**EARLY PREDICTION OF SEPSIS USING**

**DECISION TREES**

**DSCI – 6612 – 02**

**INTRO TO ARTIFICIAL INTELLIGENCE**

**Professor:**

**Vahid Behzadan**

**Team Members:**

**Hemanth Pathipati**

**Kavitha Madiraju**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topic** | **Page No.** |
|  | **ABSTRACT** | x |
| **1.** | **INTRODUCTION** | 1 |
|  | 1.1 Motivation | 2 |
|  | 1.2 Problem Definition | 2 |
|  | 1.3 Objective of the Project | 3 |
|  | 1.4 Feasibility study | 3 |
|  | 1.5 Organization of Documentation | 4 |
| **2.** | **PROBLEM ANALYSIS** | 5 |
|  | 3.1 Existing approach | 6 |
|  | 3.1.1 Drawbacks | 6 |
|  | 3.2 Proposed System | 6 |
|  | 3.2.1 Advantages | 7 |
|  | 3.3 Software and Hardware Requirements | 7 |
|  | 3.4 Architecture | 8 |
|  | 3.5 Algorithm | 8 |
| **3.** | **PROCESS MODEL** | 12 |
|  | 4.1 Data Collection & Preprocessing | 13 |
|  | 4.2 Exploratory Data Analysis | 15 |
|  | 4.3 Feature Engineering | 15 |
|  | 4.4 Feature Selection | 17 |
|  | 4.5 Model Building & Training | 17 |
| **4.** | **EVALUATION METRICS** | 18 |
| **5.** | **IMPLEMENTATION** | 21 |
| **6.** | **EXPERIMENTAL RESULTS** | 34 |
| **7.** | **CONCLUSION** | 42 |
|  | **REFERENCES** | 44 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Fig. No.** | **Name of the figure** | **Page No.** |
| 1 | Architecture of Proposed Model | 8 |
| 2 | Working model of Decision Tree | 9 |
| 3 | Data obtained after performing Feature Engineering | 16 |
| 4 | Data obtained after performing Label Encoding | 17 |
| 5 | Percentage of people affected by Sepsis | 35 |
| 6 | Bar Chat of People Effected with Sepsis | 35 |
| 7 | Histogram of Features | 36 |
| 8 | Pair plots (a) | 37 |
| 9 | Pair plots (b) | 38 |
| 10 | Pair plots (c) | 39 |
| 11 | Pair plots (d) | 40 |
| 12 | Output of Decision Tree | 41 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No.** |
| 1 | Vital Signs | 13 |
| 2 | Laboratory Values | 14 |
| 3 | Body Demographics | 15 |
| 4 | List of columns with less than 92% missing values | 15 |

**ABSTRACT**

Sepsis is a challenging problem in the field of medical research which leads to death if not diagnosed early. The diagnosis of sepsis is not easy, because the signs and symptoms of sepsis are similar to those of normal health issues like Shivering, fever, or very cold, Extreme pain or general discomfort (“worst ever”), Pale or discoloured skin, headache etc. To address this problem, we are proposing a Decision Trees architecture which uses the medical data to predict the sepsis. It uses clinical variables like Bilirubin, Fibrinogen and other lab values. In addition to this it also uses sepsis criteria to the prediction task. With maximum depth and leaf nodes, decision trees are implemented using two functions: Gini index and entropy. As a result, prediction accuracy improves more quickly than test results, which require longer to validate.

*Key words*: Sepsis, Machine learning, prediction, cohorts, decision trees, assessment metrics, accuracy.

**CHAPTER 1**

**INTRODUCTION**

Sepsis is a major public health concern that causes significant morbidity, mortality, and healthcare costs. Early detection and antibiotic treatment enhance sepsis outcomes. Researchers have created algorithms for early sepsis detection in response to this, but due to differences in patient demographics, clinical characteristics and sepsis criteria [1], prediction tasks, assessment measures, and other aspects, direct comparison of such systems has proven problematic. Sepsis is a life-threatening organ failure caused by an abnormal host response to infection. This can be ascribed in part to difficulties in detecting sepsis early and initiating timely and adequate therapy. A major barrier to early detection is distinguishing sepsis from sickness symptoms (e.g., inflammation) that have similar clinical signs (e.g., change in vitals), symptoms (e.g., fever), and molecular manifestations (e.g., dysregulated host response). Clinical research suggests that SEPSIS is more common among very young children, older adults, people with chronic illnesses, and people with a weakened immune system.

* 1. ***Motivation***

The main motivation for this project is to reduce the mortality rate using machine learning techniques. This helps medical labs to predict whether the patient is affected by SEPSIS or not within less amount of time compared to manual evaluation such that the patient can be treated with proper medication.

* 1. ***Problem Definition***

Sepsis is a potentially fatal illness that develops when the body's response to infection results in tissue damage, organ failure, or death. Nearly 1.7 million individuals in the United States get sepsis each year, and 270,000 people die from it; sepsis kills nearly one-third of those who die in American hospitals. Sepsis affects an estimated 30 million individuals worldwide each year, with 6 million deaths; 4.2 million babies and toddlers are afflicted. Sepsis costs hospitals in the United States more than any other health condition, costing $24 billion per year (13 percent of total healthcare spending), with the majority of these costs incurred by patients who were not diagnosed with sepsis at the time of admission. Globally, the cost of sepsis is much higher, with the developing world being the most vulnerable. Sepsis is a severe public health problem that causes significant morbidity, mortality, and healthcare costs. Early detection and antibiotic treatment of sepsis are crucial for bettering sepsis outcomes, with each hour of delayed treatment associated with a 48% increase in death. Clinicians have proposed new definitions for sepsis to assist address this challenge, but the fundamental requirement to detect and treat sepsis early remains unmet, sand basic questions concerning the limits of early detection remain unaddressed.

* 1. ***Objective of the Project***

The main objectives of this project are:

* To study the various existing methods used in the literature for early detection of sepsis.
* To propose an efficient classification model for early prediction of sepsis.
* To analyze the proposed system with state-of-art existing algorithms.
* To increase the accuracy of the prediction model.
  1. ***Feasibility Study***

In this stage the feasibility of the venture is examined, and the strategic agreement is advanced with extremely broad arrangement for undertaking and few quotes. During framework investigation the achievability investigation of the proposed framework is to be done. This is to ensure that the proposed approach will not be a burden to the company. For possibility investigation, some comprehension of the significant prerequisites for the framework is fundamental. Three key contemplations engaged with feasibility investigation are:

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This evaluation is done to determine the framework's financial impact on the organization. The number of resources that an organization can devote to the framework's innovative work is limited. Consumption should be encouraged. As a result, the constructed framework kept within budget, which was made possible by the fact that a large portion of the innovations were created in-house and are publicly available. Only the products that have been tweaked must be purchased.

**TECHNICAL FEASIBILITY**

This examination is done to determine the framework's specific capability, i.e., the framework's specialized requirements. Any framework built should not rely on the availability of specialist resources. This will result in a surge in demand for the available specialized assets. As a result, the customer will be subjected to a barrage of requests. The constructed framework should have an unassuming necessity since it only requires minor or faulty alterations to be implemented.

**SOCIAL FEASIBILITY**

The purpose of this study is to determine the client's level of acceptance of the framework. This comprises the method for preparing the client to effectively use the framework. The client should not feel threatened by the framework, but rather recognize that it is not a need. The tactics used to teach the client about the framework and familiarize him with it are solely responsible for the level of acceptance by the clients. As he is the Payable Outsourced Decryption for Functional Encryption Using Block Chai framework's final client, his level of certainty should be elevated such that he is also capable of doing some useful analysis, which is encouraged.

* 1. ***Organization of Documentation***

The rest of the document is organized as follows Chapter 2 includes Literature survey, Chapter 3 consists of Problem analysis followed by Chapter 4 with Process Model. Chapter 5 includes Evaluation Metrics, Chapter 6 consists of Implementation followed by Chapter 7 with Experimental Results. Finally, Chapter 8 contains Conclusion of the project.

**CHAPTER 2**

**PROBLEM ANALYSIS**

***3.1 Existing Approach***

The current existing method mainly relies on XGBoost, Challenge Algorithm scoring, Neural Network algorithms. The missing values are handled by categorically encoding the features. Features with missingness greater than 92% are removed. In case if there are missing values that are highly important for the model [2] classification, it is imputed by using the interpolation techniques and Linear Dynamic Systems (LDS). The feature selection is done by checking the correlation of feature contributions to the presence of sepsis. Data imbalance is handled using a balanced bagging classifier which automatically create balanced samples of the input data. Even though the data is processed clearly, in the RNN fixed sliding window it outperformed other models with an AUC of 0.82 and precision of 0.21 In few implementations XGBoost algorithm is used in the training phase to boost the speed of the training process as it is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. Earlier studies used the MIMIC-3 datasets,[4] the researchers to predict the SOFA (Sequential Organ Failure Assessment) score. To detect sepsis in patients, they utilized an ensemble machine learning models comprised of forest-based implementations. The other implementation [5] used Retrospective analysis to extract data from Electronic Health Records (EHR). The comparison to Logistic Regression assured 92% of accuracy.

***3.1.1 Drawbacks***

1. A few studies employed neural network-based techniques, which take a long time to generate nodes and provide no information about the structure of the function being approximated.
2. XGBoosters are hardly scalable. Gradient boosting is extremely sensitive to outliers since each classifier is obliged to correct the errors made by the previous learners.
3. When using Random Forest (many trees), the process may become too sluggish and useless for real-time predictions. Despite the fact that this algorithm is too fast to train, it is fairly slow to generate predictions once trained.
4. The complexity of algorithmic operations and functions is determined by hardware configurations.
5. xLGBM causes overfitting since it generates more complicated trees.

***3.2 Proposed System***

This work provides data processing methods and evaluates the efficacy of feature engineering by label encoding and combining two or more features to produce a new feature that successfully improves prediction outcomes. Instead of using typical imputation strategies to deal with missing data values, label encoding is used for the best outcomes. Decision Tree has a greater categorization capacity with the addition of the Gini index and entropy functions. The Gini index estimated the likelihood of a randomly picked feature being mistakenly categorized, whereas the entropy function could measure the level of uncertainty in the value of a random variable or the result of a random process.

***3.2.1 Advantages***

The advantages of this project are:

1. Easy to comprehend and interpret.
2. Despite the fact that it does not handle missing values, it requires minimal data preparation. Other procedures frequently need data standardization, the creation of dummy variables, and the removal of NAN values.
3. The cost of utilizing the tree (predicting data) is inversely proportional to the quantity of data points utilized to train the tree.
4. Statistical tests can be used to validate a model. This allows us to account for the model's accuracy.
5. Performs well even when the real model from which the data were created violates certain of its assumptions.
6. It takes less time to categorize the features, which reduces the execution time.

***3.3 Software & Hardware Requirements***

SOFTWARE REQUIREMENTS:

Operating system: Windows 7/Windows 8/Windows 10/Ubuntu

Tools: Jupyter Notebook, Anaconda Navigator

HARDWARE REQUIREMENTS:

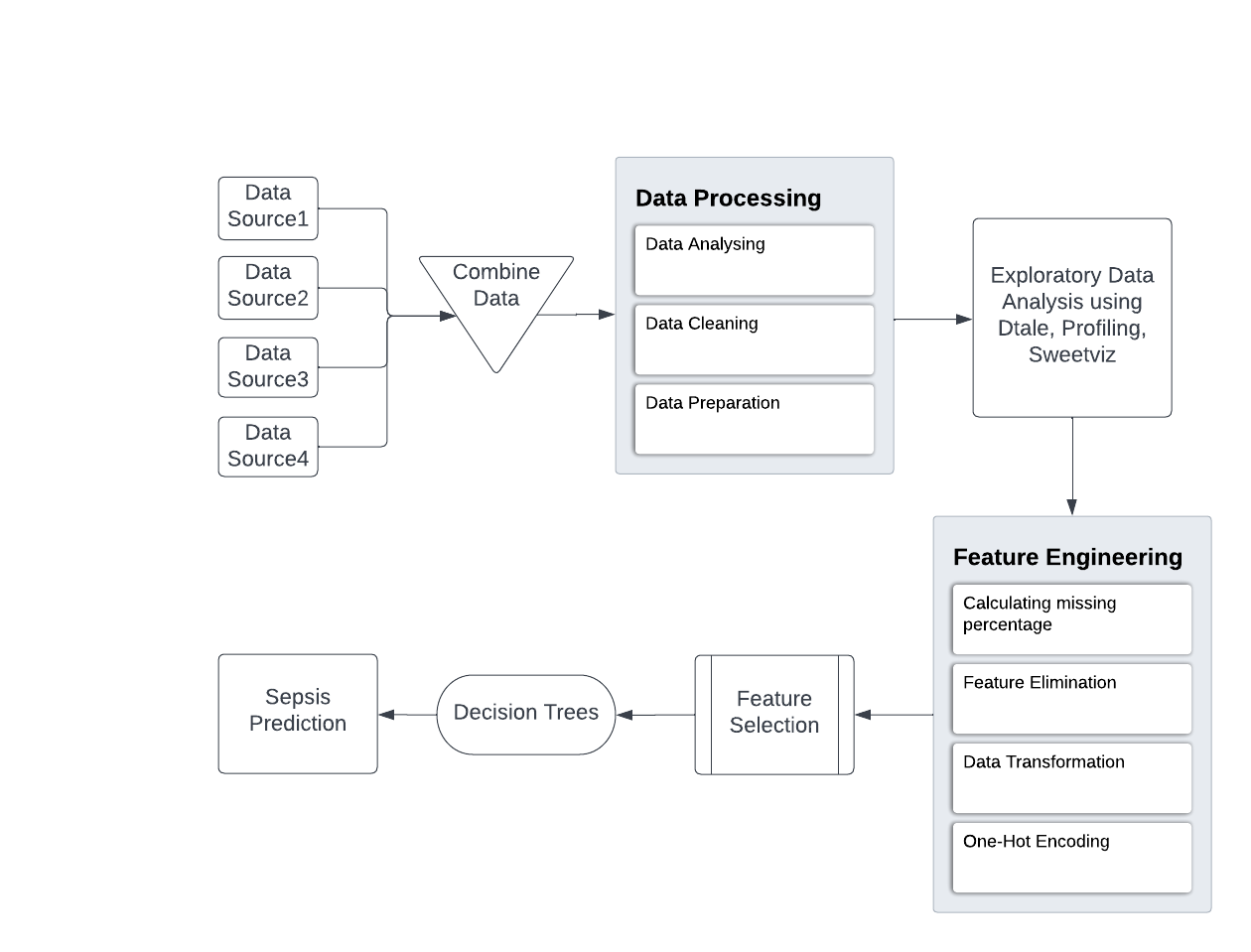
Processor: Intel i5

GPU: RTX 3050Ti

RAM: 8GB (minimum)

Hard Disk: 256GB

***3.4 Architecture***



**Fig-1: Architecture of proposed model**

***3.5 Algorithms***

**Decision Tree Classification Algorithm:**

Decision Trees can be used for classification as well as regression problems. In our case, we are using decision trees [10] for classifying the features.

Classification involves two steps:

Learning step (development of model based on training data)

Prediction step (prediction of response for given data)

We start from the root of the tree when using Decision Trees to forecast a class label for a record. The record's attribute and the root attribute's values are compared. Based on the comparison, we follow the branch that corresponds to that value and jump to the next node. In decision trees, the internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome. Decision Tree asks a question and based on the response(yes/no), it further split the tree into subtrees.

**General structure of decision tree:**



**Fig-2: Working Model of Decision Tree**

**Types of decision trees:**

Decision trees are categorized according to the type of target variable. It can be of two types:

* **Categorical Variable Decision Tree**: Decision Tree which has a categorical target variable then it called a Categorical variable decision tree.
* **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree.

**Working of decision tree algorithm:**

Decision trees use a variety of ways to determine whether to split a node into two or more sub-nodes. As sub-nodes are produced, the homogeneity of the resulting sub-nodes increases. To put it another way, when the target variable increases, the node's purity improves. The decision tree splits the nodes into subnodes based on all available variables and then chooses the split that produces the most homogeneous sub-nodes.

Steps:

1. Beginning with the root node which consists of complete dataset.
2. Finding the best attribute in the dataset using Attribute Selection Measure (ASM).
3. Divide the root node into subsets that contains possible values for the best attributes.
4. Decision tree is generated, which contains the best attribute.
5. Recursively repeat the process from step 3 to 5 until there is no further classification of nodes and the last node is called as leaf node.

**Attribute Selection Measures (ASM):**

When the dataset comprises N properties, deciding which attribute to place at the root or at different levels of the tree as internal nodes is a challenging task. The problem cannot be solved by choosing any node as the root at random. We may end up with bad results and low precision if we use a random technique.

To solve such issues, we have a technique called Attribute Selection Measures or ASM. Using ASM, we can easily select the best attribute for the nodes of the tree.

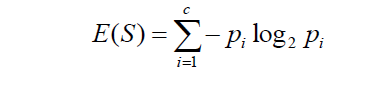
The criteria for ASM are:

* Entropy
* Information gain
* Gini Index

These criteria will be used to calculate the value of each feature. The values are sorted, and the features are arranged in a tree, with the highest-valued attribute (in this case, information gain) at the top. When using Information Gain as a criterion, we assume categorical qualities, while when using the Gini index, we assume continuous attributes.

**Entropy:**

Entropy quantifies the randomness of the information being processed. The higher the entropy, the more difficult it is to make any conclusions from the data. A coin flip is an example of an action that produces random information. Mathematical formula for entropy:

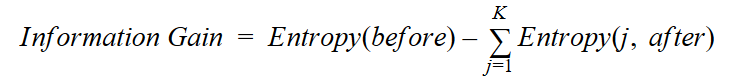


Where ‘Pi’ is simply the frequentist probability of an element/class ‘i’ in our data. For simplicity’s sake let’s say we only have two classes, a positive class and a negative class. Therefore ‘i’ here could be either + or (-).

**Information Gain:**

Information gain (IG) is a statistical feature that measures how well a specific variable differentiates training instances based on its categorization criteria. The key to building a decision tree is to choose a feature with the most information gain and the lowest entropy. It computes the difference between the dataset's entropy before and after splitting based on the specified attribute values.

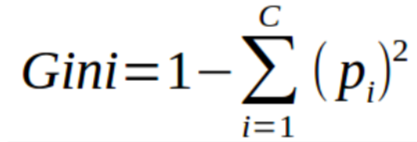
Mathematical expression:



Where “before” is the dataset before the split, K is the number of subsets generated by the split, and (j, after) is subset j after the split.

**Gini Index:**

The Gini index is a cost function that is used to evaluate splits in a dataset. It is calculated by subtracting the sum of the squared probability of each class from one. It chooses larger partitions with simple implementation, whereas information gain prefers smaller partitions with varying values.



The Gini Index incorporates the categorical target variable "Success" or "Failure." It only performs binary splits. A higher Gini index value indicates greater inequality and heterogeneity.

**Steps to Calculate Gini index for a split:**

1. Calculate Gini for sub-nodes, using the above formula for success(p) and failure(q) (p²+q²).
2. Calculate the Gini index for split using the weighted Gini score of each node of that split.

CART (Classification and Regression Tree) uses the Gini index method to create split points.

**CHAPTER 3**

**PROCESS MODEL**

This project is majorly divided into six modules or sub-projects. They are:

* Data Collection and Pre-processing
* Exploratory Data Analysis
* Feature Engineering
* Feature Selection
* Model Building and Training
* Model Validation

***4.1 Data Collection and Pre-Processing***

It is the first and foremost step of any Machine Learning or Deep Learning project. While performing the literature survey from IEEE Journals we came across the datasets like Physionet Clinical, National Health Care Group, MIMIC-III etc. But we choose Physionet Clinical dataset as it consists vital signs [Table-1], Laboratory values [Table-2], and body demographics [Table-3] and sepsis label, totally it consists of 41 features. In the preprocessing step we have combined all the psv filles into a single csv file.

***Vital signs (columns 1-8)***

|  |  |
| --- | --- |
| HR | Heart rate (beats per minute) |
| O2Sat | Pulse oximetry (%) |
| Temp | Temperature (Deg C) |
| SBP | Systolic BP (mm Hg) |
| MAP | Mean arterial pressure (mm Hg) |
| DBP | Diastolic BP (mm Hg) |
| Resp | Respiration rate (breaths per minute) |
| EtCO2 | End tidal carbon dioxide (mm Hg) |

**Table-1: Vital Signs**

***Laboratory values (columns 9-34)***

|  |  |
| --- | --- |
| BaseExcess | Measure of excess bicarbonate (mmol/L) |
| HCO3 | Bicarbonate (mmol/L) |
| FiO2 | Fraction of inspired oxygen (%) |
| pH | 0-14 |
| PaCO2 | Partial pressure of carbon dioxide from arterial blood (mm Hg) |
| SaO2 | Oxygen saturation from arterial blood (%) |
| AST | Aspartate transaminase (IU/L) |
| BUN | Blood urea nitrogen (mg/dL) |
| Alkalinephos | Alkaline phosphatase (IU/L) |
| Calcium | (mg/dL) |
| Chloride | (mmol/L) |
| Creatinine | (mg/dL) |
| Bilirubin Direct | Bilirubin direct (mg/dL) |
| Glucose | Serum glucose (mg/dL) |
| Lactate | Lactic acid (mg/dL) |
| Magnesium | (mmol/dL) |
| Phosphate | (mg/dL) |
| Potassium | (mmol/L) |
| Bilirubin\_total | Total bilirubin (mg/dL) |
| TroponinI | Troponin I (ng/mL) |
| Hct | Hematocrit (%) |
| Hgb | Hemoglobin (g/dL) |
| PTT | partial thromboplastin time (seconds) |
| WBC | Leukocyte count (count\*10^3/µL) |
| Platelets | (count\*10^3/µL) |

**Table-2: Laboratory Values**

**Body Demographics (columns 35-40)**

***Body Demographics (35-40)***

|  |  |
| --- | --- |
| Age | Years (100 for patients 90 or above) |
| Gender | Female (0) or Male (1) |
| Unit1 | Administrative identifier for ICU unit (MICU) |
| Unit2 | Administrative identifier for ICU unit (SICU) |
| HospAdmTime | Hours between hospital admit and ICU admit |
| ICULOS | ICU length-of-stay (hours since ICU admit) |

**Table-3: Body Demographics**

***4.2 Exploratory Data Analysis***

In the EDA part we have studied the data set to know about the hidden patterns and wisdom hidden in the data. We have used Histograms, pair plots and correlation matrix to gain insights from the data like how data is distributed, how features are correlated and the relationships between the features. Now-a-days automated tools like Dtale, SweetViz, AutoViz, Pandas Profiling are available for the EDA process we have also utilized those to cross verify our observations. After the EDA part we understood the data patterns and the distribution of data over the labels. The insights we obtained are the data contains 98% of sepsis negative data and 2% of sepsis positive data. Due to this an imbalance exists in the data.

***4.3 Feature Engineering***

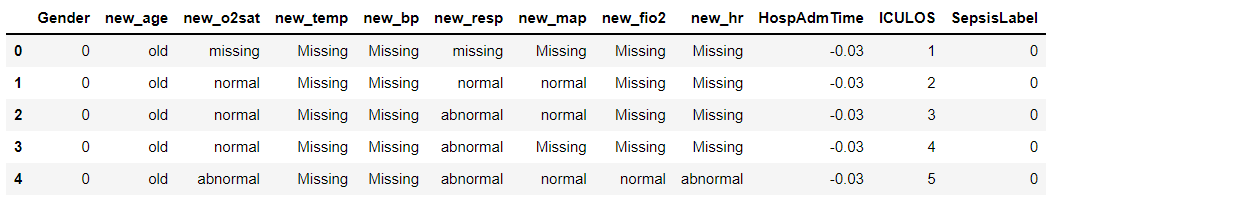
The majority portion of the data have missing values, some features contain more than 92 % of missing values. So, we have considered the features which consists the missing rate less than 92%. The features which are having missing rate less than 92% are shown in Table-4.

|  |  |
| --- | --- |
| HR | 7.743297 |
| O2Sat | 12.034000460903016 |
| Temp | 66.22543222319862 |
| SBP | 15.214990997196596 |
| MAP | 10.234072888519716 |
| DBP | 48.137774547517324 |
| Resp | 9.777854876328579 |
| BaseExcess | 89.57751401069194 |
| HCO3 | 91.94913859252375 |
| FiO2 | 85.81203767249043 |
| pH | 88.53491087098007 |
| PaCO2 | 91.23537202723885 |
| BUN | 91.84024392405736 |
| Chloride | 91.67538246086757 |
| Glucose | 87.76859629707478 |
| Potassium | 89.1385165507234 |
| Hct | 88.22316822706817 |
| Hgb | 91.16408400590564 |
| Age | 0.0 |
| Gender | 0.0 |
| Unit1 | 48.86002476720599 |
| Unit2 | 48.86002476720599 |
| HospAdmTime | 0.0010129736601524018 |
| ICULOS | 0.0 |
| SepsisLabel | 0.0 |

**Table-4 List of Columns with less than 92% missing values**

We use only these features for further transformations in the pipeline. We have converted the numerical data into categorical data by classifying the features into different labels. In this process we have considered the numerical ranges to classify into class labels.

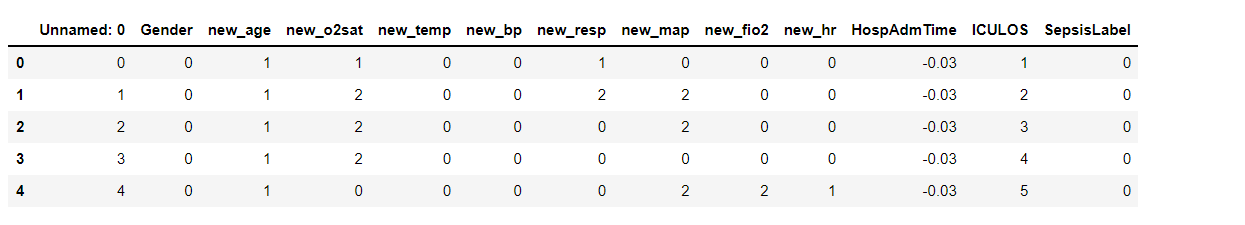
Age is classified into 3 classes as Infant [0-10], Adult [11-60] and old [>60]. Heart rate is divided into 2 classes as Normal and Abnormal by considering the age factor. O2sat is segregated into 2 classes Normal [95-100] and Abnormal [0-95]. Temperature is labelled into Normal [36-38] and Abnormal. The blood pressure is break downed into 4 classes as Low, Normal, Elevated and High by considering the SBP and DBP values. The respiration rate is grouped into 2 classes as Normal and Abnormal by considering the age and resp values. Mean Arterial Pressure is divided into 2 classes as Normal [70-100] and Abnormal. FIO2 is categorized into 2 classes as Normal [0-0.8] and Abnormal [>=0.8]. In the above categorization of data, the missing values are encoded as class named as “Missing”. The outcome of this process looks like below table



**Fig-3: Data obtained after performing Feature Engineering**

***4.4 Feature Selection***

From the transformed features we consider the features which are significant [paper reference number] by nature. Along with these features we have considered the other features which play a significant role in the prediction process like Gender, HospAdmitTime and ICULOS. From this data we have performed the label encoding to convert the categorical data into feasible numerical data. The resulting data format is given in the below table

****

**Fig-4: Data obtained after performing Label Encoding**

***4.5 Model Building and Training***

In this step the model is built using the prerequisites of the problem statement which are formulated. The model building process is then followed by model training phase in which the model built is trained using the processed dataset. In the training phase the model will learn about the problem with the help of data. The knowledge about the problem is represented using statistical and mathematical modelling which is then used by the model to get information about the problem.

For forecasting the sepsis label, we use Decision Tree, which has three functions: gini, entropy, and predict. We calculate the gini value using the gini function, which is then used to calculate accuracy. The DecisionTreeClassifier function is used in the entropy function to determine the entropy value. The accuracy is calculated by comparing the predicted values to the original values using the return values of the gini and entropy functions. Finally, we compute two values: one using the Gini index and the other using the entropy.

**CHAPTER 4**

**EVALUATION METRICS**

The model performance is to be evaluated using performance matrices other than Accuracy. Monitoring only accuracy gives an incomplete picture of model’s performance and can impact the effectiveness.

***Confusion matrix:***

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. “The confusion matrix shows the ways in which our classification model is confused when it makes prediction.”

* It gives us insight not only into the errors being made by your classifier but more importantly the types of errors that are being made
* In it, we have four classes:
  + - * 1. True Positive - for correctly predicting event values
        2. False Positive - for incorrectly predicting event values
        3. True Negative - for correctly predicting no-event values
        4. False Negative - for incorrectly predicting no-event values

|  |  |  |
| --- | --- | --- |
|  | Predicted No | Predicted Yes |
| Actual No | True Negative | False Positive |
| Actual Yes | False Negative | True Positive |

***Sensitivity***

Sensitivity is the percentage of actual one’s that were correctly predicted. It shows the percentage of ones that were correctly predicted by the model. It is also called as true positive rate or recall.

Sensitivity = TP/Actual Yes

***Specificity***

It is the percentage or proportion of zeros that are correctly predicted. It is also called as True Negative Rate.

Specificity = TN/ Actual No

***Detection Rate***

It is the proportion of the whole sample where the events were detected correctly. It reflects the accuracy of the model.

***Precision***

Precision is the number of true positives divided by the total number of positive predictions.

Precision = TP/ predicted Yes

***ROC Curve***

A ROC (Receive Operating Characteristic Curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

* True Positive Rate
* False Positive Rate

**CHAPTER 5**

**IMPLEMENTATION**

### **Data Collection:**

### fout = open("out.csv","a")

### # first file:

### for line in open('sh1.csv'):

### fout.write(line)

### # now the rest:

### for num in range(2,201):

### f = open("sh"+str(num)+".csv")

### f.next()

### for line in f:

### fout.write(line)

### f.close()

### fout.close()

### **Data Preprocessing:**

### import pandas as pd

### import numpy as np

### import matplotlib.pyplot as plt

### import seaborn as sns

### df = pd.read\_csv("data.psv", delimiter="|")

### df.columns

### df1 = df['SepsisLabel'].value\_counts()

### no\_sepsis = (df1[0]/(df1[0]+df1[1]))\*100

### yes\_sepsis = (df1[1]/(df1[0]+df1[1]))\*100

### labels = 'No Sepsis', 'Sepsis'

### sizes = [no\_sepsis, yes\_sepsis]

### explode = (0, 0.1)

### fig1, ax1 = plt.subplots()

### ax1.pie(sizes, explode = explode, labels = labels, autopct = '%1.0f%%', shadow=True, startangle = 90)

### ax1.axis('equal')

### plt.show()

### plt.figure(figsize=(8, 8))

### sns.countplot('SepsisLabel', data=df)

### plt.title('Unbalanced Classes')

### plt.show()

### **Exploratory Data Analysis:**

### import numpy as np

### import pandas as pd

### import matplotlib.pyplot as plt

### df= pd.read\_csv("Data.psv",delimiter='|')

### df.head()

### df.columns

### df[['HR', 'O2Sat', 'Temp', 'SBP', 'MAP', 'DBP', 'Resp', 'EtCO2',

### 'BaseExcess', 'HCO3', 'FiO2', 'pH', 'PaCO2', 'SaO2', 'AST', 'BUN',

### 'Alkalinephos', 'Calcium', 'Chloride', 'Creatinine', 'Bilirubin\_direct',

### 'Glucose', 'Lactate', 'Magnesium', 'Phosphate', 'Potassium',

### 'Bilirubin\_total', 'TroponinI', 'Hct', 'Hgb', 'PTT', 'WBC',

### 'Fibrinogen', 'Platelets', 'Age', 'Gender', 'Unit1', 'Unit2',

### 'HospAdmTime', 'ICULOS', 'SepsisLabel']].hist(bins=20, figsize=(15,15))

### plt.savefig('hist.png')

### plt.show()

### import seaborn as sns

### sns.pairplot(df[['HR', 'O2Sat', 'Temp', 'SBP', 'MAP','SepsisLabel']],hue='SepsisLabel', diag\_kind='kde',kind='scatter',palette='husl')

### plt.savefig('HR,O2Sat,Temp,SBP,MAP,SepsisLabel.png')

### plt.show()

### sns.pairplot(df[['FiO2', 'pH', 'PaCO2', 'SaO2', 'AST', 'BUN',

### 'Alkalinephos', 'Calcium', 'Chloride', 'Creatinine','SepsisLabel']],hue='SepsisLabel',diag\_kind='kde',kind='scatter',palette='husl')

### plt.savefig('FiO2,pH,PaCO2,SaO2,AST,BUN,Alkalinephos,Calcium,Chloride,Creatinine,SepsisLabel.png')

### plt.show()

### sns.pairplot(df[['Bilirubin\_direct',

### 'Glucose', 'Lactate', 'Magnesium', 'Phosphate', 'Potassium',

### 'Bilirubin\_total', 'TroponinI', 'Hct', 'Hgb','SepsisLabel']],hue='SepsisLabel',diag\_kind='kde',kind='scatter',palette='husl')

### plt.savefig('Bilirubin\_direct,Glucose,Lactate,Magnesium,Phosphate,Potassium,Bilirubin\_total,TroponinI,Hct,Hgb,SepsisLabel.png')

### plt.show()

### sns.pairplot(df[['PTT', 'WBC',

### 'Fibrinogen', 'Platelets', 'Age',

### 'HospAdmTime', 'ICULOS','SepsisLabel']],hue='SepsisLabel',diag\_kind='kde',kind='scatter',palette='husl')

### plt.savefig('PTT,WBC,Fibrinogen,Platelets,Age,HospAdmTime,ICULOS,SepsisLabel.png')

### plt.show()

### **Feature Engineering:**

### import pandas as pd

### import numpy as np

### from sklearn.pipeline import Pipeline

### from sklearn.pipeline import make\_pipeline

### from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder

### from sklearn.impute import SimpleImputer

### from sklearn.linear\_model import LogisticRegression

### from sklearn.svm import SVC

### from sklearn.neighbors import KNeighborsClassifier

### from sklearn.tree import DecisionTreeClassifier, export\_graphviz

### from sklearn.compose import make\_column\_transformer, ColumnTransformer

### from sklearn.metrics import precision\_score, accuracy\_score, recall\_score, \

### average\_precision\_score, precision\_recall\_curve, confusion\_matrix

### import seaborn as sns

### from subprocess import call

### from IPython.display import Image

### import warnings

### warnings.filterwarnings('ignore')

### import matplotlib.pyplot as plt

### %matplotlib inline

### plt.style.use('ggplot')

### from pandas.plotting import scatter\_matrix

### original\_data = pd.read\_csv('data.psv', sep ='|')

### missing\_data = original\_data.isnull().sum()

### missing\_percent = (missing\_data/original\_data.shape[0])\*100

### refined\_columns = list(missing\_percent[missing\_percent < 92].index)

### sepsis\_data = original\_data[refined\_columns]

### sepsis\_data.head()

### sepsis\_data.describe()

# Feature Engineering - Building newer features

## 1. Age

Three categories -

Child - Age less than 10 year

Adult - Age more than 10 year and less than 60 years

Senior - Age more than 60

def fe\_new\_age(data):

data.loc[data['Age'] >=60, 'new\_age'] = 'old'

data.loc[data['Age'] <10, 'new\_age'] = 'infant'

data.loc[(data['Age'] >=10) & (data['Age'] <60), 'new\_age'] = 'adult'

return data

### sepsis\_data = fe\_new\_age(sepsis\_data)

## 2. Heart Rate

The new feature designed for heart rate takes into account both Age and Heart Rate in a patient. It has three categories - normal, abnormal, missing

The 'normal' HR for a child (Age < 10) is in the range of 70 to 110 beats per minute.

The 'normal' HR for a adult and senior (Age 10+) is in the range of 60 to 100

Any other values recorded is marked as 'abnormal'.

The value 'missing' is filled in place of null/nan values

def fe\_new\_hr(data):

data.loc[(data['HR'] >= 70) & (data['HR'] < 110 ) & (data['Age'] < 10), 'new\_hr'] = 'normal'

data.loc[(data['HR'] > 60) & (data['HR'] < 100) & data['Age'] >= 10, 'new\_hr'] = 'normal'

data.loc[((data['HR'] < 70) | (data['Age'] >= 110)) & (data['Age']<10), 'new\_hr'] = 'abnormal'

data.loc[(data['HR'] >= 100) & (data['Age'] >= 10), 'new\_hr'] = 'abnormal'

data['new\_hr'].fillna('Missing', inplace=True)

return data

sepsis\_data = fe\_new\_hr(sepsis\_data)

## 3. O2Sat

The blood oxygen level measured with an oximeter is called your oxygen saturation level. This is a percentage of how much oxygen your blood is carrying compared to the maximum it is capable of carrying.

The new feature designed for pulse oximetry takes into three catogories

'Normal' is found to be between 95% - 100% in healthy children and adults alike

'Abnormal' is for anything otherwise

'Missing' is a null or nan case is observed

def fe\_new\_o2sat(data):

data.loc[(data['O2Sat'] >= 95) & (data['O2Sat'] < 100), 'new\_o2sat'] = 'normal'

data.loc[(data['O2Sat'] < 95) & (data['O2Sat'] >= 0), 'new\_o2sat'] = 'abnormal'

data['new\_o2sat'].fillna('missing', inplace=True)

return data

sepsis\_data = fe\_new\_o2sat(sepsis\_data)

## 4. Temperature

The new feature designed for temperature takes into three categories:

Body temperature for any healthy person (child, adult and senior alike) is 'normal' when found between 36 Deg C to 38 Dec C.

Anything above or below this range is labeled as 'abnormal'

'Missing' is a null or nan case is observed

def fe\_new\_temp(data):

data.loc[(data['Temp'] >= 36) & (data['Temp'] < 38),'new\_temp'] = 'normal'

data.loc[(data['Temp'] < 36) | (data['Temp'] >= 38),'new\_temp'] = 'abnormal'

data['new\_temp'].fillna('Missing', inplace=True)

return data

sepsis\_data = fe\_new\_temp(sepsis\_data)

## 5. Blood Pressure

We will be combining two forms of Blood Pressure here - Systolic blood pressure (SBP) and Diastolic Blood Pressure (DBP) in the dataset.

SBP - When your heart beats, it squeezes and pushes blood through your arteries to the rest of your body. This force creates pressure on those blood vessels, and that's your systolic blood pressure

DBP - The diastolic reading, or the bottom number, is the pressure in the arteries when the heart rests between beats. This is the time when the heart fills with blood and gets oxygen.

The new feature will compare the two BP and according to the below table categorize into four categories - low, normal, elevated and high, and missing

def fe\_new\_bp(data):

data.loc[(data['SBP'] < 90) & (data['DBP'] < 60), 'new\_bp'] = 'low'

data.loc[(data['SBP'].between(90,120, inclusive=True)) & (data['DBP'].between(60,80, inclusive=True)), 'new\_bp'] = 'normal'

data.loc[(data['SBP'].between(120,140, inclusive=True)) & (data['DBP'].between(80,90, inclusive=True)),'new\_bp'] = 'elevated'

data.loc[(data['SBP'] > 140 ) & (data['DBP'] > 90 ), 'new\_bp'] = 'high'

data['new\_bp'].fillna('Missing', inplace=True)

return data

sepsis\_data = fe\_new\_bp(sepsis\_data)

## 6. Respiration Rate

The new feature designed will have 3 categories - normal, abnormal and missing. The normal respiratory rate for different age groups are as shown below:

For healthy adults (Age > 18) is between 12 and 20 breaths per minute. Normal respiratory rates for children in breaths per minute are as follows:

birth to 1 year: 30 to 60

1 to 3 years: 24 to 40

3 to 6 years: 22 to 34

6 to 12 years: 18 to 30

12 to 18 years: 12 to 16

Any other range for respiratory rates are labeled as 'abnormal' and the missing values are labeled as 'missing'

def fe\_new\_resp(data):

data.loc[(data['Resp'].between(30, 60)) & (data['Age'] < 1), 'new\_resp'] = 'normal'

data.loc[(data['Resp'].between(24, 40)) & (data['Age'].between(1, 3)), 'new\_resp'] = 'normal'

data.loc[(data['Resp'].between(22, 34)) & (data['Age'].between(3, 6)), 'new\_resp'] = 'normal'

data.loc[(data['Resp'].between(18, 30)) & (data['Age'].between(6, 12)), 'new\_resp'] = 'normal'

data.loc[(data['Resp'].between(12, 16)) & (data['Age'].between(12, 18)), 'new\_resp'] = 'normal'

data.loc[(data['Resp'].between(12, 20)) & (data['Age'] > 18), 'new\_resp'] = 'normal'

data.loc[((data['Resp'] < 30) | (data['Resp'] > 60)) & (data['Age'] <1) ,'new\_resp'] = 'abnormal'

data.loc[((data['Resp'] < 24) | (data['Resp'] > 40)) & (data['Age'].between(1, 3)) ,'new\_resp'] = 'abnormal'

data.loc[((data['Resp'] < 22) | (data['Resp'] > 34)) & (data['Age'].between(3, 6)) ,'new\_resp'] = 'abnormal'

data.loc[((data['Resp'] < 18) | (data['Resp'] > 30)) & (data['Age'].between(6, 12)) ,'new\_resp'] = 'abnormal'

data.loc[((data['Resp'] < 12) | (data['Resp'] > 16)) & (data['Age'].between(12, 18)) ,'new\_resp'] = 'abnormal'

data.loc[((data['Resp'] < 12) | (data['Resp'] > 20)) & (data['Age'] > 18) ,'new\_resp'] = 'abnormal'

data['new\_resp'].fillna('missing', inplace = True)

return data

sepsis\_data = fe\_new\_resp(sepsis\_data)

7. Mean Arterial Pressure

MAP is the measurement that explains the average blood pressure in a person's blood vessels during a single cardiac cycle. Mean arterial pressure is significant because it measures the pressure necessary for adequate perfusion of the organs of the body.

The normal MAP range is between 70 and 100 mmHg.

High MAP can cause stress on the heart because it has to work harder than normal to push against the elevated pressure in the vessels. When the MAP gets below 60, vital organs in the body do not get the nourishment they need for survival MAP is directly affected by factors such as:

• Amount of blood pumped out of the heart per minute (cardiac output)

• Heart rate (beats per minute)

• Blood pressure

• Resistance to blood flow in the vessels

A change in any of these factors will alter the mean arterial pressure and cause negative effects on the body

def fe\_new\_map(data):

data.loc[(data['MAP'] >= 70) & (data['MAP'] < 100),'new\_map'] = 'normal'

data.loc[(data['MAP'] < 70) | (data['MAP'] >= 100),'new\_map'] = 'abnormal'

data['new\_map'].fillna('Missing', inplace=True)

return data

sepsis\_data = fe\_new\_map(sepsis\_data)

## 8. Fraction of inspired oxygen

The percentage of individual gases in air (oxygen, nitrogen, etc.) doesn't change with altitude, but the atmospheric (or barometric) pressure does. FIO2, the fraction of inspired oxygen in the air, is thus 21% (or .21) throughout the breathable atmosphere.

def fe\_new\_fio2(data):

data.loc[(data['FiO2'] < 0.8 ) ,'new\_fio2'] = 'normal'

data.loc[(data['FiO2'] >= 0.8 ),'new\_fio2'] = 'abnormal'

data['new\_fio2'].fillna('Missing', inplace=True)

return data

sepsis\_data = fe\_new\_fio2(sepsis\_data)

## Feature Selection - Selecting relevant features for prediction

## columns\_new = ['Gender', 'new\_age', 'new\_o2sat', 'new\_temp', 'new\_bp', 'new\_resp', 'new\_map', 'new\_fio2', 'new\_hr', 'HospAdmTime', 'ICULOS']

## target\_col = ['SepsisLabel']

## test\_cols = columns\_new + target\_col

all\_data\_train = sepsis\_data[test\_cols]

all\_data\_train.head()

***Dealing with imbalance dataset:***

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

training\_data = pd.read\_csv("NewFeatures.csv",sep=',')

training\_data.head()

for col in training\_data.columns.values:

# Encoding only categorical variables

if (training\_data[col].dtypes=='object'):

# Using whole data to form an exhaustive list of levels

data=training\_data[col]

le.fit(data.values)

training\_data[col]=le.transform(training\_data[col])

training\_data.head()

training\_data.shape

df= pd.DataFrame(training\_data)

df.to\_csv('training\_data.csv')

***Implementing Decision Tree:***

# Importing the required packages

import numpy as np

import pandas as pd

from sklearn.metrics import confusion\_matrix

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

from sklearn.metrics import classification\_report

df=pd.read\_csv("training\_data.csv")

# Function importing Dataset

def importdata():

balance\_data = pd.read\_csv("training\_data.csv")

# Printing the dataset shape

print ("Dataset Length: ", len(balance\_data))

print ("Dataset Shape: ", balance\_data.shape)

# Printing the dataset obseravtions

print ("Dataset: ",balance\_data.head())

return balance\_data

# Function to split the dataset

def splitdataset(balance\_data):

# Separating the target variable

cols = ['Gender', 'new\_age', 'new\_o2sat', 'new\_temp', 'new\_bp', 'new\_resp', 'new\_map', 'new\_fio2', 'new\_hr', 'HospAdmTime', 'ICULOS']

X = balance\_data[cols]

Y = balance\_data['SepsisLabel']

# Splitting the dataset into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, Y, test\_size = 0.3, random\_state = 100)

return X, Y, X\_train, X\_test, y\_train, y\_test

# Function to perform training with giniIndex.

def train\_using\_gini(X\_train, X\_test, y\_train):

# Creating the classifier object

clf\_gini = DecisionTreeClassifier(criterion = "gini",

random\_state = 100,max\_depth=10, min\_samples\_leaf=3,min\_samples\_split=20)

# Performing training

clf\_gini.fit(X\_train, y\_train)

return clf\_gin

# Function to perform training with entropy.

def tarin\_using\_entropy(X\_train, X\_test, y\_train):

# Decision tree with entropy

clf\_entropy = DecisionTreeClassifier(

criterion = "entropy", random\_state = 100, max\_depth = 10, min\_samples\_leaf = 3, min\_samples\_split=20)

# Performing training

clf\_entropy.fit(X\_train, y\_train)

return clf\_entropy

# Function to make predictions

def prediction(X\_test, clf\_object):

# Predicton on test with giniIndex

y\_pred = clf\_object.predict(X\_test)

print("Predicted values:")

print(\*y\_pred)

# Function to calculate accuracy

def cal\_accuracy(y\_test, y\_pred):

print("Confusion Matrix: ", confusion\_matrix(y\_test, y\_pred))

print ("Accuracy : ", accuracy\_score(y\_test,y\_pred)\*100)

print("Report : ", classification\_report(y\_test, y\_pred))

def main():

# Building Phase

data = importdata()

X, Y, X\_train, X\_test, y\_train, y\_test = splitdataset(data)

### clf\_gini = train\_using\_gini(X\_train, X\_test, y\_train)

### clf\_entropy = tarin\_using\_entropy(X\_train, X\_test, y\_train)

### # Operational Phase

### print("Results Using Gini Index:")

### # Prediction using gini

### y\_pred\_gini = prediction(X\_test, clf\_gini)

### cal\_accuracy(y\_test, y\_pred\_gini)

### print("Results Using Entropy:")

### # Prediction using entropy

### y\_pred\_entropy = prediction(X\_test, clf\_entropy)

### cal\_accuracy(y\_test, y\_pred\_entropy)

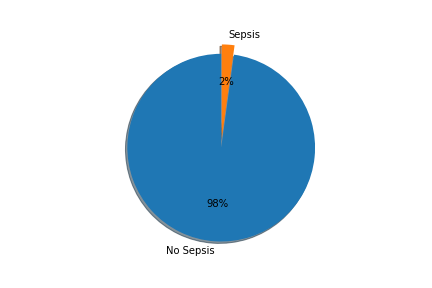
### if \_\_name\_\_=="\_\_main\_\_":

### main()

**CHAPTER 6**

**EXPERIMENTAL RESULTS**

**Data Preprocessing:**

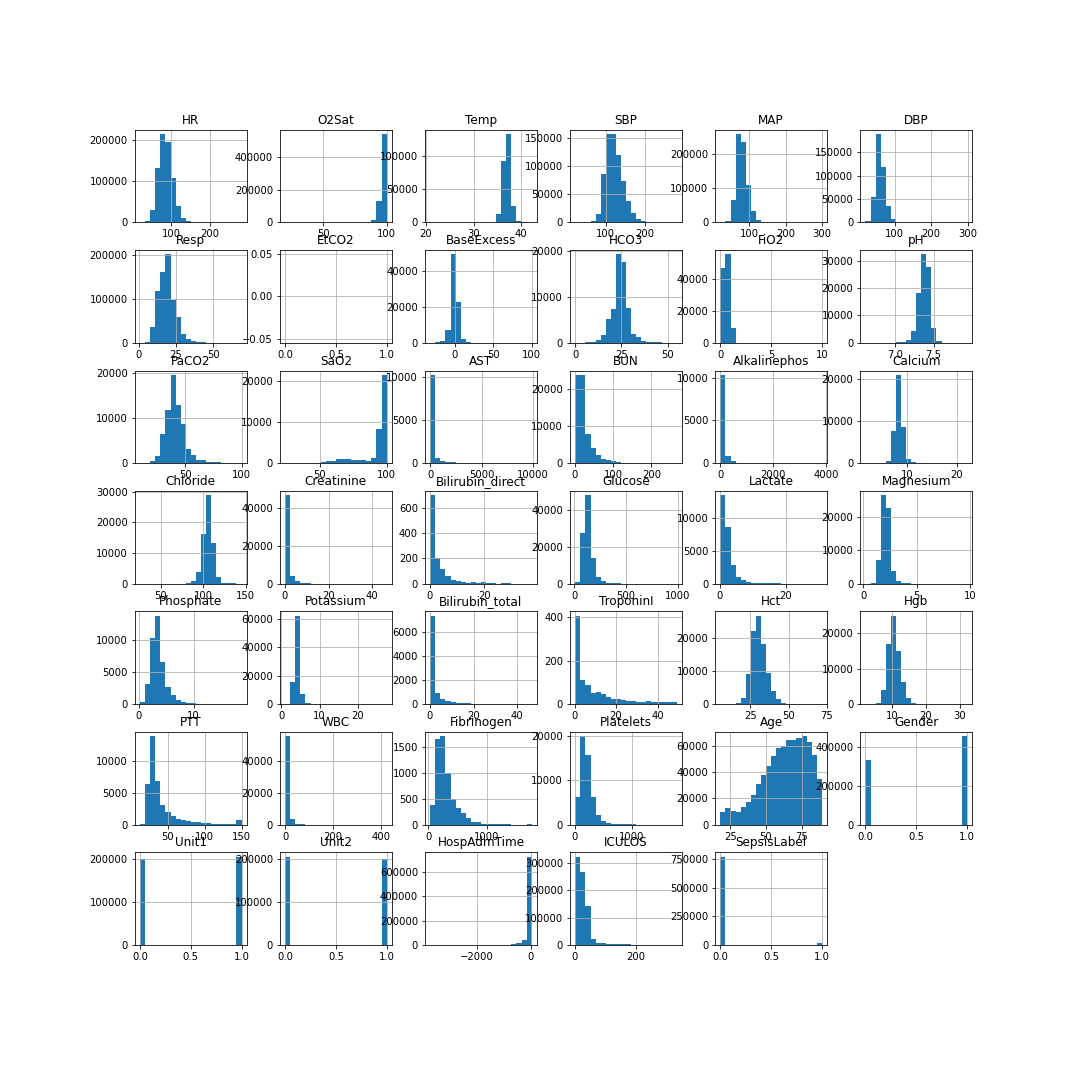
****

**Fig-5: Percentage of people affected by sepsis**

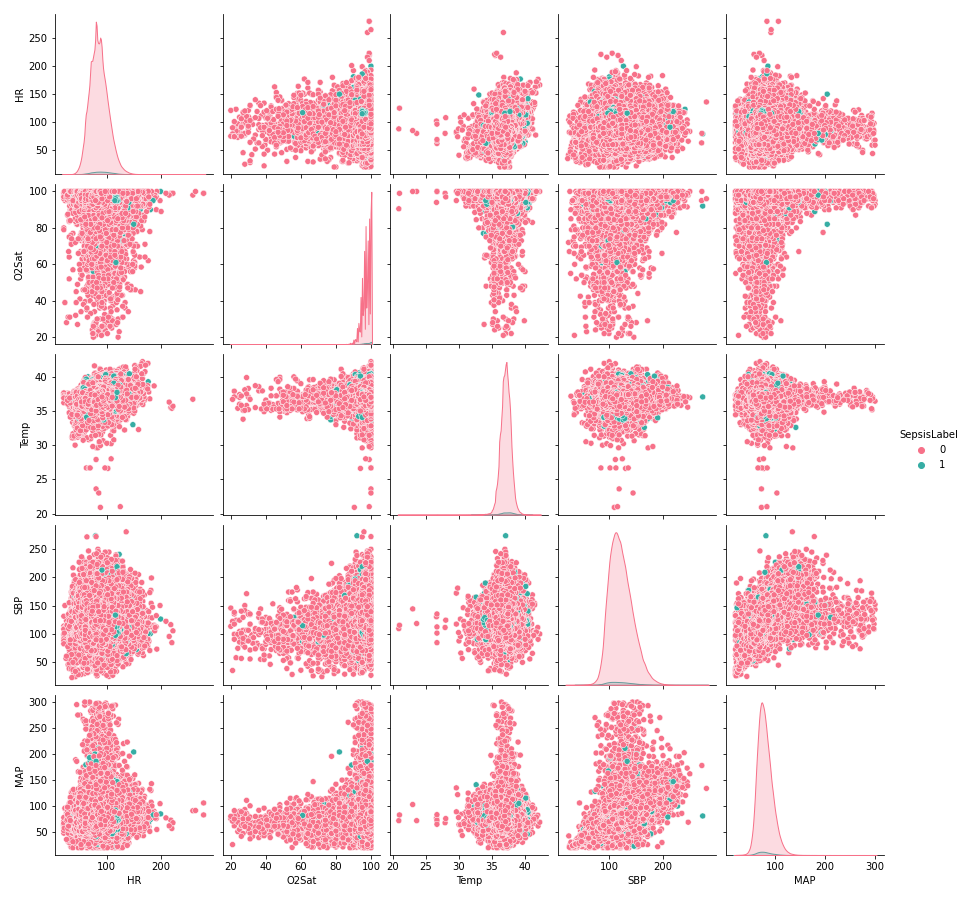
****

**Fig-6: Bar Chat of People Effected with Sepsis**

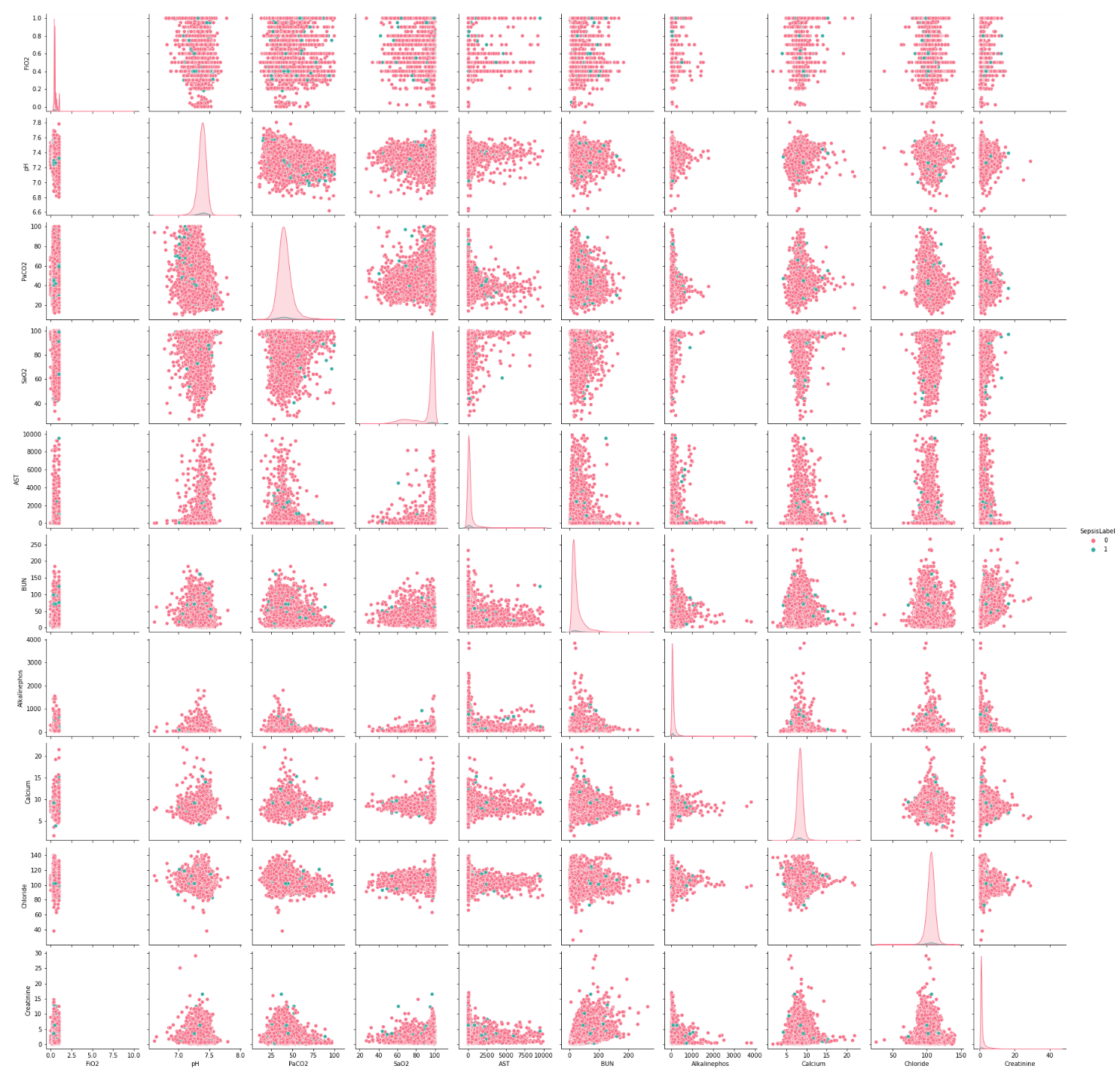
**Exploratory Data Analysis:**

****

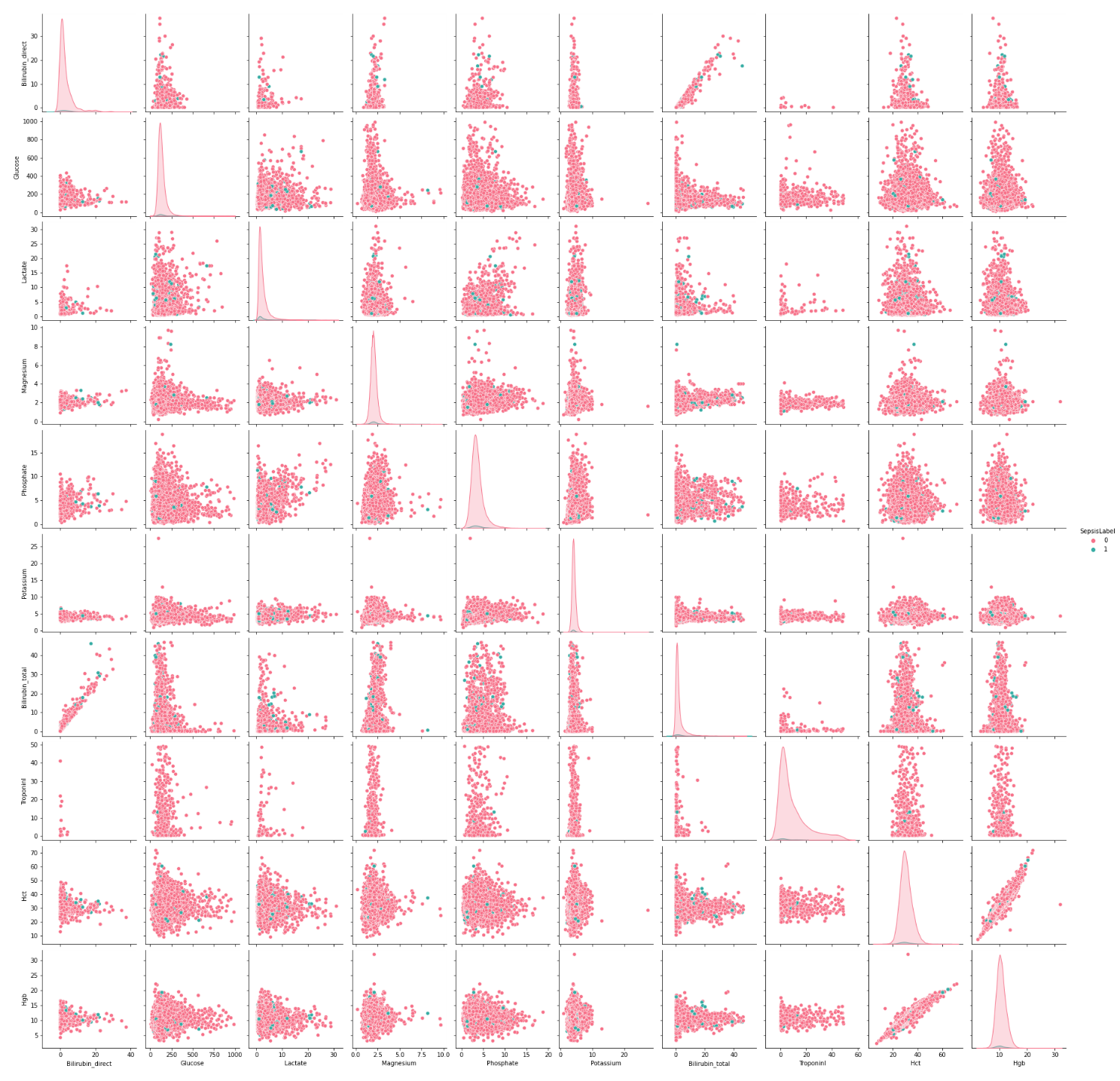
**Fig-7: Histogram of features**

****

**Fig-8: Pair Plots (a)**

****

**Fig-9: Pair Plots (b)**

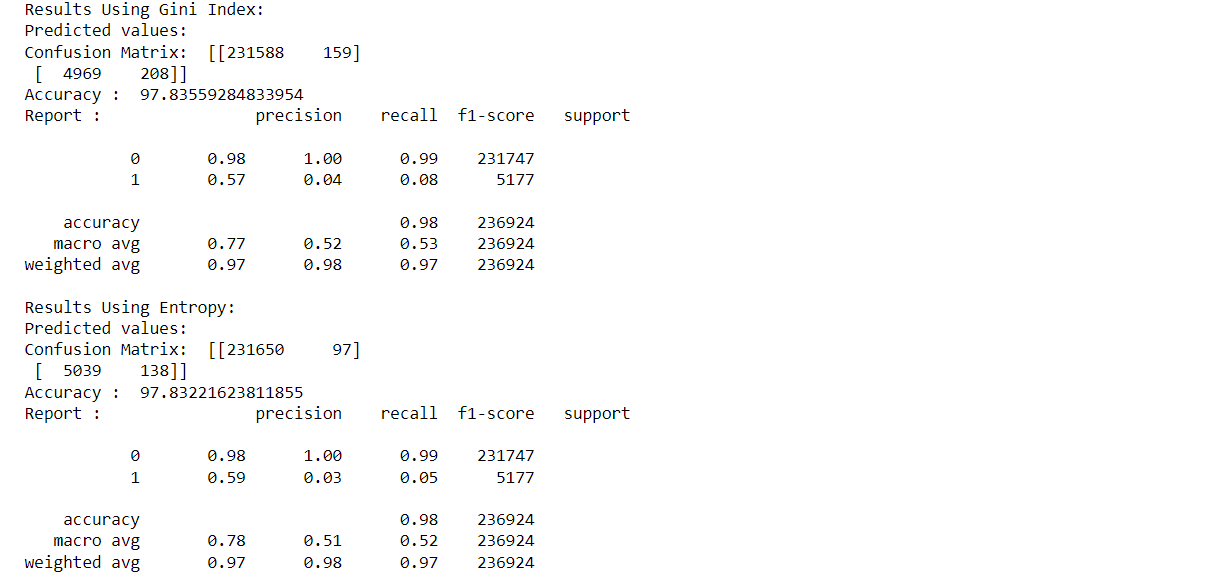
****

**Fig-10: Pair Plots (c)**

****

**Fig-11: Pair Plots (d)**

**Implementing Decision Tree:**

****

**Fig-12: Output of Decision Tree.**

**CHAPTER 7**

**CONCLUSION**

Decision trees have a stronger potential for categorization than the other algorithm types. Label encoding is used instead of typical imputation approaches to achieve better results.  
This strategy enabled us to train the model efficiently while also taking into account other factors such as minimum leaves, split, random value, and maximum depth. This model ensured the confusion matrix: [[231650 97]] [5039 138]]. The AUC is 97.83221623811855.

***Future Scope:***

The future focus would be on detecting at-risk patients and surfacing crucial areas that should be considered for sepsis. It is possible to forecast the disease-prone percentage using modern approaches. This would make it much easier to recognize the stage. Other factors can be considered while monitoring patients in real time to anticipate disease.

**REFERENCES**

[1] <https://www.upgrad.com/blog/gini-index-for-decision-trees/>

[2] <https://physionet.org/content/challenge-2019/1.0.0/>

[3] <https://github.com/kskaran94/Sepsis_Identification>

[4] <https://github.com/sedab/EarlySepsisPrediction>

[5] <https://github.com/onurhalityenice/Diabetes-Feature-Enginnering>