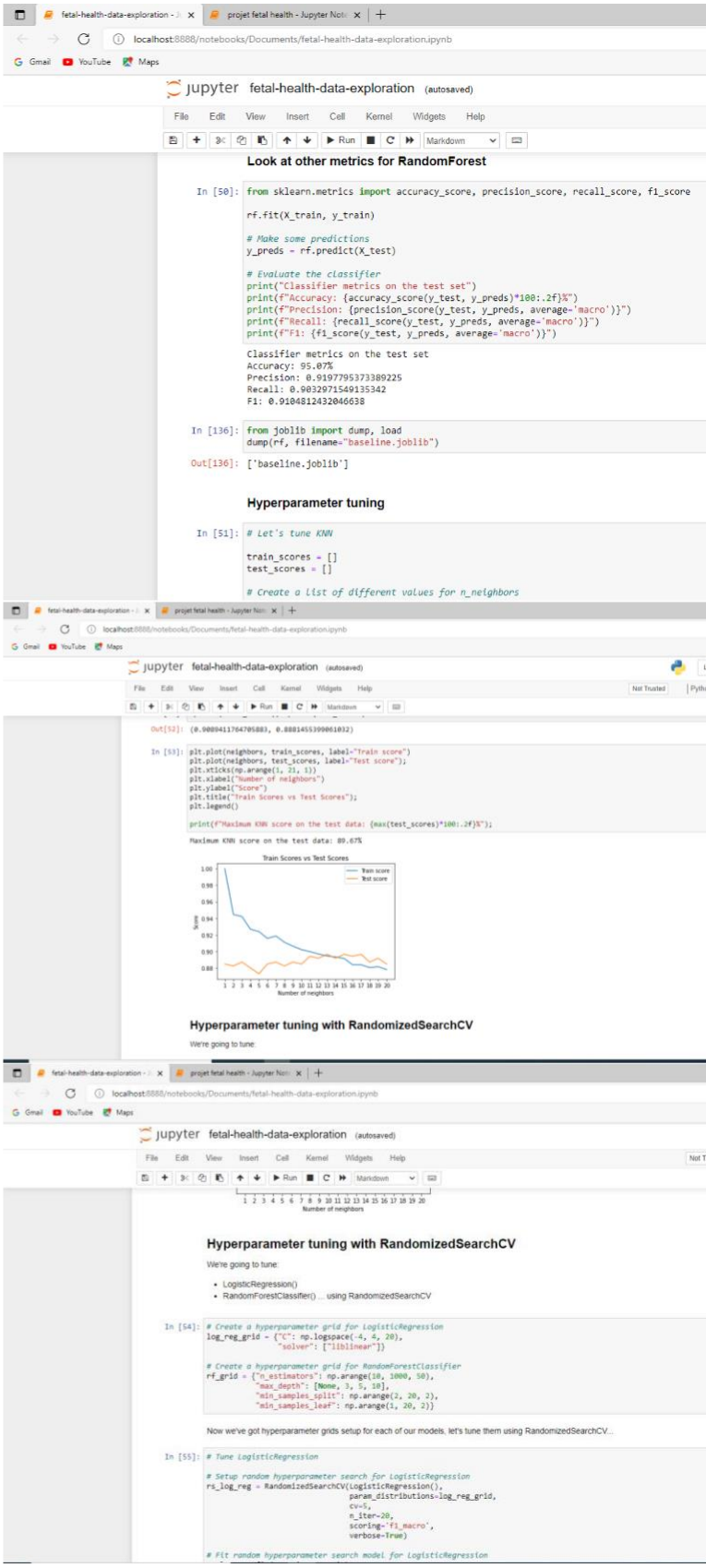

Look at Initial Fits of a RandomForestClassifier

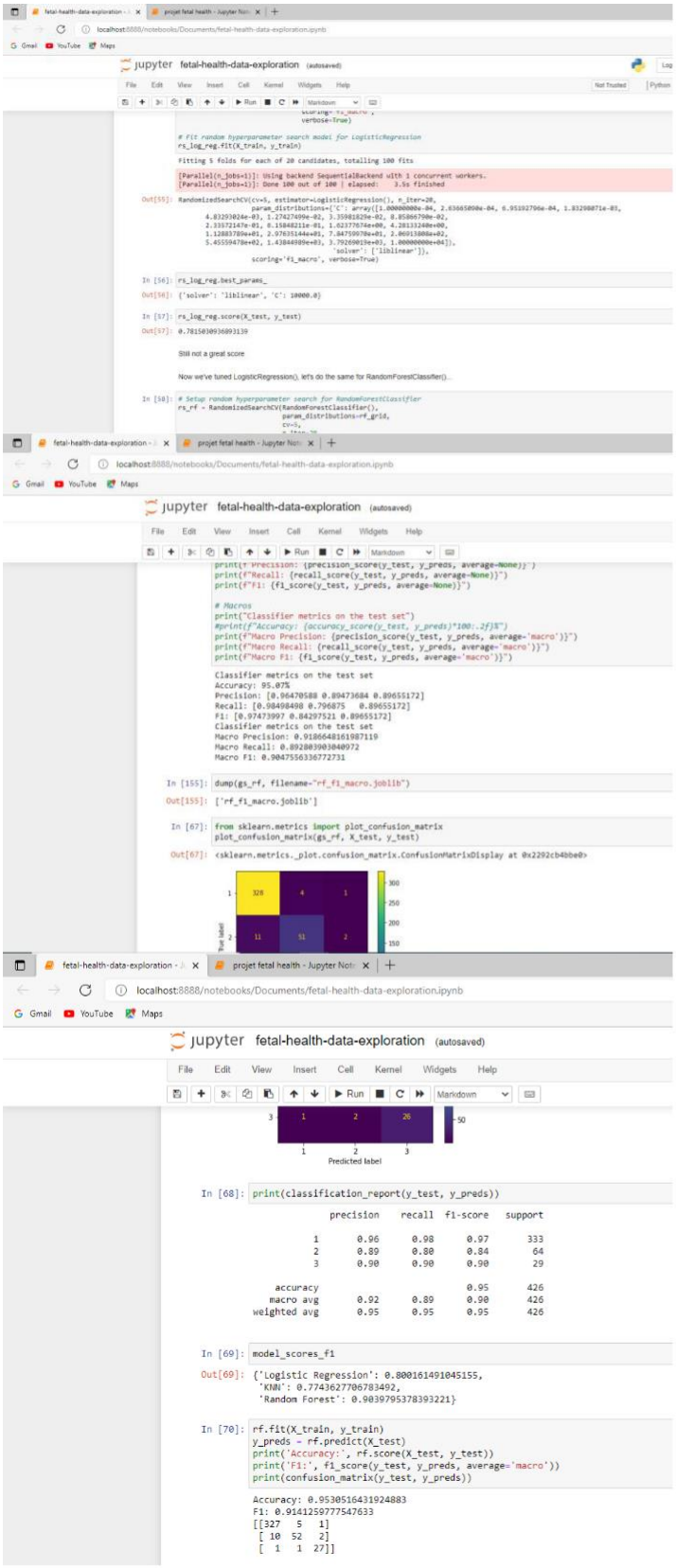

Trying Naive Bayes Classifier
Not as good as RandomForest
Accuracy: 0.8028169014084507
F1: 0.6908934075900689
Trying Schotastic Gradient Descent (SGD)
Not as good as RandomForest

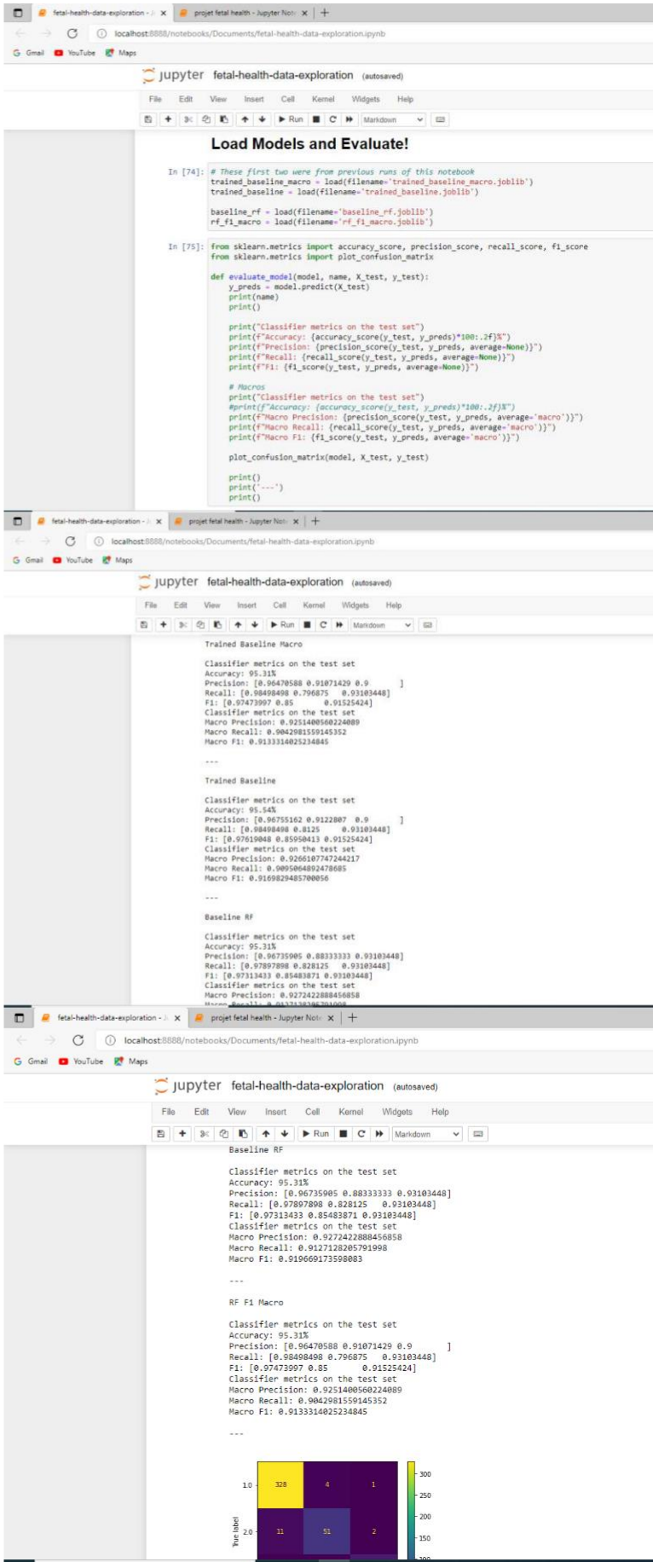

Out[38]:
In [39]:
In [40]:
Out[40]: 0.903036686182797
In [41]:


NORMALIZED
Accuracy: 0.8427230046948356
F1: 0.7617089844306033
STANDARDIZED
Accuracy: 0.8568075117370892
F1: 0.7340141424366639
Trying Logistic Regression and KNN
Not as good as Random Forest


jupyter fetal-health-data-exploration (autosaved)
Trying Logistic Regression and KNN
Not as good as Random Forest
Model Comparison
Look at other metrics for RandomForest


jupyter fetal-health-data-exploration (autosaved)
Look at other metrics for RandomForest
Hyperparameter tuning
Hyperparameter tuning with RandomizedSearchCV


jupyter fetal-health-data-exploration (autosaved)
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Still not a great score
Now we've tuned LogisticRegression(), let's do the same for RandomForestClassifier()...
Classifier metrics on the test set
Accuracy: 95.07%
Precision: [0.96470588 0.80473684 0.89655172]
Recall: [0.98498498 0.796875 0.89655172]
F1: [0.97473997 0.84297521 0.89655172]
Classifier metrics on the test set
Macro Precision: 0.9186648161987119
Macro Recall: 0.892883903840972
Macro F1: 0.9047556336772731
In [155]: dump(gs_rf, filename="rf_f1_macro.joblib")
Out[155]: ['rf_f1_macro.joblib']
In [67]: from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(gs_rf, X_test, y_test)
Out[67]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2292cb4bbe0>
Predicted label
In [68]: print(classification_report(y_test, y_preds))
precision recall f1-score support
1 0.96 0.98 0.97 333
2 0.89 0.80 0.84 64
3 0.90 0.90 0.90 29
accuracy 0.95 426
macro avg 0.92 0.89 0.90 426
weighted avg 0.95 0.95 0.95 426
In [69]: model_scores_f1
Out[69]: {'Logistic Regression': 0.800161491045155,
'KNN': 0.7743627706783492,
'Random Forest': 0.9039795378393221}
In [70]: rf.fit(X_train, y_train)
y_preds = rf.predict(X_test)
print('Accuracy:', rf.score(X_test, y_test))
print('F1:', f1_score(y_test, y_preds, average='macro'))
print(confusion_matrix(y_test, y_preds))
Accuracy: 0.9530516431924883
F1: 0.9141259777547633
[[327 5 1]
[10 52 2]
[1 1 27]]

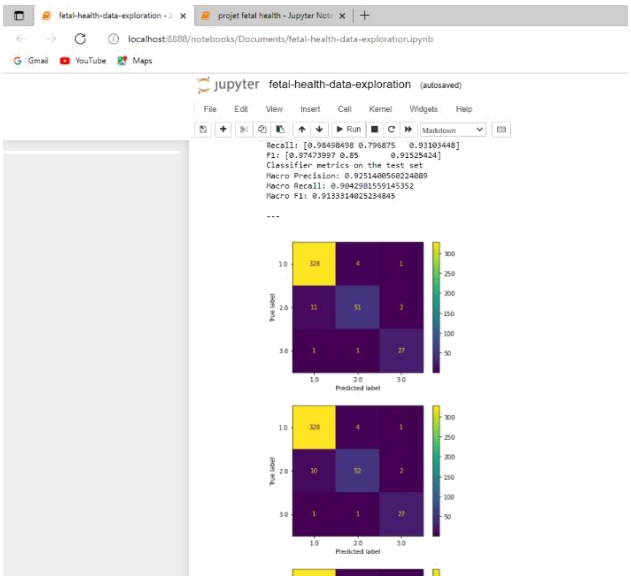

jupyter fetal-health-data-exploration (autosaved)
Load Models and Evaluate!
In [74]: # These first two were from previous runs of this notebook
trained_baseline_macro = load(filename='trained_baseline_macro.joblib')
trained_baseline = load(filename='trained_baseline.joblib')
baseline_rf = load(filename='baseline_rf.joblib')
rf_f1_macro = load(filename='rf_f1_macro.joblib')
In [75]: from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import plot_confusion_matrix
def evaluate_model(model, name, X_test, y_test):
y_preds = model.predict(X_test)
print(name)
print()
print("Classifier metrics on the test set")
print(f"Accuracy: {accuracy_score(y_test, y_preds)*100:.2f}%")
print(f"Precision: {precision_score(y_test, y_preds, average=None)}")
print(f"Recall: {recall_score(y_test, y_preds, average=None)}")
print(f"F1: {f1_score(y_test, y_preds, average=None)}")
# Macros
print("Classifier metrics on the test set")
#print(f"Accuracy: {accuracy_score(y_test, y_preds)*100:.2f}%")
print(f"Macro Precision: {precision_score(y_test, y_preds, average='macro')}")
print(f"Macro Recall: {recall_score(y_test, y_preds, average='macro')}")
print(f"Macro F1: {f1_score(y_test, y_preds, average='macro')}")
plot_confusion_matrix(model, X_test, y_test)
print()
print("---")
print()
Trained Baseline Macro
Classifier metrics on the test set
Accuracy: 95.31%
Precision: [0.96470588 0.91071429 0.9]
Recall: [0.98498498 0.796875 0.93103448]
F1: [0.97473997 0.85 0.91525424]
Classifier metrics on the test set
Macro Precision: 0.9251400560224089
Macro Recall: 0.9042981559145352
Macro F1: 0.9133314025234845

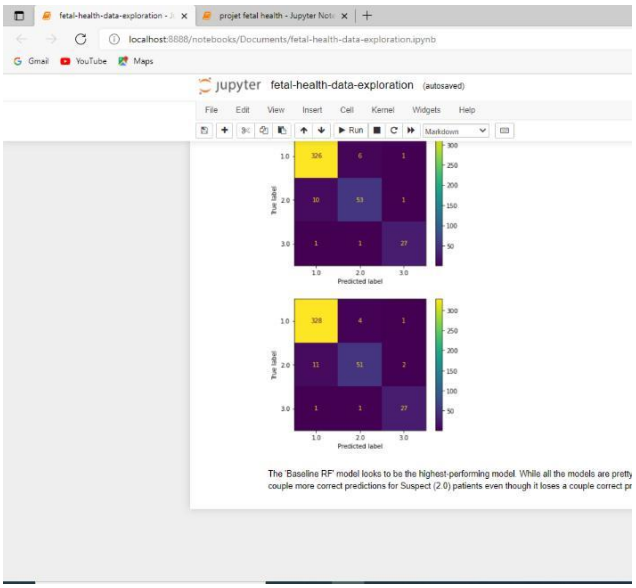
Trained Baseline
Classifier metrics on the test set
Accuracy: 95.54%
Precision: [0.96755162 0.9122807 0.9]
Recall: [0.98498498 0.8125 0.93103448]
F1: [0.97619048 0.85950413 0.91525424]
Classifier metrics on the test set
Macro Precision: 0.9266107747244217
Macro Recall: 0.9005064892478685
Macro F1: 0.9169829485700056

Baseline RF
Classifier metrics on the test set
Accuracy: 95.31%
Precision: [0.96735905 0.88333333 0.93103448]
Recall: [0.97897898 0.828125 0.93103448]
F1: [0.97313433 0.85483871 0.93103448]
Classifier metrics on the test set
Macro Precision: 0.9272422888456858
Macro Recall: 0.9127128205791998
Macro F1: 0.919669173598083

RF F1 Macro
Classifier metrics on the test set
Accuracy: 95.31%
Precision: [0.96470588 0.91071429 0.9]
Recall: [0.98498498 0.796875 0.93103448]
F1: [0.97473997 0.85 0.91525424]
Classifier metrics on the test set
Macro Precision: 0.9251400560224089
Macro Recall: 0.9042981559145352
Macro F1: 0.9133314025234845

True label





REFERENCES:

1]. Ali Gholipour, Estroff JudyA, Barnewolt CarolE, Connolly SusanA, Warfield SimonK. Fetal brain volumetry through MRI volumetric reconstruction and segmentation. Int. J. Comput. Assist. Radiol. Surg. 2011;6(3):329–39.

[2]. Anton-Rodriguez J. M, Peter Julyan, Ibrahim Djoukhadar, David Russell, D. Gareth Evans, Alan Jackson, and Julian C. Matthews, (2019) "Comparison of a Standard Resolution PET-CT Scanner With an HRRT Brain Scanner for Imaging Small Tumors Within the Head," IEEE Transactions on Radiation and Plasma Medical Sciences, vol. 3, no.4.

[3]. Challa M., Chinnaiyan R. (2020) Optimized Machine Learning Approach for the Prediction of Diabetes-Mellitus. In: Smys S., Tavares J., Balas V., Iliyasu A. (eds) Computational Vision and BioInspired Computing. ICCVBIC 2019. Advances in Intelligent Systems and Computing, vol 1108. Springer, Cham

[4]. G. Sabarmathi and R. Chinnaiyan, "Big Data Analytics Framework for Opinion Mining of Patient Health Care Experience," 2020 Fourth International
Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 2020, pp.352-357

[5]. G. Sabarmathi and R. Chinnaiyan, "Investigations on big data features research challenges and applications," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, 2017, pp. 782-786

[6]. G. Sabarmathi and R. Chinnaiyan, "Reliable Machine Learning Approach to Predict Patient Satisfaction for Optimal Decision Making and Quality Health Care," 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2019, pp. 1489-1493