**USER IDENTIFICATION FROM WALKING DATA**

A Course Project report submitted

in partial fulfillment of requirement for the award of degree

**BACHELOR OF TECHNOLOGY**

in

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

by

M.Nikhilesh (2103A51172)

D.Vamshi kumar (2103A51125)

Ch.Sumanth (2103A51123)

Under the guidance of

**Mr. D. Ramesh**

Assistant Professor, Department of CSE.



**Department of Computer Science and Artificial Intelligence**



**Department of Computer Science and Artificial Intelligence**

**CERTIFICATE**

This is to certify that project entitled **“USER IDENTIFICATION FROM WALKING DATA** " is the bonafide work carried out by **MAJHI NIKHILESH, D VAMSHI KUMAR, CH SUMANTH** as a Course Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** during the academic year 2022-2023 under our guidance and Supervision.

**Mr. D. Ramesh Dr. M Sheshikala**

Asst. Professor, Assoc. Prof. & HOD (CSE),

S R University, S R University,

Ananthasagar, Warangal. Ananthasagar, Warangal.

**ACKNOWLEDGEMENT**

We express our thanks to Course co-coordinator **Mr. D. Ramesh, Asst. Prof.** for guiding us from the beginning through the end of the Course Project. We express our gratitude toHead of the department CS&AI, **Dr. M. Sheshikala, Associate Professor** for encouragement, support and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dean, School of Computer Science and Artificial Intelligence, **Dr C. V. Guru Rao**, for his continuous support and guidance to complete this project in the institute.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

**ABSTRACT**

Mobile and wearable technology are becoming more prevalent and play an increasingly significant part in our daily lives. Mobile devices are used to store highly sensitive data, such as vital documents and files. Using data from accelerometers in mobile and wearable devices, we want to categorize people and identify their routines. Previous research on gait analysis revealed the best results in terms of minimizing mistake rates in spotting a person using their motion data.

**CONTENTS**

***ABSTRACT iv***

**Chapter No. Title Page No.**

**1 INTRODUCTION** 01

1.1 Problem Statement 01

1.2 Existing System 01

1.3 Proposed System 02

1.4 Objectives 02

1.5 Architecture 02

**2 Literature survey** 03

2.1 Analysis of the Survey 03

**3. Data pre-processing** 05

3.1 Dataset Description 05

3.2 Pre-processing through Standardscaler 07

3.2 Data Visualization 08

**4. METHODOLOGY** 12

4.1 Procedure to solve 12

4.1.1 Using KNN 12

4.1.3 Using Logistic Regression 13

4.2 Software Description 14

4.2.1 Through KNN 14

4.2.3 Through Logistic Regression 20

**5.**  **RESULTS**  29

**6.** **CONCLUSION** 30

6.1 Conclusion 30

6.2 Future scope 30

**REFERENCES** 31

# CHAPTER 1

# INTRODUCTION

Monitoring fetal growth during pregnancy is one of the most challenging and complicated processes in the medical field. According to the World Health Organization (WHO), roughly 810 pregnant women die each day even though preventable measures have been taken (WHO, 2021). The maternal mortality ratio (MMR) is considerably low in developed countries and high in underdeveloped countries. Common complications behind high MMR include pre-eclampsia, improper monitoring of unborn baby condition and mother, and gestational diabetes.

Machine Learning (ML) techniques can help medical experts make early decisions during complex situations like diagnosis, effectively decreasing the MMR and high complications during labour. Classifying the stages of fetal health is a challenging task, but this can be outstandingly handled by ML classification techniques. KNN and Logistic regression are the classification methods employed here.

## Problem Statement

A fetal is basically an unborn offspring which is in the embryo stage until it comes to the world. During the pregnancy process, each three-month period is known by a name called trimester. During this process the fetus grows and develops and along with it the regular checkups are very important.

A pregnancy lasts for 9 months and in this long period there may be various reasons which may cause disability or mortality in the newborn which is a very severe case and this needs to be avoided. One of the main tools to analyze the health of the fetal in the womb is by doing a CTG(Cardiotocography) which generally is used to evaluate the heart beat and the uterine contractions. The data generated is used by the doctor to analyze the health and give his wording. There is a room for error hence the doctors are not reliable to analyze the data. Hence different machine and deep learning algorithms have been there which can analyze the data and predict the fetals health based on it.

The main motive is to find the prediction accuracy using the different classification models and compare which model performs better.

## Existing System

Predicting health of fetal has been studied extensively. Works related include classification model that was built KNN and SVM which helps with better precision than some simple classification. SVM is better in dealing with datasets with more dimensions and it is less prone to over fitting and under fitting. To support advanced KNN; basic indicators such as mean, variance and standard deviation are required.

## Proposed System

With the assist of dataset obtained we create 2 different machine algorithms, specifically KNN and Logistic Regression and examine the outcomes of accuracy and find which models performs better and is reliable.

## Objectives

* Compare the accuracy in 2 specific classification-based system learning algorithms.
* To establish machine learning algorithms are reliable for automatic results.
* Smoother the troublesome method throughout the child’s fetal health and mothers’ maternity.

## Architecture

The Supervised-Learning-approach as a qualitative-data with KNN classification, logistic Regression and its target to predict health of fetal, which might be normal, suspect or pathological.

## CHAPTER 2

## LITERATURE SURVEY

**2.1 ANALYSIS OF the survey**

The dataset that has been used is the CTG data which is observed to be beneficial to identify the abnormalities. The visual analysis along which the decision support system focuses has been made on the machine learning models that have been used. As the machine learning doesn’t perform well on the basis of accuracy the ensemble model has been used which has bagged an accuracy of 99.02% after the 10-fold cross validation has been employed. Hence, this can be used to classify the normal and the pathological cases of the ctg data. We will have data and with the help of that data inputs that we have we predict the fetal health.

A standard procedure done during the third trimester is fetal monitoring. Fetal monitoring is checking the health of the unborn baby. Fetal growth entirely depends on the mother's health. To avoid such complications, a continuous measuring of fetus health and growth rate is done with cardiotocography. The cardiotocography aims to track the fetus' heartbeat and parallelly measure the mother's uterine contractions. This process would be performed during the final trimester, once the fetus' growth functions fully with heart rate. This method is considered cost-effective and straightforward, and hence, it is to be carried out by medical experts for early detection of fetal status and to reduce fetal mortality. The result of cardiotocographic (CTG) will trace uterine contraction of the mother, most importantly heart rate of fetus, occurrence of acceleration, series of deceleration, and much more complicated measure of fetus. There are many ML techniques available for classifying the fetus's normal, suspected, and pathological stages. The results show that the ML technique will form a framework widely used for the automated system in analyzing early fetal health.

Artificial Intelligence (AI) techniques have recently provided a significant decision for early diagnosis and multi-classifications. A significant comparison was made between 15 ML techniques defending healthy vs. affected fetuses. The features are extracted from CTG signal recording. These effectively evaluate large amounts of real-time data to provide better solutions and develop a framework for other models to perform classification. CTC signals to directly assess the heart rate of the patients and give accurate results and updates to the medical experts. The effectiveness of using cardiotocography is discussed for the wellbeing of the fetal during labor. CTG is responsible for measuring the fetal heart rate and the contraction of the womb; hence this plays a vital role in assessing fetal before birth and also during labor.

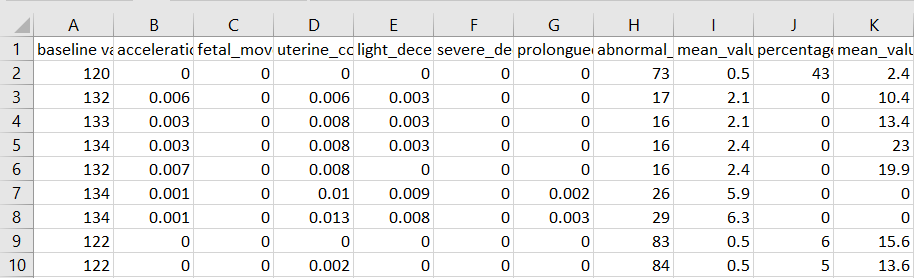
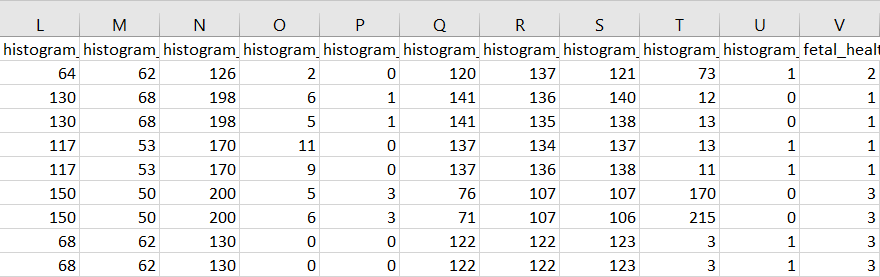
This would also measure the frequency of baby movement. A recent rush in the deployment of two ML techniques, namely K-nearest neighbors (K-NN) Classifier, Logistic Regression which are evaluated on high dimensional data, proves that the classifier KNN fairly dominates the other technique in giving accurate diagnostic indices. Both the Classifiers work well and use 30 ranked features to determine common risk factors in the prediction model. While using ML algorithms, feature extraction and selection are among the most used methods to select optimized features for prediction in the model.

## CHAPTER 3

## DATA PRE-PROCESSING

This dataset contains 2126 records of features extracted from Cardiotocogram exams, which were then classified by expert obstetrician into 3 classes: Normal, Suspect, Pathological. The dataset has already been thoroughly analyzed by experts; thus, no cleaning was required to handle missing values, data cleaning, or noisy value handling.

### 3.1 Dataset Description

This dataset contains 2126 rows of data and 22columns (features) that we could focus onto build our prediction model.

Fetal and maternal health have a close correlation. Reducing the fetal death rate and monitoring fetal health conditions are essential for maintaining good health for both mother and child. CTG or electronic fetal monitoring (EFM) is the prenatal test for monitoring uterine contractions and fetal heartbeat during pregnancy and childbirth. Two transducers are deployed to monitor these parameters, and the classification of fetal health state (FHS) is done based on the baseline value, permissible variability, decelerations, and accelerations. Doctors manually investigate these values and categorize the health state of the fetus. Table 1 gives an insight into the parameters and the classification of fetal health conditions that is widely followed. The health status is labelled as normal if all the four parameters mentioned in Table 1 are under health limits. A suspicious health state can be anticipated if any values do not adhere to a healthy state. The results are clinically proven, which is represented in the table. The results are examined by medical experts and classified with a respective label for each feature. This data is collected by monitoring heartbeat and uterine reductions during pregnancy by the method called CTG using an EMF system.

**TABLE 1: Chart for assessing fetal health condition from EFM**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Baseline heart beat (bpm) | Variability (bpm) | Decelerations | Accelerations | Fetal health state |
| 110–160 | > = 5 | — | Present/absence of accelerations  with other parameters in normal states | Healthy |
| 100–109/110–160 | > = 40 and < 5 for less than 1.5 h | Early, Variable/Single (for less than 3 min) | — | Need further investigation |
| <100 or > 180/ sinusoidal for > = 10 min | <5 or > 90 for greater that 1.5 h | Atypical, late, single (for more than 3 min) | — | Abnormal |

**TABLE 2: Attributes of the Cardiotocographic data**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute (s) | Range | Attribute | Range |
| Fetal heart rate baseline | (160–106) | Width of FHR histogram | (3–180) |
| Accelerations | (0–0.019) | Minimum of FHR histogram | (50–159) |
| Fetal movements | (0–0.481) | Maximum of FHR histogram | (122–238) |
| Uterine contractions | (0–0.015) | Count of histogram peaks | (0–18) |
| Light decelerations | (0–0.015) | Count of histogram zeros | (0–10) |
| Severe decelerations | (0–0.001) | Histogram mode | (60–187) |
| Prolonged decelerations | (0–0.005) | Histogram mean | (73–182) |
| Percentage of time with abnormal short-term variability (S-TV) | (12–87) | Histogram median | (77–186) |
| Mean value of S-TV | (7–0.2) | Histogram variance | (0–269) |
| Percentage of time with abnormal | (0–91) | Histogram tendency | 1 |
| long-term variability (L-TV) | | | |

Mean value of L-TV (0–50.7) Fetal status (1-Normal, 2-Needs Reassurance, 3-Pathological)

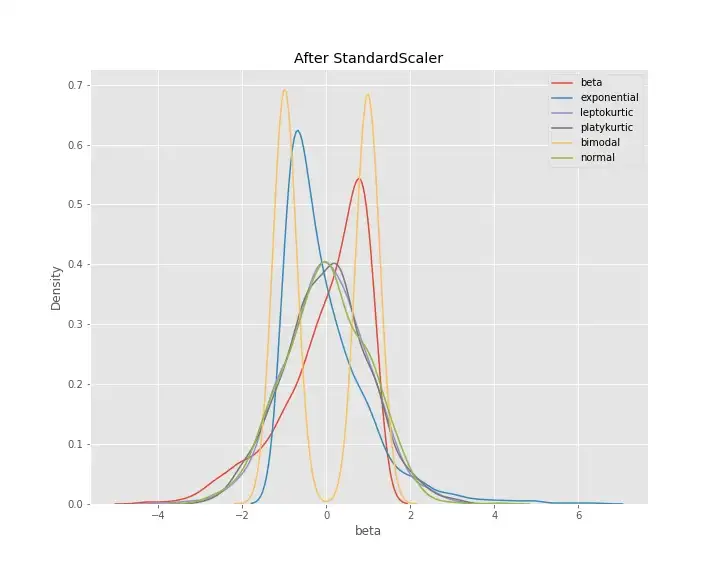
Medical experts classify recorded data manually and fit it into the range where the value gets satisfied, and an apprehensive state is given when the value does not provide the range.

ML algorithms can assist doctors in concluding the fetal health status by predicting the health state. The CTG data collected by the University of Porto from 1831 recordings are used as a

benchmark dataset to assess the performance of the ML models. These recordings are automated by SisPorto2.0 (CTG Analysis Program) and are made available in UCI-ML. The attribute details of the CTG data are presented in Table 2.

**3.2 PRE-PROCESSING THROUGH STANDARDSCALER**

Many machine learning algorithms work better when features are on a relatively similar scale and close to normally distributed. We use ‘StandardScaler’, a scikit-learn method to preprocess data.

* Scale generally means to change the range of the values. The shape of the distribution doesn’t change. Think about how a scale model of a building has the same proportions as the original, just smaller. That’s why we say it is drawn to scale. The range is often set at 0 to 1.
* Standardize generally means changing the values so that the distribution’s standard deviation equals one. Scaling is often implied.
* StandardScaler standardizes a feature by subtracting the mean and then scaling to unit variance. Unit variance means dividing all the values by the standard deviation.
* StandardScaler results in a distribution with a standard deviation equal to 1. The variance is equal to 1 also, because variance = standard deviation squared and 1 squared = 1, StandardScaler makes the mean of the distribution approximately 0.

**3.3 DATA VISUALIZATION**

The following are plotting of each feature against the target.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# CHAPTER 4

# METHODOLOGY

**4.1. PROCEDURE TO SLOVE THE GIVEN PROBLEM**

**4.1.1 Using KNN**

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.



By calculating the Euclidean distance, we get the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B.

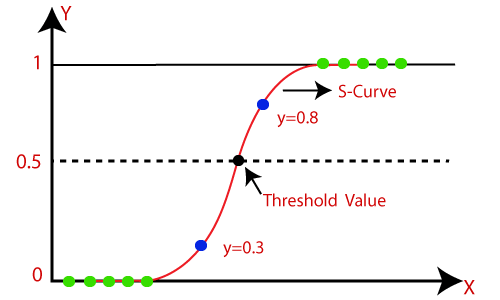
As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

**4.1.2. Using Logistic Regression**

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 regression is used when the dependent variable is binary such as click on a given advertisement link or not, spam detection, Diabetes prediction, the customer will purchase or not, an employee will leave the company or not.

Logistic regression uses Maximum Likelihood Estimation (MLE) approach i.e., it determines the parameters (mean and variance) that are maximizing the likelihood to produce the desired output.



Logistic Regression uses a sigmoid or logit function which will squash the best fit straight line that will map any values including the exceeding values from 0 to 1 range. So, it forms an “S” shaped curve.

Sigmoid function removes the effect of outlier and makes the output between 0 to 1.

**The**[**logistic function**](https://en.wikipedia.org/wiki/Logistic_function)**is of the form:**

where μ is a [location parameter](https://en.wikipedia.org/wiki/Location_parameter)

s is a [scale parameter](https://en.wikipedia.org/wiki/Scale_parameter).

## 4.2 SOFTWARE DESCRIPTION

We used the Google colab service to test our machine learning algorithms written in Python. The jupyter notebooks with the results shown below.

## 4.2.1 THROUGH KNN

KNeighborsClassifier

Upload dataset

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():

  print('User uploaded file "{name}" with length {length} bytes'.format(

      name=fn, length=len(uploaded[fn])))

Choose Files fetal\_health.csv

**fetal\_health.csv**(text/csv) - 228715 bytes, last modified: 9/10/2022 - 100% done

Saving fetal\_health.csv to fetal\_health.csv

User uploaded file "fetal\_health.csv" with length 228715 bytes

Importing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

Loading dataset

ds=pd.read\_csv("fetal\_health.csv")

print("Shape of dataset : ",ds.shape)

print("Dataset :-\n",ds.head())

Shape of dataset : (2126, 22)

Dataset :-

baseline value accelerations fetal\_movement uterine\_contractions \

0 120.0 0.000 0.0 0.000

1 132.0 0.006 0.0 0.006

2 133.0 0.003 0.0 0.008

3 134.0 0.003 0.0 0.008

4 132.0 0.007 0.0 0.008

light\_decelerations severe\_decelerations prolongued\_decelerations \

0 0.000 0.0 0.0

1 0.003 0.0 0.0

2 0.003 0.0 0.0

3 0.003 0.0 0.0

4 0.000 0.0 0.0

abnormal\_short\_term\_variability mean\_value\_of\_short\_term\_variability \

0 73.0 0.5

1 17.0 2.1

2 16.0 2.1

3 16.0 2.4

4 16.0 2.4

percentage\_of\_time\_with\_abnormal\_long\_term\_variability ... histogram\_min \

0 43.0 ... 62.0

1 0.0 ... 68.0

2 0.0 ... 68.0

3 0.0 ... 53.0

4 0.0 ... 53.0

histogram\_max histogram\_number\_of\_peaks histogram\_number\_of\_zeroes \

0 126.0 2.0 0.0

1 198.0 6.0 1.0

2 198.0 5.0 1.0

3 170.0 11.0 0.0

4 170.0 9.0 0.0

histogram\_mode histogram\_mean histogram\_median histogram\_variance \

0 120.0 137.0 121.0 73.0

1 141.0 136.0 140.0 12.0

2 141.0 135.0 138.0 13.0

3 137.0 134.0 137.0 13.0

4 137.0 136.0 138.0 11.0

histogram\_tendency fetal\_health

0 1.0 2.0

1 0.0 1.0

2 0.0 1.0

3 1.0 1.0

4 1.0 1.0

[5 rows x 22 columns]

Count of each class in Target

count=ds['fetal\_health'].value\_counts()

print("Count of each class in Target :-\n",count)

Count of each class in Target :-

1.0 1655

2.0 295

3.0 176

Name: fetal\_health, dtype: int64

Assigning X and y values

X=ds.iloc[:,:-1].values

print("X Shape is : ",X.shape)

print("X is :-\n",X);

X Shape is : (2126, 21)

X is :-

[[1.20e+02 0.00e+00 0.00e+00 ... 1.21e+02 7.30e+01 1.00e+00]

[1.32e+02 6.00e-03 0.00e+00 ... 1.40e+02 1.20e+01 0.00e+00]

[1.33e+02 3.00e-03 0.00e+00 ... 1.38e+02 1.30e+01 0.00e+00]

...

[1.40e+02 1.00e-03 0.00e+00 ... 1.52e+02 4.00e+00 1.00e+00]

[1.40e+02 1.00e-03 0.00e+00 ... 1.51e+02 4.00e+00 1.00e+00]

[1.42e+02 2.00e-03 2.00e-03 ... 1.45e+02 1.00e+00 0.00e+00]]

y=ds.iloc[:,-1].values

print("y Shape is : ",y.shape)

print("y is :-\n",y)

y Shape is : (2126,)

y is :-

[2. 1. 1. ... 2. 2. 1.]

Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 50)

print("Shape of X\_train is : ",X\_train.shape)

print("X\_train is :-\n",X\_train)

Shape of X\_train is : (1488, 21)

X\_train is :-

[[1.27e+02 7.00e-03 0.00e+00 ... 1.30e+02 4.50e+01 0.00e+00]

[1.43e+02 0.00e+00 2.10e-02 ... 1.47e+02 1.00e+00 0.00e+00]

[1.20e+02 8.00e-03 1.03e-01 ... 1.26e+02 2.50e+01 0.00e+00]

...

[1.33e+02 0.00e+00 1.00e-03 ... 1.32e+02 2.10e+01 1.00e+00]

[1.30e+02 6.00e-03 0.00e+00 ... 1.37e+02 4.00e+00 0.00e+00]

[1.31e+02 7.00e-03 4.00e-03 ... 1.43e+02 2.50e+01 0.00e+00]]

print("Shape of y\_train is : ",y\_train.shape)

print("y\_train is :-\n",y\_train)

Shape of y\_train is : (1488,)

y\_train is :-

[1. 2. 1. ... 1. 1. 1.]

print("Shape of X\_test is : ",X\_test.shape)

print("X\_test is :-\n",X\_test)

Shape of X\_test is : (638, 21)

X\_test is :-

[[1.29e+02 2.00e-03 1.30e-02 ... 1.34e+02 4.00e+00 1.00e+00]

[1.34e+02 9.00e-03 1.00e-03 ... 1.51e+02 4.20e+01 0.00e+00]

[1.30e+02 2.00e-03 1.00e-03 ... 1.36e+02 1.50e+01 0.00e+00]

...

[1.37e+02 0.00e+00 4.00e-03 ... 1.39e+02 1.00e+00 0.00e+00]

[1.27e+02 6.00e-03 0.00e+00 ... 1.29e+02 4.10e+01 0.00e+00]

[1.25e+02 4.00e-03 1.00e-03 ... 1.19e+02 3.00e+01 0.00e+00]]

print("Shape of y\_test is : ",y\_test.shape)

print("y\_test is :-\n",y\_test)

Shape of y\_test is : (638,)

y\_test is :-

[1. 1. 1. 2. 1. 1. 1. 1. 2. 3. 3. 1. 1. 2. 1. 2. 3. 1. 1. 3. 1. 1. 1. 1.

1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3. 3. 1.

1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2. 2. 1. 1. 1.

2. 1. 2. 1. 3. 3. 2. 2. 1. 1. 1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3.

1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3. 1. 1. 1. 1.

2. 1. 1. 1. 1. 3. 1. 1. 2. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.

1. 1. 3. 1. 1. 1. 1. 1. 1. 3. 1. 1. 1. 1. 3. 1. 1. 1. 1. 1. 2. 1. 1. 1.

1. 2. 2. 1. 1. 1. 1. 1. 1. 2. 1. 3. 1. 1. 1. 1. 1. 2. 1. 2. 1. 3. 1. 1.

1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1.

1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 2. 3. 1. 1. 2. 1. 2. 1. 2. 1. 2.

1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 3.

3. 1. 1. 2. 3. 1. 3. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 2.

2. 1. 1. 1. 1. 1. 1. 1. 3. 1. 1. 3. 1. 1. 1. 1. 2. 2. 1. 1. 1. 1. 1. 1.

1. 1. 1. 1. 1. 1. 3. 1. 1. 1. 1. 3. 1. 1. 2. 1. 2. 1. 1. 3. 1. 2. 1. 1.

1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 3. 1. 1. 3. 3. 2. 2. 1. 2. 1. 1. 1. 1. 1.

1. 1. 1. 1. 1. 1. 3. 1. 3. 1. 1. 1. 1. 1. 3. 2. 1. 1. 1. 1. 1. 1. 3. 1.

1. 1. 1. 1. 2. 2. 1. 1. 2. 1. 3. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1.

2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 3. 2. 1. 2. 1. 1. 1. 1. 1. 2. 1. 1.

1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2.

2. 1. 1. 1. 1. 2. 1. 3. 1. 1. 1. 1. 1. 1. 2. 1. 1. 1. 2. 3. 2. 1. 1. 1.

1. 2. 1. 1. 1. 1. 3. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 3. 2. 1. 1.

3. 1. 3. 1. 1. 1. 1. 2. 2. 1. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1.

1. 2. 1. 1. 1. 2. 1. 1. 1. 1. 1. 3. 3. 3. 1. 1. 1. 3. 1. 1. 1. 2. 2. 1.

1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 3. 1. 3. 3. 1. 3.

1. 1. 1. 3. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 1. 1. 2. 1. 1. 1. 1. 1. 1. 1.

1. 3. 1. 1. 1. 2. 3. 1. 3. 2. 1. 1. 1. 1. 1. 1. 1. 1. 2. 3. 2. 2. 1. 1.

2. 1. 1. 1. 1. 1. 2. 1. 1. 3. 1. 2. 1. 1.]

Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print("Now X\_train is :-\n",X\_train)

Now X\_train is :-

[[-0.63437049 0.97001811 -0.19565677 ... -0.58503156 0.90519778

-0.54442589]

[ 0.98214376 -0.83035762 0.28240115 ... 0.60852908 -0.6078638

-0.54442589]

[-1.34159548 1.22721464 2.14910351 ... -0.86586935 0.21744252

-0.54442589]

...

[-0.02817765 -0.83035762 -0.17289211 ... -0.44461266 0.07989146

1.11562683]

[-0.33127407 0.71282157 -0.19565677 ... -0.09356541 -0.50470051

-0.54442589]

[-0.23024193 0.97001811 -0.10459812 ... 0.32769128 0.21744252

-0.54442589]]

print("Now X\_test is :-\n",X\_test)

Now X\_test is :-

[[-0.43230621 -0.31596456 0.10028385 ... -0.30419376 -0.50470051

1.11562683]

[ 0.07285449 1.48441117 -0.17289211 ... 0.88936687 0.80203449

-0.54442589]

[-0.33127407 -0.31596456 -0.17289211 ... -0.16377486 -0.12643512

-0.54442589]

...

[ 0.37595092 -0.83035762 -0.10459812 ... 0.04685348 -0.6078638

-0.54442589]

[-0.63437049 0.71282157 -0.19565677 ... -0.65524101 0.76764673

-0.54442589]

[-0.83643478 0.19842851 -0.17289211 ... -1.3573355 0.38938133

-0.54442589]]

Training the KNeighborsClassifier model on the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier()

classifier.fit(X\_train, y\_train)

KNeighborsClassifier()

Predicting the Test set results

y\_pred = classifier.predict(X\_test)

print("Predicted vs Actual on Test set")

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

Predicted vs Actual on Test set

[[1. 1.]

[1. 1.]

[1. 1.]

...

[2. 2.]

[1. 1.]

[1. 1.]]

Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

print("Confusion Matrix is :-\n",cm)

Confusion Matrix is :-

[[475 14 3]

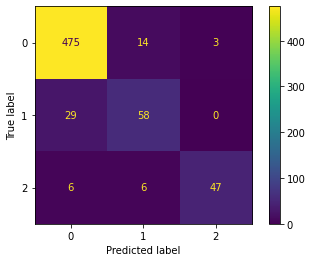
[ 29 58 0]

[ 6 6 47]]

Graphical Confusion Matrix display

from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay(cm).plot()

plt.show()

Accuracy of the KNeighborsClassifier

from sklearn.metrics import accuracy\_score

#print("Accuracy : ",accuracy\_score(y\_test,y\_pred))

print("Accuracy : ",accuracy\_score(classifier.predict(X\_test),y\_test))

Accuracy : 0.9090909090909091

## 4.2.2 THROUGH Logistic Regression

Logistic Regression

Upload dataset

from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():

  print('User uploaded file "{name}" with length {length} bytes'.format(

      name=fn, length=len(uploaded[fn])))

Choose Files fetal\_health.csv

**fetal\_health.csv**(text/csv) - 228715 bytes, last modified: 9/10/2022 - 100% done

Saving fetal\_health.csv to fetal\_health.csv

User uploaded file "fetal\_health.csv" with length 228715 bytes

Importing Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

Loading dataset

ds=pd.read\_csv("fetal\_health\_logistic.csv")

print("Shape of dataset : ",ds.shape)

print("Dataset :-\n",ds.head())

Shape of dataset : (1950, 22)

Dataset :-

baseline value accelerations fetal\_movement uterine\_contractions \

0 120 0.000 0.0 0.000

1 132 0.006 0.0 0.006

2 133 0.003 0.0 0.008

3 134 0.003 0.0 0.008

4 132 0.007 0.0 0.008

light\_decelerations severe\_decelerations prolongued\_decelerations \

0 0.000 0.0 0.0

1 0.003 0.0 0.0

2 0.003 0.0 0.0

3 0.003 0.0 0.0

4 0.000 0.0 0.0

abnormal\_short\_term\_variability mean\_value\_of\_short\_term\_variability \

0 73 0.5

1 17 2.1

2 16 2.1

3 16 2.4

4 16 2.4

percentage\_of\_time\_with\_abnormal\_long\_term\_variability ... histogram\_min \

0 43 ... 62

1 0 ... 68

2 0 ... 68

3 0 ... 53

4 0 ... 53

histogram\_max histogram\_number\_of\_peaks histogram\_number\_of\_zeroes \

0 126 2 0

1 198 6 1

2 198 5 1

3 170 11 0

4 170 9 0

histogram\_mode histogram\_mean histogram\_median histogram\_variance \

0 120 137 121 73

1 141 136 140 12

2 141 135 138 13

3 137 134 137 13

4 137 136 138 11

histogram\_tendency fetal\_health

0 1 2

1 0 1

2 0 1

3 1 1

4 1 1

[5 rows x 22 columns]

Count of each class in Target

count=ds['fetal\_health'].value\_counts()

print("Count of each class in Target :-\n",count)

Count of each class in Target :-

1 1655

2 295

Name: fetal\_health, dtype: int64

Assigning X and y values

X=ds.iloc[:,:-1].values

print("X Shape is : ",X.shape)

print("X is :-\n",X);

X Shape is : (1950, 21)

X is :-

[[1.20e+02 0.00e+00 0.00e+00 ... 1.21e+02 7.30e+01 1.00e+00]

[1.32e+02 6.00e-03 0.00e+00 ... 1.40e+02 1.20e+01 0.00e+00]

[1.33e+02 3.00e-03 0.00e+00 ... 1.38e+02 1.30e+01 0.00e+00]

...

[1.40e+02 1.00e-03 0.00e+00 ... 1.52e+02 4.00e+00 1.00e+00]

[1.40e+02 1.00e-03 0.00e+00 ... 1.51e+02 4.00e+00 1.00e+00]

[1.42e+02 2.00e-03 2.00e-03 ... 1.45e+02 1.00e+00 0.00e+00]]

y=ds.iloc[:,-1].values

print("y Shape is : ",y.shape)

print("y is :-\n",y)

y Shape is : (1950,)

y is :-

[2 1 1 ... 2 2 1]

Splitting the dataset into Train set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 50)

print("Shape of X\_train is : ",X\_train.shape)

print("X\_train is :-\n",X\_train)

Shape of X\_train is : (1462, 21)

X\_train is :-

[[1.45e+02 2.00e-03 0.00e+00 ... 1.56e+02 3.00e+00 1.00e+00]

[1.35e+02 2.00e-03 0.00e+00 ... 1.39e+02 2.00e+00 0.00e+00]

[1.30e+02 1.00e-03 0.00e+00 ... 1.34e+02 1.00e+00 0.00e+00]

...

[1.36e+02 0.00e+00 1.00e-03 ... 1.38e+02 1.00e+00 1.00e+00]

[1.34e+02 0.00e+00 0.00e+00 ... 1.26e+02 6.30e+01 0.00e+00]

[1.37e+02 2.00e-03 5.00e-03 ... 1.39e+02 3.80e+01 1.00e+00]]

print("Shape of y\_train is : ",y\_train.shape)

print("y\_train is :-\n",y\_train)

Shape of y\_train is : (1462,)

y\_train is :-

[2 1 1 ... 1 1 1]

print("Shape of X\_test is : ",X\_test.shape)

print("X\_test is :-\n",X\_test)

Shape of X\_test is : (488, 21)

X\_test is :-

[[1.42e+02 0.00e+00 0.00e+00 ... 1.46e+02 1.00e+00 0.00e+00]

[1.27e+02 7.00e-03 5.00e-03 ... 1.34e+02 7.00e+00 1.00e+00]

[1.37e+02 1.00e-03 0.00e+00 ... 1.42e+02 0.00e+00 0.00e+00]

...

[1.22e+02 5.00e-03 0.00e+00 ... 1.34e+02 1.20e+01 1.00e+00]

[1.22e+02 5.00e-03 0.00e+00 ... 1.34e+02 1.20e+01 1.00e+00]

[1.38e+02 5.00e-03 0.00e+00 ... 1.61e+02 2.70e+01 1.00e+00]]

print("Shape of y\_test is : ",y\_test.shape)

print("y\_test is :-\n",y\_test)

Shape of y\_test is : (488,)

y\_test is :-

[1 1 2 2 1 1 1 1 1 1 1 2 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1

1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 2 1 1 1 1 1 1 1 2 1 2

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 2

1 1 2 1 1 1 2 1 2 2 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 2 1 2 1 1 1 1 1 1 2 1

1 1 1 2 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2

2 2 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 2 1 1 1 1 2 1 1 1 2 1 1 1

1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 2 1 1 2

1 1 2 2 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 2 1 1

2 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 2 1 2 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1

1 1 1 1 1 1 1]

Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

print("Now X\_train is :-\n",X\_train)

Now X\_train is :-

[[ 1.20819127 -0.36734948 -0.18805479 ... 1.30921449 -0.59634702

1.07436477]

[ 0.17853482 -0.36734948 -0.18805479 ... -0.03833249 -0.64179424

-0.62004871]

[-0.3362934 -0.6216416 -0.18805479 ... -0.43466984 -0.68724145

-0.62004871]

...

[ 0.28150047 -0.87593371 -0.16255051 ... -0.11759996 -0.68724145

1.07436477]

[ 0.07556918 -0.87593371 -0.18805479 ... -1.06880959 2.1304858

-0.62004871]

[ 0.38446611 -0.36734948 -0.06053341 ... -0.03833249 0.99430546

1.07436477]]

print("Now X\_test is :-\n",X\_test)

Now X\_test is :-

[[ 0.89929434 -0.87593371 -0.18805479 ... 0.51653979 -0.68724145

-0.62004871]

[-0.64519034 0.90411108 -0.06053341 ... -0.43466984 -0.41455817

1.07436477]

[ 0.38446611 -0.6216416 -0.18805479 ... 0.19946992 -0.73268866

-0.62004871]

...

[-1.16001856 0.39552686 -0.18805479 ... -0.43466984 -0.1873221

1.07436477]

[-1.16001856 0.39552686 -0.18805479 ... -0.43466984 -0.1873221

1.07436477]

[ 0.48743176 0.39552686 -0.18805479 ... 1.70555183 0.49438611

1.07436477]]

Training Logistic Regression model on the Train set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression()

classifier.fit(X\_train, y\_train)

LogisticRegression()

Predicting the Test set results

y\_pred = classifier.predict(X\_test)

print("Predicted vs Actual on Test set")

print(np.concatenate((y\_pred.reshape(len(y\_pred),1), y\_test.reshape(len(y\_test),1)),1))

Predicted vs Actual on Test set

[[1 1]

[1 1]

[1 2]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[2 2]

[1 1]

[1 1]

[2 2]

[1 1]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[2 1]

[1 1]

[1 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[2 1]

[1 1]

[1 1]

[1 1]

[2 2]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]

[1 1]]

Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix, accuracy\_score

print("Confusion Matrix is :-\n",cm)

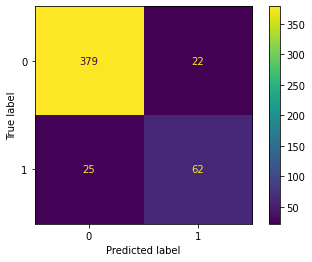
Confusion Matrix is :-

[[379 22]

[ 25 62]]

Graphical Confusion Matrix display

from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay(cm).plot()

plt.show()

Accuracy of the Logistic Model

from sklearn.metrics import accuracy\_score

#print("Accuracy : ",accuracy\_score(y\_test,y\_pred))

print("Accuracy : ",accuracy\_score(classifier.predict(X\_test),y\_test))

Accuracy : 0.9036885245901639

### CHAPTER 5

### RESULTS

**Accuracy tHROUGH KNN**

|  |  |
| --- | --- |
| ML model | Accuracy |
| KNN | 0.9090909090909091 |

**Accuracy FROM Logistic Regression on Partial Data**

|  |  |
| --- | --- |
| ML model | Accuracy |
| Logistic Regression | 0.9036885245901639 |

### CHAPTER 6

### CONCLUSION AND FUTURE SCOPE

### 6.1 CONCLUSION

Finally, after performing all the steps needed to get the results from preparation to preprocessing to performing the models (Logistic regression and KNN) we conclude that the KNN model with 90.90909090909091 percent accuracy performs relatively better than Logistic Regression of 90.36885245901639 percent accuracy.

## 6.2 FUTURE SCOPE

As there is a lot of possibility of improvement in this based on the data as modern real time data can be collected which can be used to test all the different models that are present and to create a new accuracy based on this. Another thing that can be done is to test the model and also create a comparison on the new data. The data collection would take a long time hence till then multiple times the data should be collected from different sources.

### REFERENCES

1. <https://www.karger.com/Article/Fulltext/505021>
2. <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
3. <https://onlinelibrary.wiley.com/doi/full/10.1111/exsy.12899>
4. <https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification>
5. <https://www.researchgate.net/publication/356666279>
6. <https://www.ijert.org/research/fetal-health-prediction-using-classification-techniques-IJERTV10IS110182.pdf>