

PROFESSIONAL TRAINING REPORT
AT
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in

Information Technology

By

GOUTHAM KUMAR.J
(REG.NO.37120037)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMILNADU.
JANUARY – 2021



**SATHYABAMA INSTITUTE OF
SCIENCE AND TECHNOLOGY**
(Established under Section 3 of UGC Act, 1956)



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bona-fide work of **GOUTHAM KUMAR J (Reg. No. 37120037)** who underwent the professional training in “**PREDICTING HOSPITAL ADMISSIONS FROM THE EMERGENCY DEPARTMENT USING DATAMINING**” under our supervision from to June 2019 to August.

Internal Guide

Dr. V. NIRMALRANI

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, GOTHAM KUMAR.J (Reg. No. 37120037) hereby declare that the Professional Training Report on **“PREDICTING HOSPITAL ADMISSIONS FROM THE EMERGENCY DEPARTMENT USING DATAMINING”** done by me under the guidance of Dr. V. Nirmalrani, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kindencouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing , **Dr.SUBHASHINI M.E., Ph.D. ,** Head of the Department of Information and Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. V.Nirmalrani ,for his valuable, guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

The Emergency Departments, which are life saving for many people have negative impacts too, if many people are crowded in the ED, there are chances that some people lose their life even before getting admitted in the ED. Therefore, we need to explore the use of innovative methods to improve patient flow and prevent overcrowding. The Machine Learning, can help us in learning the data, analyzing the data and predicting the future. The ED's overcrowding can be reduced by predicting the future of patients admission using Machine Learning and making the resources available before hands. The Base Paper has used the DNN algorithm to build the predictive model. We identify several factors related to hospital admissions, including hospital site, age, arrival mode, triage category, care group, previous admission in the past month, and previous admission in the past year. Practical implementation of the models developed in decision support tools that would provide a snapshot of predicted admissions from the ED at a given time, allowing for advance resource planning and the avoidance bottlenecks in patient flow, as well as comparison of predicted and actual admission rates. When interpretability is a key consideration, EDs should consider adopting logistic regression models, although GBM's will be useful where accuracy is paramount. The (DNN) Deep Neural Network implemented is used to predict the data and to improve the accuracy.

TABLE OF CONTENTS

CAPTHERS NO	TITLE	PAGE NO
1	INTRODUCTION	1
1.1	Outline of Project	1
1.2	Domain Introduction	2
1.3	Benifits of Deep Neural Network	2
2	AIM AND SCOPE OF THE PROJECT	3
2.1	Aim	3
2.2	Scope of the project	4
2.3	Objecive	4
2.4	System requirement	5
2.4.1	Hardware requirements	5
2.4.2	Software requirements	5
2.5	Software features	6
2.5.1	Python	6
2.5.2	Spyder	6
2.5.3	Keras	6
2.5.4	Anaconda	7
2.5.5	IDE Setup	7

2.5.6	Algorithm	7
3	METHODOLOGY	8
3.1	Existing System	9
3.2	Proposed System	9
3.2.1	Advantages of Proposed System	10
3.3	System Architecture	10
3.4	Modules	11
3.5	Module description	12
3.5.1	Data Extraction	13
3.5. 2	Pre-processing	15
3.5.3	Data visualisation	15
3.5.4	Data splitting	16
3.5.5	Model Tuning	17
3.5.6	Predicting Admissions	18
3.5.7	Evaluation of model performance	19
3.6	Data Flow Diagram	20
3.6.1	DFD Level For Data Pre-processing	21
3.6.2	DFD Level For Feature Extraction	22
3.6.3	DFD Level For Splitting Data	23
3.6.4	DFD Level For Classifying Data	24
3.6.5	DFD Level For Prediction	25

3.7	Use Case Diagram For Load Forecasting	26
3.8	Sequence Diagram For Load Forecasting	27
3.9	System Testing	27
3.9.1	White box testing	29
3.10	Black box testing	30
3.10.1	Functional testing	30
4	RESULTS AND DISCUSSIONS	31
4.1	Experimental Results	32
5	CONCLUSION AND FUTURE WORK	34
5.1	Summary	34
5.2	Conclusion	34
5.3	Future work	35
	REFERENCES	36
	APPENDIX	37
	A. Source code	38
	B. Screenshots	39
	C. Publication	45

LIST OF FIGURES

FIGURE No.	FIGURE NAME
3.1	DNN Layers
4.1	Architectural Diagram
4.2	Data extraction
4.3	Pre-processing
4.4	One hot encoding
4.5	Transforming data
4.6	Data visualization
4.7	Splitting data
4.8	Modelling
4.9	Prediction
4.10	Evaluation
4.11	Data flow diagram
4.12	Functionality
4.13	DFD Level for Pre-processing Data
4.14	DFD Level for Feature Extraction
4.15	DFD Level for Splitting Data

4.16	DFD Level for Classifying Data
4.17	DFD Level for Prediction
4.18	Use case diagram
4.19	Sequence diagram
5.1	Dataset Collected
5.2	Dataset for Cleaning
5.3	Accuracy Prediction for DNN
5.4	Epoch

LIST OF ABBREVIATIONS

ED	Emergency Department
ML	Machine Learning
DM	Data Mining
GBM	Gradient Boosted Machines
DNN	Deep Neural Network
MAPE	Mean Absolute Percentage Error
FMM	Fuzzy Min-Max
CART	Classification And Regression Trees
MLP	Multi-Layer Perceptron
IDE	Integrated Development Environment
ESI	Emergency Severity Index
DFD	Data Flow Diagram

CHAPTER 1

INTRODUCTION

1.1 OUTLINE OF THE PROJECT

The Emergency Departments, which are life saving for many people. But if many people are crowded in the ED, it may sometimes have negative impacts too such as increased wait time, ambulance diversion, reduced staff morale, adverse patient outcomes such as increased mortality, and cancellation of elective procedures .There are chances that some people lose their life even before getting admitted in the ED. Therefore we need to explore the use of innovative methods to improve patient flow and prevent overcrowding.

The Machine Learning, can help us in learning the data, analyzing the data and predicting the future .One mechanism that could help to reduce ED crowding and improve patient flow is the use of data mining to identify patients at high risk of an inpatient admission, therefore allowing measures to be taken to avoid bottlenecks in the system. For example, a model that can accurately predict hospital admissions could be used for inpatient bed management, staff planning and to facilitate specialized work streams within the ED. in future using Machine Learning and making the resources available beforehand.

This study draws on this data to achieve two objectives. The first is to create a model that accurately predicts admission to hospital from the ED department, and the second is to evaluate the performance of common machine learning algorithms in predicting hospital admissions. We also suggest use cases for the implementation of the model as a decision support and performance management tool.

1.2 DOMAIN INTRODUCTION

The Deep Neural Network (DNN) is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modelling to process data in complex ways.

A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy.” One of the key uses of these sophisticated neural networks is dealing with unlabelled or unstructured data. The phrase “deep learning” is also used to describe these deep neural networks, as deep learning represents a specific form of machine learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

1.3 BENEFITS OF DEEP NEURAL NETWORK

Neural networks use randomness by design to ensure they effectively learn the function being approximated for the problem. Randomness is used because this class of machine learning algorithm performs better with it than without.

The most common form of randomness used in neural networks is the random initialization of the network weights. Although randomness can be used in other areas, here is just a short list:

- Randomness in Initialization, such as weights.
- Randomness in Regularization, such as dropout.
- Randomness in Layers, such as word embedding.
- Randomness in Optimization, such as stochastic optimization.

CHAPTER 2

AIM AND SCOPE OF THE PROJECT

2.1 AIM

The aim of the project is to create a model that can accurately predict hospital admissions could be used for inpatient bed management, staff planning and to facilitate specialized work streams within the ED. The main aim of the project is to reduce the ED's overcrowding predicting the admissions patients in future using Machine Learning and making the resources available beforehand.

2.2 SCOPE OF THE PROJECT

To develop an ED focused tools, a subset of which use machine learning, to help with decision making and resource planning. ED prediction models using machine learning includes clinical conditions, probability of patient admission from the ED.

2.3 OBJECTIVE

Our Aim ED-ML should meet three basic requirements, i.e., apt resource allocations, avoid bottleneck, and to improve the ED admissions efficiently. Efficient ED admission through the three algorithms by Byron Graham can be further improved by improvising the system through Deep Neural Network and achieving the efficient ED admissions.

2.4 SYSTEM REQUIREMENTS

The System requirements are the minimum hardware and software requirements needed for the implementation and execution of the model.

2.4.1 HARDWARE REQUIREMENT

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by the software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented

SYSTEM : PENTIUM DUAL CORE

RAM : 8 GB

HARD DISK : 120 GB

2.4.2 SOFTWARE REQUIREMENTS

The software requirements document is the software specification of the system. It should include both a definition and a specification of a requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's progress throughout the development activity.

OPERATING SYSTEM : WINDOWS 10

PROGRAMMING LANGUAGE : PYTHON

TOOLS : ANACONDA, SPYDER, PYTHON

DATABASE : KERAS LIBRARY

2.5 SOFTWARE FEATURES

The software features gives a detailed about all the software tools and database used in the model.

2.5.1 Python

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative,

functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation. Python is a multiparadigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution, which binds method and variable names during program execution. Python's design offers some support for functional programming in the Lisp tradition. It has `filter()`, `map()`, and `reduce()` functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules that implement functional tools borrowed from Haskell and Standard ML.

2.5.2 Spyder

Spyder is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. It is released under the MIT license. Initially created and developed by Pierre Raybaut, Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available crossplatform through Anaconda, on Windows with WinPython and Python (x,y), on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian,

Fedora, Gentoo Linux, openSUSE and Ubuntu. Spyder uses for its GUI, and is designed to use either of the PyQt or PySide Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

2.5.3 Keras

Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. It was developed by François Chollet, a Google engineer. Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend. So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function. Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.

2.5.4 Anaconda

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing that aims to simplify package management and deployment. Package versions are managed by the package management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda is a Python-based data processing and scientific computing platform. It has built in many very useful third-party libraries. Installing Anaconda is equivalent to automatically installing Python and some commonly used libraries such as Numpy,

Pandas, Scrip, and Matplotlib, so it makes the installation so much easier than regular Python installation. It is painful and you need to consider compatibility, thus it is highly recommended to directly install Anaconda.

2.5.5 IDE Setup

The Machine Learning Concepts can be well implemented through PYTHON . We have a numerous Python Tools but for the DNN implementation Anaconda can meet our needs. Spyder in Anaconda is chosen as the IDE setup . The latest Anaconda Spyder has Python 3.6 version. Python 3.6 is unstable to hold the KERAS backend . Apart from the inability to build over Python 3.6 , KERAS requires two additional library packages THEANO and TENSORFLOW. Tensorflow can be implemented over Python 3.5 only.

A cloning environment is built using the Anaconda Prompt . The Python 3.5 is cloned using the command .

conda create -n py35 python=3.5 anaconda

After activating the cloned environment the Spyder is installed using the command . So the Python3.5 has its own Spyder version , which does not interfere with the base version of Anaconda .

conda install spyder

The required libraries can be set up using the below commands

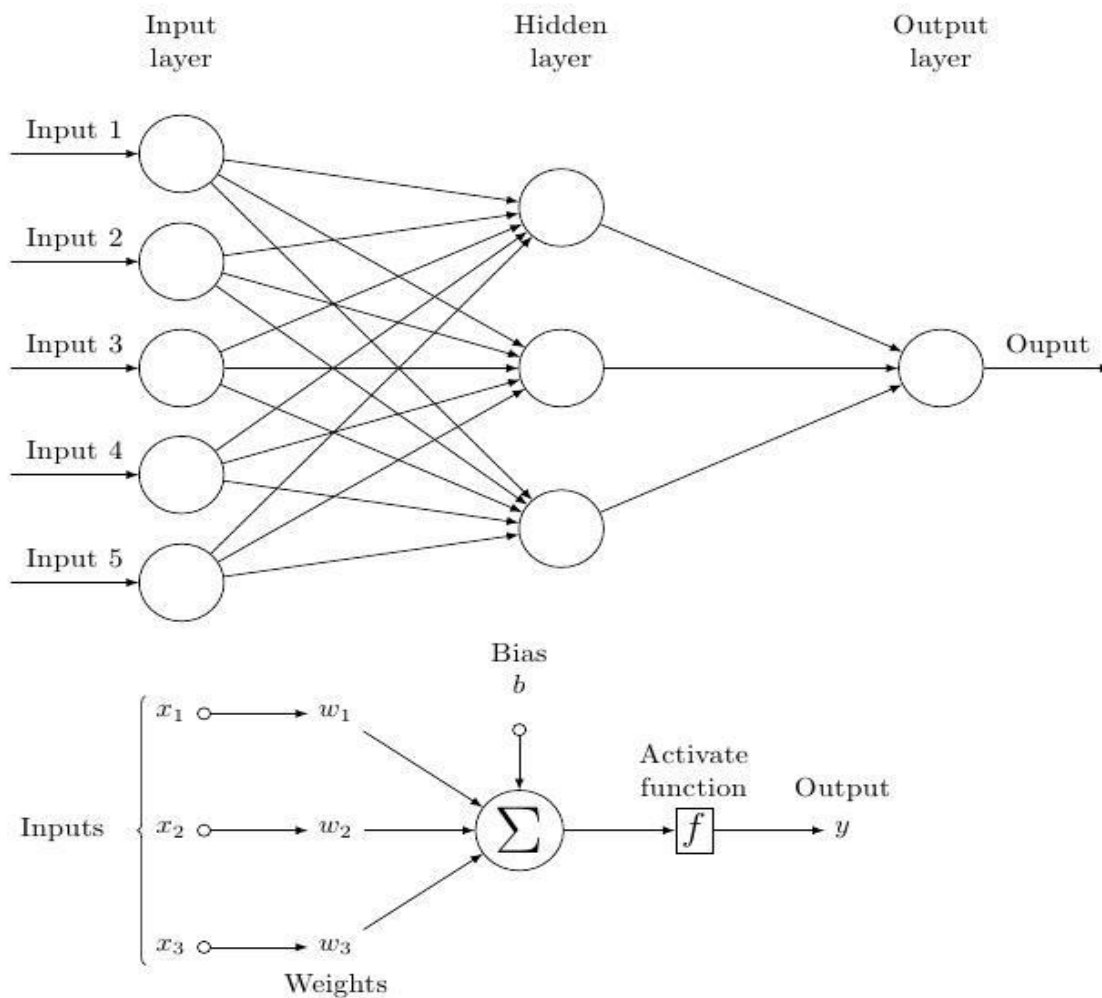
conda install theano

conda install tensorflow

conda install keras

2.6 ALGORITHM

The machine learning algorithm that has to be applied to the training data to build



the model is Deep Neural Network.

Fig. 2.1 DNN Layers

A DNN is a neural network with a certain level of complexity, it is a neural network with more than two layers.

Deep neural networks use complicated mathematical modelling to process data in complex ways. A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through

various layers of simulated neural connections. One of the key uses of these sophisticated neural networks is dealing with unlabelled or unstructured data.

The phrase “deep learning” is also used to describe these deep neural networks, as deep learning represents a specific form of machine learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

CHAPTER 3

METHODOLOGY

3.1 EXISTING SYSTEM

Emergency department (ED) crowding can have serious negative consequences for patients and staff, such as increased wait time, ambulance diversion, reduced staff morale, adverse patient outcomes such as increased mortality, and cancellation of elective procedures. There are a range of possible causes of ED crowding depending on the context, with some of the main reasons including increased ED attendances, inappropriate attendances, and a lack of alternative treatment options, a lack of inpatient beds, ED staffing shortages, and closure of other local ED departments. Using a range of clinical and demographic data relating to elderly patients, *LaMantia et al* used logistic regression to predict admissions to hospital, and ED re-attendance.

Baumann and Strout also found an association between the ESI and admission of patients aged over 65. Boyle et al. used historical data to develop forecast models of ED presentations and admissions. Model performance was evaluated using the mean absolute percentage error (MAPE), with the best attendance model achieving a MAPE of around 7%, and the best admission model achieving a MAPE of around 2% for monthly admissions. The use of historical data by itself to predict future events has the advantage of allowing forecasts further into the future, but has the disadvantage of not incorporating data captured at arrival and through triage, which may improve the accuracy of short term forecasting of admissions.

Sun et al developed a logistic regression model using two years of routinely collected administrative data to predict the probability of admission at the point of triage. Risk of admission was related to age, ethnicity, arrival mode, patient acuity score, existing chronic conditions, and prior ED attendances or admission in the past three months. Although their data showed the admission of more females than males, gender was not significant in the final model. Similarly, Cameron et al developed a logistic regression model to predict the probability of admissions at triage, using two years of routine administration data collected from hospitals in Glasgow. The most

important predictors in their model included triage category, age, National Early Warning Score, arrival by ambulance, referral source, and admission within the last year, with an area under the curve of the receiver operating characteristic (AUCROC) of 0.877.

Although these models highlight the usefulness of logistic regression in predicting ED admissions, Xie achieved better performance using a Coxian Phase model over logistic regression model, with the former AUC-ROC of 0.89, and the latter 0.83. Wang et al used a range of machine learning algorithms to predict admissions from the ED, comparing the ability of fuzzy min-max neural networks (FMM) to other standard data mining algorithms including classification and regression trees (CART), Multi-Layer Perceptron (MLP), random forest, and AdaBoost.

3.2 PROPOSED SYSTEM

The Proposed system increases the prediction of patient admission in the ED by reducing the bottleneck. The system has an additional classification called Deep Neural Network algorithm. This increases the accuracy of the new system.

The Multilayer Perceptron is a feed forward neural network model capable of classification and regression. A multilayer perceptron is composed of fully Connected layers of artificial neurons forming a connected graph (in the simplest case one hidden, one final layer), where neurons are modelled very loosely on biological neurons; and as a feed forward net, the output of each layer becomes the input of the next one. Multilayer Perceptrons can be trained through the Backpropagation Algorithm. The Backpropagation Algorithm is an application of gradient descent that takes into account the contribution to the error by the neurons of the hidden layer. Through differential calculus the gradients of all neurons can be calculated and weights updated accordingly.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Adding the proposed system to the old system increases the predictions by reducing the overfitting.

- Another benefit of implementing the model developed here is that it can help to improve planning and resource allocation in hospitals.

3.3 SYSTEM ARCHITECTURE

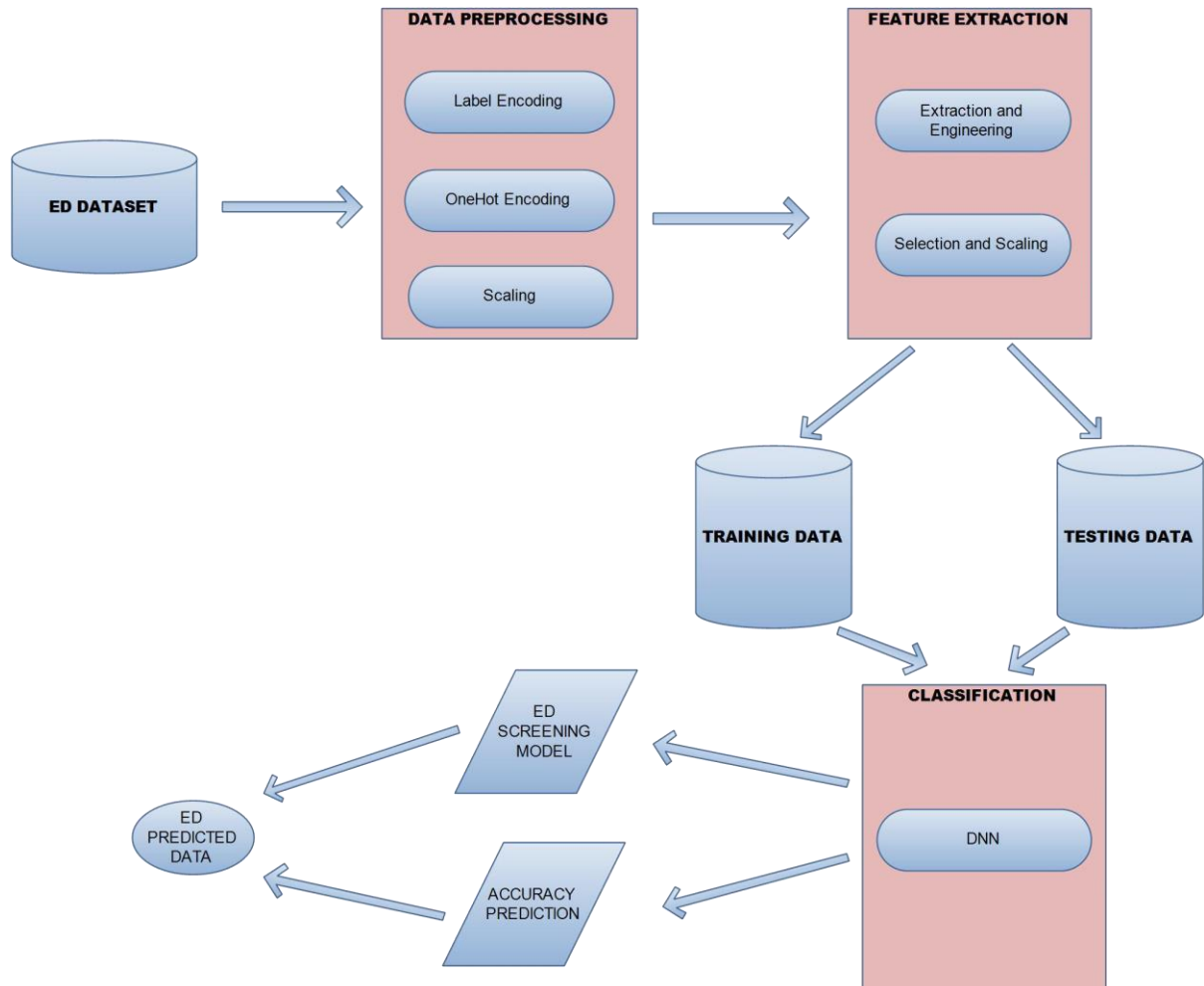


Fig 4.1 Architectural design

3.4 MODULES

The modules for this study involved seven data mining tasks,

1. Data extraction
2. Data cleansing and feature engineering
3. Data visualisation

4. Data splitting
5. Model tuning
6. Predicting admissions
7. Evaluation of model performance

These steps help to ensure the models are optimal and prevent against overfitting. A range of variables were considered in the model building, with the final variables decided upon based on previous studies, significance in the models, and the impact of inclusion on the performance of the model. The final models consisted of variables describing whether the patient was admitted to hospital; hospital site; date and time of attendance; age; gender; arrival model; care group; Manchester triage category; and whether the patient had a previous admission to the hospital within the last week, month, or year. The care group is a series of categories indicating the pathway a patient should take. The Manchester triage category is a scale rating the severity of the condition, and used for prioritisation. Prior admissions were measured objectively by querying the hospital database.

Feature engineering was also carried out on the date of attendance to disaggregate it into components relating to year, day of the week, and month of the year. The dependent variable in all models was admission to the hospital from the ED. Most of the variables included in the model are mandatory on the ED system, and recorded using of drop down menus. This led to a relatively clean dataset for analysis, with list wise deletion of cases with missing data. Patients attending direct assessment units and observation units are excluded from the analysis, as these patients follow a different pathway to those attending the main ED.

3.5 Module Description

To build the complete prediction model, it is essential to carry out step-by-step execution of all required modules.

3.5.1 Data extraction

We have collected the Emergency Department dataset of Ireland for Neural Network Processing. We are using this data info to predict ED for the current year and make appropriate preparations.

Importing the Pandas gives massive functionality to work on data. Along with Pandas importing all the required libraries are even imported.

```

1 import numpy as np
2 import pandas as pd
3 import matplotlib
4
5
6 import theano
7 import tensorflow
8 import keras
9
10
11 dataset = pd.read_csv('D:\\ljjfpiware\\Emergency Department\\ED_Dataset.csv')
12 X = dataset.iloc[:, 0:34].values
13 y = dataset.iloc[:, 34].values

```

Fig.4.2 Data extraction

3.5.2 Pre-processing

The data has to be cleaned before loading into the Neural Classifier.

Label Encoding

Before running a model, we need to make this data ready for the model. To convert any kind of categorical text data into model-understandable numerical data, we use the Label Encoder class, all we have to do is to label encode the first column. We import the Label Encoder class from the sklearn library, fit and transform the first column of the data, and then replace the existing text data with the new encoded data.

```

14
15 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
16 labelencoder_X_1 = LabelEncoder()
17 X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
18
19 labelencoder_X_2 = LabelEncoder()
20 X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
21
22
23 X
24 labelencoder_X_3 = LabelEncoder()
25 X[:, 3] = labelencoder_X_3.fit_transform(X[:, 3])
26 |
27
28 X
29 labelencoder_X_4 = LabelEncoder()
30 X[:, 4] = labelencoder_X_4.fit_transform(X[:, 4])
31
32
33 labelencoder_X_5 = LabelEncoder()
34 X[:, 5] = labelencoder_X_5.fit_transform(X[:, 5])
35
36 labelencoder_X_6 = LabelEncoder()
37 X[:, 6] = labelencoder_X_6.fit_transform(X[:, 6])
38

```

Fig.4.3 Pre-processing

One hot Encoding

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. This is why we use one hot encoder to perform “binarization” of the category and include it as a feature to train the model.

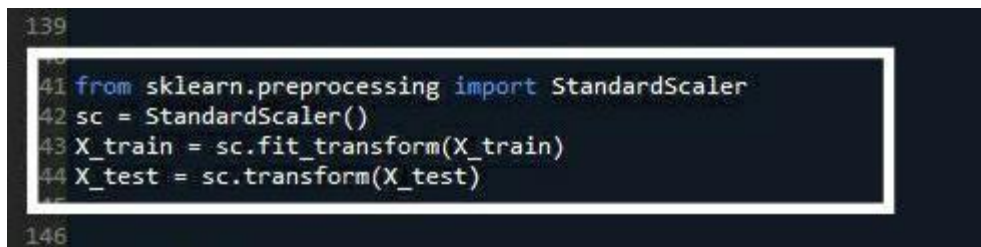
A screenshot of a code editor window titled 'temp.py'. The code shows the process of one-hot encoding categorical features. It starts with creating three LabelEncoder objects for features at indices 28, 29, and 30 of the dataset X. Each feature is then transformed using its respective LabelEncoder. Finally, a OneHotEncoder is created with categorical_features = [1], and the transformed features are combined back into the main dataset X using X[:, 1:].

```
118
119
120 labelencoder_X_28 = LabelEncoder()
121 X[:, 28] = labelencoder_X_28.fit_transform(X[:, 28])
122
123
124 labelencoder_X_29 = LabelEncoder()
125 X[:, 29] = labelencoder_X_29.fit_transform(X[:, 29])
126
127 X
128 labelencoder_X_30 = LabelEncoder()
129 X[:, 30] = labelencoder_X_30.fit_transform(X[:, 30])
130
131
132 onehotencoder = OneHotEncoder(categorical_features = [1])
133 X = onehotencoder.fit_transform(X).toarray()
134 X = X[:, 1:]
```

Fig.4.4 One hot encoding

Standard Scaler

Standard Scaler is available in Scikit Learn we are fitting and transforming Standard Scaler method on train data. We have to standardize our scaling so we will use the same fitted method to transform/scale test data.

A screenshot of a code editor window showing the use of StandardScaler. It imports StandardScaler from sklearn.preprocessing, creates an instance 'sc', and then uses 'sc.fit_transform(X_train)' to fit and transform the training data, and 'sc.transform(X_test)' to transform the test data using the fitted scaler.

```
139
140
141 from sklearn.preprocessing import StandardScaler
142 sc = StandardScaler()
143 X_train = sc.fit_transform(X_train)
144 X_test = sc.transform(X_test)
145
146
```

Fig.4.5 Transforming data

3.5.3 Data visualisation

The extracted features are usually plotted against the output to check its relation to the output.

```

173
174
175 import numpy as np
176 import pandas as pd
177 import seaborn as sns
178 sns.set_palette('husl')
179 import matplotlib.pyplot as plt
180
181
182 tmp = dataset.drop('id', axis=1)

```

```

184 g = sns.violinplot(y='disposition', x='esi', data=dataset, inner='quartile')
185 plt.show()
186
187 g = sns.violinplot(y='disposition', x='age', data=dataset, inner='quartile')
188 plt.show()
189
190 g = sns.violinplot(y='disposition', x='asthma', data=dataset, inner='quartile')
191 plt.show()
192
193 g = sns.violinplot(y='disposition', x='allergy', data=dataset, inner='quartile')
194 plt.show()

```

Fig.4.6 Data visualization

3.5.4 Data splitting

In statistics and machine learning we usually split our data into two subsets: **training data** and **testing data** (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. When we do that, one of two things might happen: we overfit our model or we underfit our model. We don't want any of these things to happen, because they affect the predictability of our model—we might be using a model that has lower accuracy and/or is ungeneralised (meaning you can't generalize your predictions on other data).

Overfitting

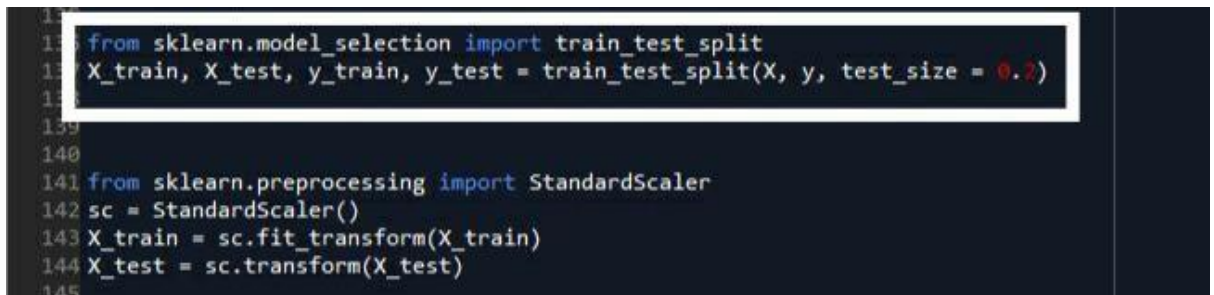
Overfitting means that model we trained has trained “too well” and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex (i.e. too many features/variables compared to the number of observations). This model will be very accurate on the training data but will probably be very not accurate on untrained or new data. It is because this model is not generalized (or not as generalized), meaning we can generalize the results and can't make any inferences on other data, which is, ultimately, what we are trying to do. Basically, when this happens, the model learns or describes the “noise” in the training data instead of the actual relationships between variables in the data. This noise, obviously, isn't part in of any new dataset, and cannot be applied to it.

Underfitting

In contrast to overfitting, when a model is underfitted, it means that the model does not fit the training data and therefore misses the trends in the data. It also means the model cannot be generalized to new data. As you probably guessed this is usually the result of a very simple model (not enough predictors/independent variables). It could also happen when, for example, we fit a linear model to data that is not linear. It almost goes without saying that this model will have poor predictive ability (on training data and can't be generalized to other data).

Train / Test Split

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.



```
138 from sklearn.model_selection import train_test_split
139 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
140
141 from sklearn.preprocessing import StandardScaler
142 sc = StandardScaler()
143 X_train = sc.fit_transform(X_train)
144 X_test = sc.transform(X_test)
145
```

Fig.4.7 Splitting data

3.5.5 Model Tuning

A deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modelling to process data in complex ways. A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through various layers of simulated neural connections.

Many experts define deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy.” One of the key uses of these sophisticated neural networks is dealing with unlabelled

or unstructured data. The phrase “deep learning” is also used to describe these deep neural networks, as deep learning represents a specific form of machine learning where technologies using aspects of artificial intelligence seek to classify and order information in ways that go beyond simple input/output protocols.

```
143
144
145 import keras
146 from keras.models import Sequential
147 from keras.layers import Dense
148
149
150
151 classifier = Sequential()
152 classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim = 31))
153 classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
154 classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
155 classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
156 classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 100)
```

Fig.4.8 Modelling

We can train our model on training data. We use fit method to the fit our model **Batch size** is used to specify the number of observation after which we want to update weight. **Epoch** is nothing but the total number of iterations. Choosing the value of batch size and epoch is trial and error there is no specific rule for that.

3.5.6 Predicting admissions

Predicting the test set result. The prediction result will give you probability of the People getting admitted in the hospital for the given period. We can also allocate the resources and doctors assigned to them. We will convert that probability into binary 0's and 1's.

```
160 y_pred = classifier.predict(X_test)
161 y_pred = (y_pred > 0.5)
162
163 from sklearn.metrics import confusion_matrix
164 from sklearn.metrics import accuracy_score
```

Fig.4.9 Prediction

3.5.7 Evaluation of model performance

This is the final step where we are evaluating our model performance. We build Confusion matrix to check the accuracy of model.

```

163 from sklearn.metrics import confusion_matrix
164 from sklearn.metrics import accuracy_score
165
166 cm = confusion_matrix(y_test, y_pred)
167 print(cm)
168 print('Accuracy' + str(accuracy_score(y_test, y_pred)))
169
170
171 from sklearn.metrics import confusion_matrix
172 cm = confusion_matrix(y_test, y_pred)
173
174

```

Fig.4.10 Evaluation

3.6 DATA FLOW DIAGRAM

DFDs are used to Specify Functions of the Information System and how data flow from function to function. It's a collection of function that manipulates data. On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. A DFD provides no information about the timing of processes, or about whether processes will operate in sequence or in parallel.

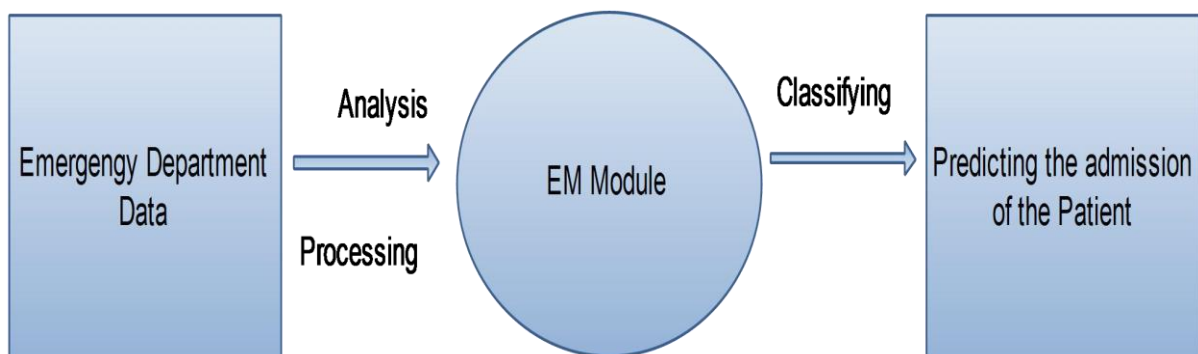
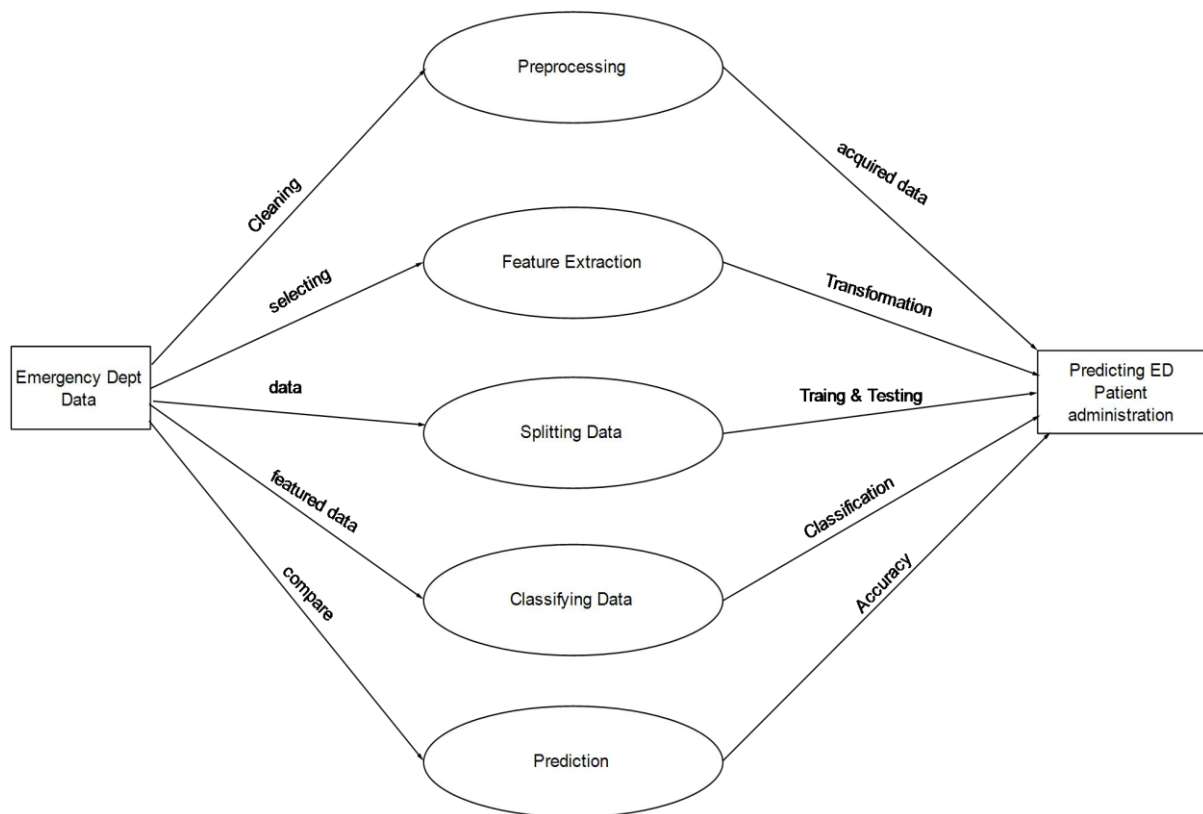


Fig.4.11 Data flow diagram

The figure explains the outer overall functionality of the proposed system application this indicates that have collected the Emergency Department dataset from various source and analysis and processing of data have



done by fetching the data from many hospitals.

Fig.3.12 Functionality

The figure above is an overall representation of each module and their functionality in this project. This Level one Data Flow Diagram shows how the various modules help in the providing the output which predicts the accurate Emergency patient admission information data with high frequency dataset.

3.6.1 DFD LEVEL FOR DATA PREPROCESSING

It explains the data is cleaned before loading to the Neural Network Classifier. Import the Label Encoder class from the sclera library, fit and transform the first column of the data, and then replace the existing text data with the new encoded data

.One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction.
This work fitting and transforming StandardScaler method on train data.

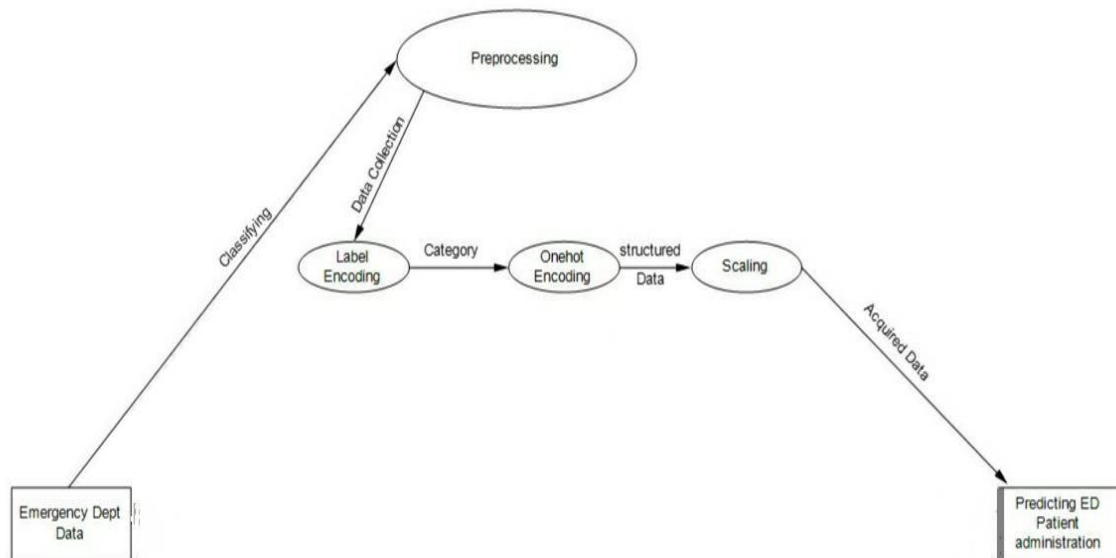


Fig. 4.13 DFD Level for Pre-processing Data

3.6.2 DFD LEVEL FOR FEATURE EXTRACTION

It explains the Feature extraction by Layering of data. For feature extraction, first select the data and process data for original feature. Then reduce the feature by transform refining data. This can be done by selecting the data by CSV file and process the data for original feature and it was transformed and refining data to reduced feature, finally it predicts the accuracy of weather.

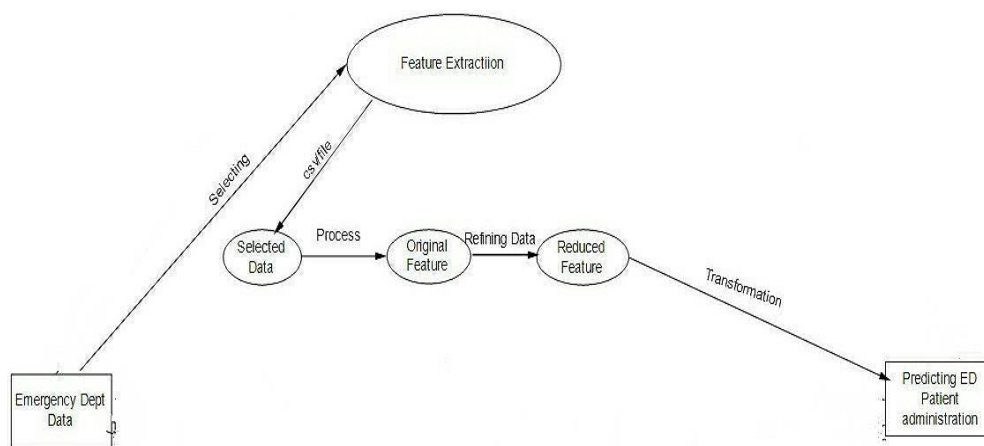


Fig.3.14 DFD Level for Feature Extraction

3.6.3 DFD LEVEL FOR SPLITTING DATA

It explains the Splitting data by two ways are Training dataset and Testing dataset. In statistics and machine learning we usually split our data into two subsets: **training data** and **testing data** (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. When this work do that, one of two thing might happen: we overfit our model or we underfit our model. It don't want any of these things to happen, because they affect the predictability of our model— Work might be using a model that has lower accuracy and/or is not generalized .

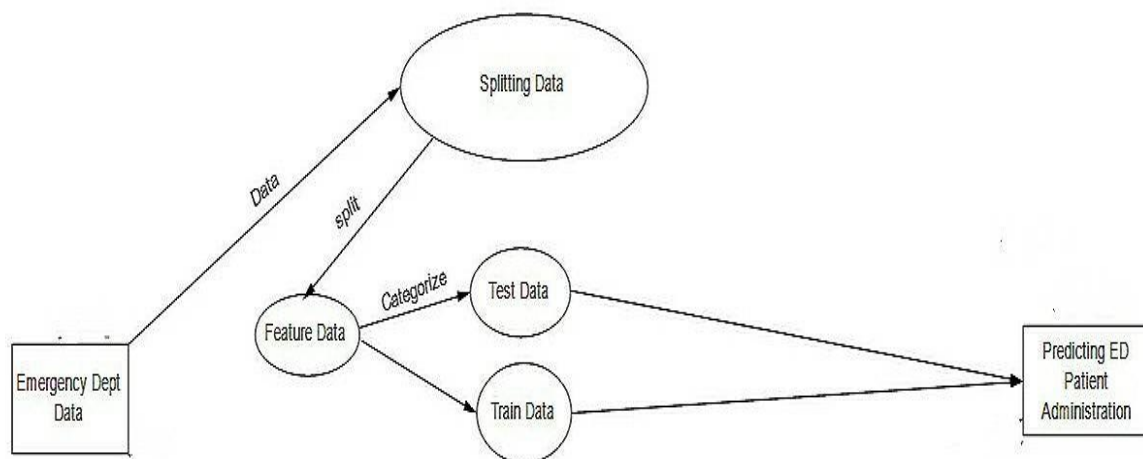


Fig.3.15 DFD Level for Splitting Data

3.6.4 DFD LEVEL FOR CLASSIFYING DATA

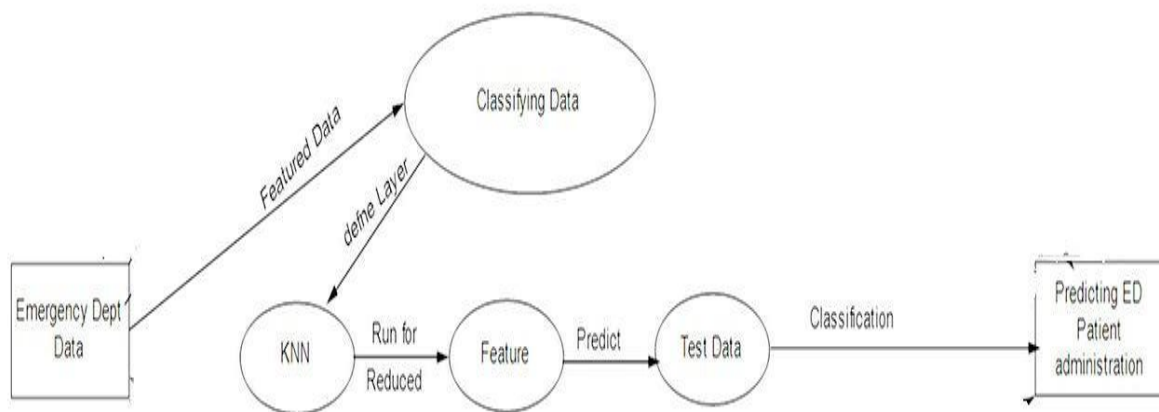


Fig.3.16 DFD Level for Classifying Data

It explains that Classification of data through DNN (Deep Neural Network) Algorithm Neural networks use randomness by design to ensure they effectively learn the function being approximated for the problem. Randomness is used because this class of machine learning algorithm performs better with it than without. The most common form of randomness used in neural networks is the random initialization of the network weights. A deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modeling to process data in complex ways. A neural network, in general, is a technology built to simulate the activity of the human brain – specifically, pattern recognition and the passage of input through various layers of simulated neural connections. Deep neural networks as networks that have an input layer, an output layer and at least one hidden layer in between. Each layer performs specific types of sorting and ordering in a process that some refer to as “feature hierarchy.”

3.6.5 DFD LEVEL FOR PREDICTION

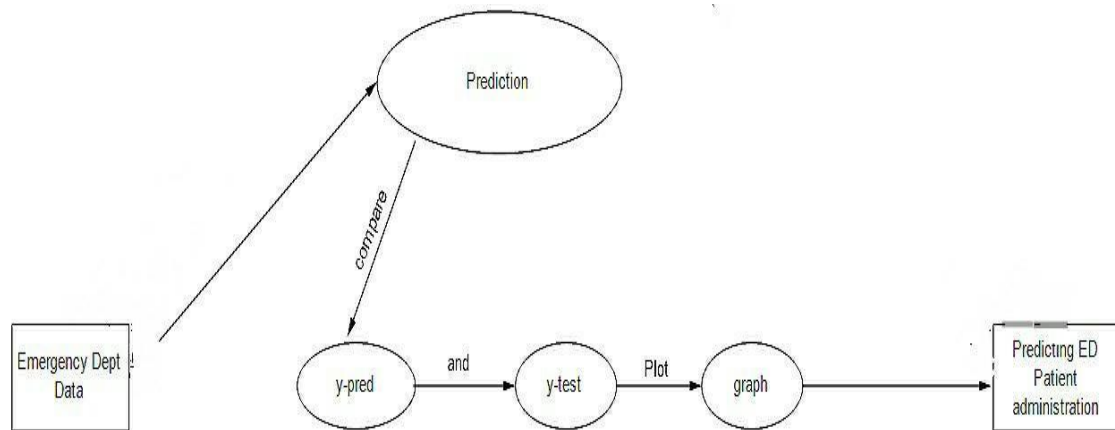


Fig.4.17 DFD Level for Prediction

It explains how the accuracy emergency department prediction occurs using the classifier, list with predicted values and plot the rainfall graph. **Predicting** the test set result. The prediction result will give you probability of patient admitting in the hospital. This will convert that probability into binary 0 and 1. 1 for attending in the hospital and 0 for not attending the hospital. This is the final step where this work evaluating our model performance. It predicts the Confusion matrix to check the accuracy of model.

3.7 USE CASE DIAGRAM FOR LOAD FORECASTING

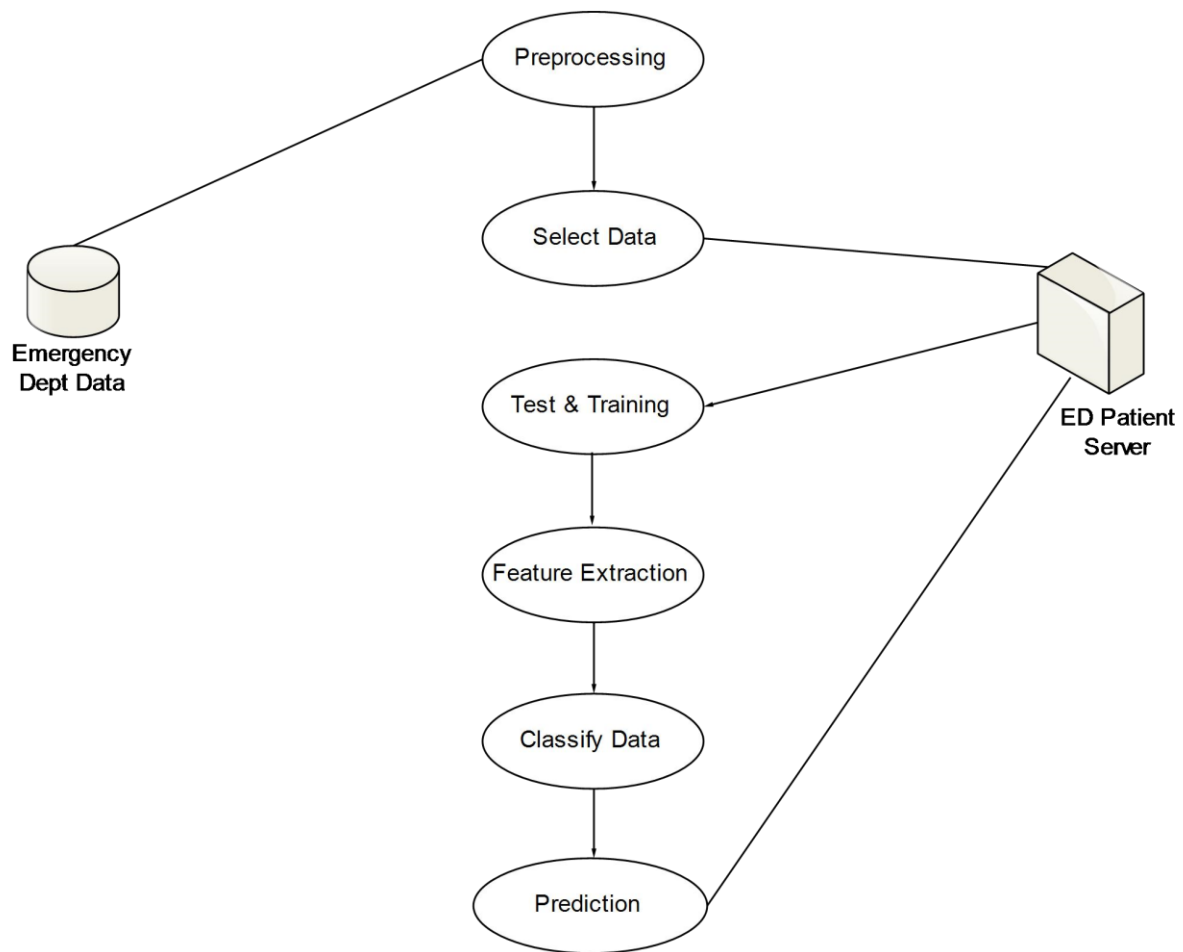


Fig.4.18 Use case diagram

Above the figure shows that it provides a global view of the actors involved in a system and the actions that the system performs, which in turn provide an observable result that is of value to the actors. Actors are an abstraction of the entity that interacts with the software. Here the system is partitioned simpler functions.

3.8 SEQUENCE DIAGRAM FOR LOAD FORECASTING

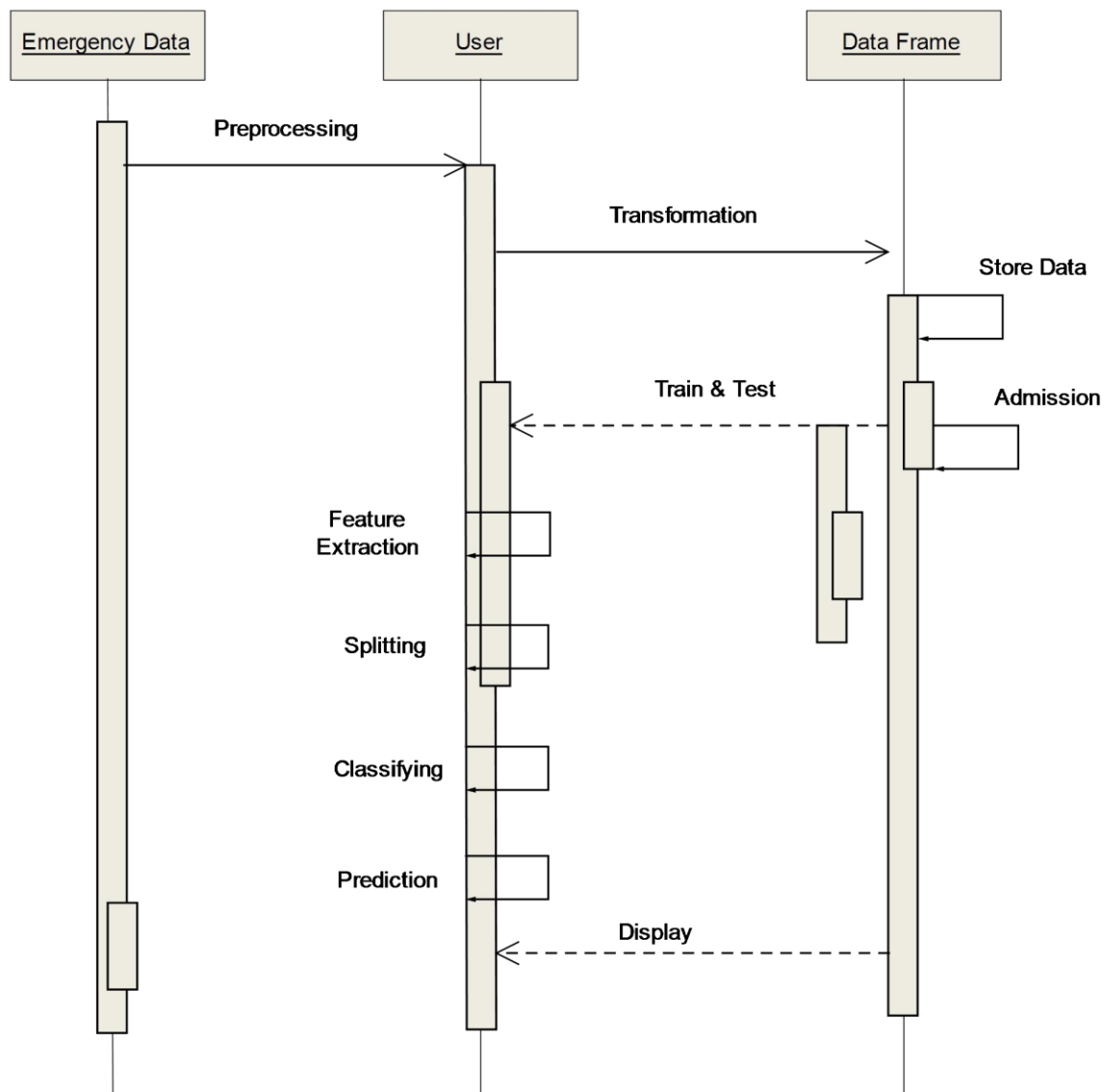


Fig.4.19 Sequence diagram

3.9 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 provide a way to check the functional of components. It is the process of exercising software with the intent 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 address a specific testing requirement.

For doing this testing process the person have to access to the source code of the product to be tested. So it is essential that the person doing this white box testing have some knowledge of the program being tested. Though not necessary it would be more worth if the programmer itself does this white box testing process since this testing process requires the handling of source code.

3.9.1 WHITE BOX TESTING

White box testing's testing which include the error in coding section it includes the error occurred during compliant and also during the development of the project. This project is verified and tested using white box testing. The following screenshots are the sample of the error coding and also how it is rectified. The error in the sample is because the coding is not proper. The coding is rewritten and corrected to solve the error.

[1] WHITE BOX TESTING FOR PRE-PROCESSING AND FEATURE EXTRACTION

It explains white box testing which describes the importing of label encoder and one hot encoder to transform the data from its bit level 0. Here found String Attribute value missing without suitable in label encoding declaration.

Error Description and Rectified Code

Here, White Box Testing for Pre-processing is done.

Data were collected and pre-processed within the dataset. Label encoding and one hot encoding were done.

[2] WHITE BOX TESTING FOR SPLITTING DATA

It explains white box testing which describes the White Box Testing for Name Error in the Splitting of weather data by layering the input and output. Here the error is naming the sequential classifier is not defined.

Error Description and Rectified Code

For Splitting Data the dataset are split into training data and testing data, the name of the classifier are not defined. So, it's a Naming Error for Splitting Data.

[3] WHITE BOX TESTING FOR PREDICTION AND ACCURACY

It explains white box testing which describes the White Box Testing for Compilation error for finding the emergency department admission prediction and accuracy. Naming index are not defined and it is out of range also database stored values are not match properly.

Error Description and Rectified Code

In Prediction and Accuracy, an error while declaring the values compile error occurs.

If values are not initialized, it's an out of range of index.

3.10 BLACK BOX TESTING

Black box testing is a testing method which helps to verify the model. It shows the total number of attempts needed to rectify an error. Test cases are built based on the specification and requirement for building the model.

3.10.1 Functional Testing

This is otherwise called as functional testing. In contrary to white box testing here the person who is doing the black box testing need not have the programming knowledge. This is because the person doing the black box testing would access the output or outcomes as the end user would access and would perform thorough functionality testing to check whether the developed module or product behaves in functionality in the way it has to be.

S. NO	INPUT	TEST DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
MOD 3	Data Preprocessing	If the dataset is given for preprocessing without cleaning.	Unwanted data are in dataset accuracy prediction is difficult.	Accuracy prediction is difficult due to unwanted data.	Pass
MOD 2	Data Visualization and descriptive statistics	Converting the layers from Float or Double to String.	Visualization is converted to String the expected graphs are generated.	While the String is converted the graph generation occurs.	Pass
MOD 3	Data splitting	Split the data for training and Testing process.	Data are splitted and the system were feed the input and output corrected.	Data were not splitted.	Fail
MOD 4	Predicting admissions	Using DNN and SVM classification of data occurs.	Epoch will occur until the count and comparison of two algorithm occurs.	Comparison of two algorithm m and epoch occur until the count.	Pass

MOD :	The evaluation of model performance	Comparison occurs and accuracy prediction of weather analyzed.	Confusion matrix generated and accuracy predicted.	Confusion matrix generated and accuracy predicted.	Pass
-------	-------------------------------------	--	--	--	------

Table 2.1 Test cases

Black box testing is a table analysis which helps the analyser to verify the project. It shows the number of attempts taken to resolve an error. Test cases are built based on the specification and requirement of the project. This project is verified and tested using test cases. These test case are built in the form of table this table includes the test description, input given expected output and the actual output it also tells the status of the test briefing whether it is fail or pass.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 EXPERIMENTAL RESULTS

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC										
id	dep_name	esi	age	gender	ethnicity	marital	status	employment	insurance	Estimated arrival	month	arrival	hour	previous	abdominal	alcohol	rel	allergy	anemia	anxiety	di	appendi	asthma	back	prob	birth	trau	bladder	cn	blindness	brain	nsca	breast	can	bronchitis	cardia	rs	
2	1 B		4	40	Male	Hispanic c	Single		Full Time	Other	101348.9	June	Tuesday	23-Feb	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	2 B		4	66	Male	Hispanic c	Married		Not Empl	Commerci	112342.6	January	Tuesday	15-18	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3 B		2	66	Male	Hispanic c	Married		Not Empl	Commerci	113931.6	July	Thursday	14-Nov	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	4 A		2	66	Male	Hispanic c	Married		Not Empl	Commerci	93826.63	July	Saturday	14-Nov	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	5 A		3	84	Female	Hispanic c	Widowed		Retired	Medicare	79084.1	November	Tuesday	10-Jul	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	6 A		3	86	Female	Hispanic c	Widowed		Retired	Medicare	149756.7	April	Monday	15-18	Admit	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	7 A		3	86	Female	Hispanic c	Widowed		Retired	Medicare	10062.8	September	Wednesday	14-Nov	Discharge	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	8 A		4	87	Female	Hispanic c	Widowed		Retired	Medicare	119346.9	March	Saturday	14-Nov	Admit	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	9 B		2	75	Male	Non-Hispi	Married		Retired	Medicare	74940.5	March	Sunday	6-Mar	Admit	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	10 B		2	75	Male	Non-Hispi	Married		Retired	Medicare	71725.73	October	Sunday	15-18	Admit	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	11 B		2	76	Male	Non-Hispi	Married		Retired	Medicare	80181.12	January	Tuesday	14-Nov	Admit	0	?	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	12 B		2	77	Male	Non-Hispi	Married		Retired	Medicare	76390.01	November	Friday	10-Jul	Admit	?	0	0	0	0	0	?	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	13 B		2	52	Male	Non-Hispi	Single		Full Time	Other	26260.98	November	Monday	15-18	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14 B		3	53	Male	Hispanic c	Significant		Disabled	Medicare	190657.8	March	Wednesday	23-Feb	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	15 B		3	53	Male	Hispanic c	Significant		Disabled	Medicare	65951.65	May	Sunday	19-22	Admit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	16 B		2	53	Male	Hispanic c	Significant		Disabled	Medicare	64327.26	July	Tuesday	14-Nov	Admit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	17 B		3	53	Male	Hispanic c	Significant		Disabled	Medicare	5097.67	August	Friday	14-Nov	Admit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	18 B		2	53	Male	Hispanic c	Significant		Disabled	Medicare	14406.41	September	Monday	14-Nov	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	19 B		1	53	Male	Hispanic c	Significant		Disabled	Medicare	158684.8	December	Friday	10-Jul	Admit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	20 A		3	33	Male	Hispanic c	Single		Full Time	Medicaid	54724.03	July	Thursday	19-22	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	21 A		4	49	Female	Hispanic c	Divorced		Not Empl	Medicaid	170866.2	December	Friday	19-22	No previo	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	22 A		2	50	Female	Hispanic c	Divorced		Not Empl	Medicaid	138555.5	March	Tuesday	23-Feb	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	23 A		2	51	Female	Hispanic c	Divorced		Not Empl	Medicaid	118913.5	June	Friday	14-Nov	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	24 A		3	45	Female	Hispanic c	Single		Not Empl	Medicaid	8487.75	February	Monday	14-Nov	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	25 B		2	45	Female	Hispanic c	Single		Not Empl	Medicaid	187616.2	March	Saturday	15-18	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	26 B		3	45	Female	Hispanic c	Single		Not Empl	Medicaid	124508.3	May	Tuesday	14-Nov	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	27 B		3	45	Female	Hispanic c	Single		Not Empl	Medicaid	170042	December	Sunday	15-18	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	28 B		4	46	Female	Hispanic c	Single		Not Empl	Medicaid	38433.35	February	Monday	14-Nov	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	29 B		3	46	Female	Hispanic c	Single		Not Empl	Medicaid	100187.4	May	Sunday	19-22	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	30 B		3	46	Female	Hispanic c	Single		Not Empl	Medicaid	53483.21	June	Tuesday	23-Feb	Discharge	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	31 A		3	50	Female	Hispanic c	Single		Full Time	Commerci	140469.4	July	Tuesday	19-22	Transfer b	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
33	32 A		5	50	Female	Hispanic c	Single		Full Time	Other	156731.9	July	Wednesday	10-Jul	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	33 A		5	51	Female	Hispanic c	Single		Full Time	Commerci	81898.81	August	Thursday	19-22	No previo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	34 A		4	51	Female	Hispanic c	Single		Full Time	Commerci	34410.55	December	Thursday	15-18	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	35 A		5	52	Female	Hispanic c	Single		Full Time	Commerci	142033.1	June	Wednesday	15-18	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	36 A		4	52	Female	Hispanic c	Single		Full Time	Commerci	27822.99	June	Thursday	10-Jul	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	37 A		5	52	Female	Hispanic c	Single		Full Time	Commerci	114066.8	July	Sunday	15-18	Discharge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig: 4.1 Dataset Collected

This data is from a retrospective study which included all adult ED visits. This dataset was derived from electronic health records of patient visits registered for emergency services only. The raw data used in this study is for Predicting hospital admission at emergency department using machine learning. The working model takes the data as input from the dataset which is available as .csv file.

FileHomeInsertPage LayoutFormulasDataReviewViewHelpTell me what you want to doShare

CutCopyFormat PainterClipboard

Calibri11A^A

B

I

U

Font

Wrap Text

Merge & Center

Alignment

General

%

Number

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Format

Cells

AutoSum

Fill

Clear

Sort & Filter

Find & Select

Editing

AD2

0

	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM
1	anxiety	di	appendici	asthma	backprobl	birthtraun	bladder	blindness	brainnsca	breastcan	bronchitis	cardiaarrs	cataract	chestpain	dizziness	paralysis	pneumon	Resource Allocated			
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

ED_Dataset

23:0812-03-2019

100%

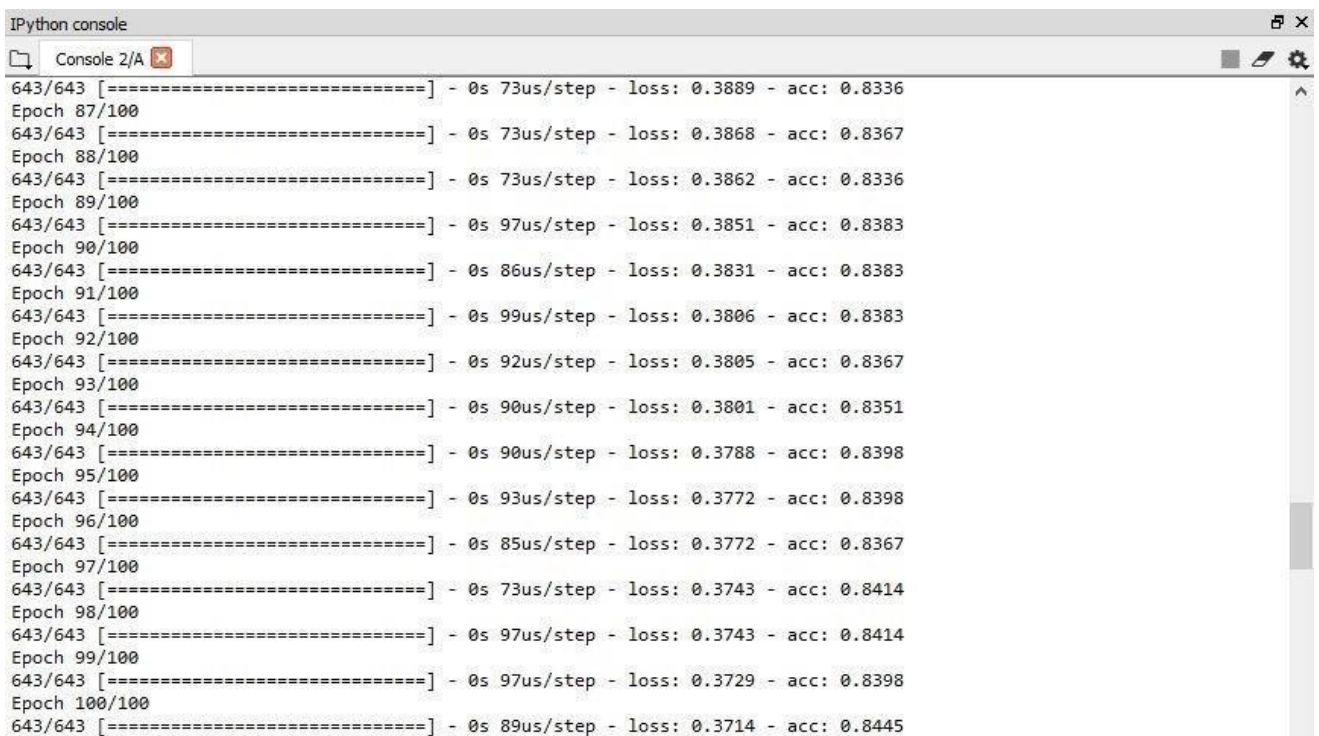
Fig: 4.2 Dataset For Cleaning

After cleansing, a Data Set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different Data Dictionary definitions of similar entities in different stores. Data cleaning differs from Data Validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

	0	1
0	108	9
1	12	32

Fig: 4.3 Accuracy Prediction For DNN

A confusion matrix is a technique for summarizing the performance of a classification algorithm. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. Classification accuracy is the ratio of correct predictions to total predictions made



```
IPython console
Console 2/A
643/643 [=====] - 0s 73us/step - loss: 0.3889 - acc: 0.8336
Epoch 87/100
643/643 [=====] - 0s 73us/step - loss: 0.3868 - acc: 0.8367
Epoch 88/100
643/643 [=====] - 0s 73us/step - loss: 0.3862 - acc: 0.8336
Epoch 89/100
643/643 [=====] - 0s 97us/step - loss: 0.3851 - acc: 0.8383
Epoch 90/100
643/643 [=====] - 0s 86us/step - loss: 0.3831 - acc: 0.8383
Epoch 91/100
643/643 [=====] - 0s 99us/step - loss: 0.3806 - acc: 0.8383
Epoch 92/100
643/643 [=====] - 0s 92us/step - loss: 0.3805 - acc: 0.8367
Epoch 93/100
643/643 [=====] - 0s 90us/step - loss: 0.3801 - acc: 0.8351
Epoch 94/100
643/643 [=====] - 0s 90us/step - loss: 0.3788 - acc: 0.8398
Epoch 95/100
643/643 [=====] - 0s 93us/step - loss: 0.3772 - acc: 0.8398
Epoch 96/100
643/643 [=====] - 0s 85us/step - loss: 0.3772 - acc: 0.8367
Epoch 97/100
643/643 [=====] - 0s 73us/step - loss: 0.3743 - acc: 0.8414
Epoch 98/100
643/643 [=====] - 0s 97us/step - loss: 0.3743 - acc: 0.8414
Epoch 99/100
643/643 [=====] - 0s 97us/step - loss: 0.3729 - acc: 0.8398
Epoch 100/100
643/643 [=====] - 0s 89us/step - loss: 0.3714 - acc: 0.8445
```

Fig: 4.4 Epoch

In neural networks generally, an epoch is a single pass through a full dataset. It is the iterations constituting one forward pass and one backward pass.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 SUMMARY

This study involved the development of a machine learning model aimed at predicting hospital admissions from the ED. This model was trained using routinely collected ED data using data mining algorithm.

The model presented in this study yield comparable, and in some cases improved performance compared to models presented in other studies. Implementation of the models as a decision support tool could help hospital decision makers to more effectively plan and manage resources based on the expected patient inflow from the ED. This could help to improve patient flow and reduce ED crowding, therefore reducing the adverse effects of ED crowding and improving patient satisfaction. The models also have potential application in performance monitoring and audit by comparing predicted admissions against actual admissions.

5.2 CONCLUSION

There are several practical applications of the models developed in this study. The predictions from the models can be automated and displayed in near real time in a clinical or performance management dashboard to assist with decision making. From a performance management and improvement perspective. The models can be used to compare the predicted decision to admit with the clinician's decision, thereby identifying patients who may have been admitted unnecessarily, or patients who typically would have been admitted.

Another benefit of implementing the model developed here is that it can help to improve planning and resource allocation in hospitals. Bed managers in the hospital would have advance information about the number of patients in the ED department who are likely to be admitted, which can be compared to bed availability to identify any potential shortfalls, which could result in delays to admission and hence longer stays in the ED department and overcrowding. Advance warning of hospital admissions can also provide the opportunity to make bed requests and preparations in advance of the admission. This is important for both the patient's experience, and from a performance management perspective.

5.3 FUTURE WORK

- Further work is needed to demonstrate the extent to which incorporation of the tool into clinical practice actually improves care or use of resources.
- The model could be used to support planning and decision making, individual level admission decisions still require clinical judgement.
- Testing the approach on large amounts of data will certainly strengthen the work, but enrolling a large population of users for the type of experiments takes a long time.
- Making improvements in the prediction model.

REFERENCES

- [1] Azadeh.A,Saberi.M, Ghaderi.S.F, and Ebrahimipour.V, "Improved estimation of electricity demand function by integration of fuzzy system and data mining approach," *Energy Convers. Manage.*, vol. 49, no. 8, pp. 2165–2177, 2008.
- [2] Baker DW, Stevens CD, Brook RH. Patients who leave a public hospital emergency department without being seen by a physician. Causes and consequences. *JAMA*. 1991;266:1085-1090.
- [3] Niu.D.X, Shi.H.F, and Wu.D.D, "Short-term load forecasting using Bayesian neural networks learned by hybrid Monte Carlo algorithm,"*Appl. Soft Comput.*, vol. 12, no. 6, pp. 1822–1827, 2012.
- [4] Eckstein M, Chan LS. The effect of emergency department crowding on paramedic ambulance availability. *Ann Emerg Med*. 2004;43:100-105.
- [5] Sulla.F, Koivisto.M, and Seppanen.J, "Statistical analysis and forecasting of damping in the nordic power system," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 306–315, Jan. 2015.
- [6] Liu.J, Fang.W, Zhang.X, and Yang.C, "An improved photovoltaic power forecasting model with the assistance of aerosol index data," *IEEE Trans. Sustainable Energy*, vol. 6, no. 2, pp. 434–442, Apr. 2015.
- [7] Liu S, Hobgood C, Brice JH. Impact of critical bed status on emergency department patient flow and overcrowding. *Acad Emerg Med*. 2003;10:382-385.
- [8] Rejc.M and Panto.M, "Short-term transmission-loss forecast for the Slovenian transmission power system based on a fuzzy-logic decision approach," *IEEE Trans.*

Power Syst., vol. 26, no. 3, pp. 1511–1521, Mar. 2011.

[9] Kezunovic.M, Xie.L, and Grijalva.S, “The role of big data in improving power system operation and protection,” in Proc. IEEE IREP Symp. Rethymnon Bulk Power Syst.

Dyn. Control-IX Optim. Security Control Emerging Power Grid, 2013, pp. 1–9.

[10] Schull MJ, Morrison LJ, Vermeulen M, et al. Emergency department overcrowding and ambulance transport delays for patients with chest pain. CMAJ. 2003;168:277283.

[11] Schull MJ, Morrison LJ, Vermeulen M, et al. Emergency department gridlock and out-of-hospital delays for cardiac patients. Acad Emerg Med. 2003;10:709-716.

[12] Hong.W.C, “Chaotic particle swarm optimization algorithm in a support vector regression electric load forecasting model,” Energy Convers. Manage., vol. 50, no. 1, pp. 105–117, 2009.

[13] Sun.W and Liang.Y, “Least-squares support vector machine based on improved imperialist competitive algorithm in a short-term load forecasting model,” J. Energy Eng., vol. 141, no. 4, pp. 04014037-1–04014037-8, 2015.

[14] Wang.Y, Xia.Q, and Kang.C, “Secondary forecasting based on deviation analysis for short-term load forecasting,” IEEE Trans. Power Syst., vol. 26, no. 2, pp. 500–507, May 2011.

[15] Li.Z et al., “Short-term wind power prediction based on extreme learning machine with error correction,” Protection Control Modern Power Syst., vol. 1, no. 1, pp. 1–8, 2016.

APPENDIX

A.Source Code

```
import numpy as np
import pandas as pd
import matplotlib
```

```
import theano
import tensorflow
import keras
```

```
dataset = pd.read_csv('D:\\jfpware\\Emergency Department\\ED_Dataset.csv')
X = dataset.iloc[:, 3:34].values
y = dataset.iloc[:, 34].values
```

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
```

```
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
```

```
X
labelencoder_X_3 = LabelEncoder()
X[:, 3] = labelencoder_X_3.fit_transform(X[:, 3])
```

```
X
labelencoder_X_4 = LabelEncoder()
X[:, 4] = labelencoder_X_4.fit_transform(X[:, 4])
```

```
labelencoder_X_5 = LabelEncoder()
```

```
X[:, 5] = labelencoder_X_5.fit_transform(X[:, 5])
```

```
labelencoder_X_6 = LabelEncoder()
```

```
X[:, 6] = labelencoder_X_6.fit_transform(X[:, 6])
```

```
labelencoder_X_7 = LabelEncoder()
```

```
X[:, 7] = labelencoder_X_7.fit_transform(X[:, 7])
```

```
labelencoder_X_8 = LabelEncoder()
```

```
X[:, 8] = labelencoder_X_8.fit_transform(X[:, 8])
```

```
labelencoder_X_9 = LabelEncoder()
```

```
X[:, 9] = labelencoder_X_9.fit_transform(X[:, 9])
```

```
labelencoder_X_10 = LabelEncoder()
```

```
X[:, 10] = labelencoder_X_10.fit_transform(X[:, 10])
```

```
labelencoder_X_11 = LabelEncoder()
```

```
X[:, 11] = labelencoder_X_11.fit_transform(X[:, 11])
```

```
labelencoder_X_12 = LabelEncoder()
```

```
X[:, 12] = labelencoder_X_12.fit_transform(X[:, 12])
```

```
X
```

```
labelencoder_X_13 = LabelEncoder()
```

```
X[:, 13] = labelencoder_X_13.fit_transform(X[:, 13])
```

```
X
```

```
labelencoder_X_14 = LabelEncoder()
```

```
X[:, 14] = labelencoder_X_4.fit_transform(X[:, 14]) labelencoder_X_15 =  
LabelEncoder()
```

```
X[:, 15] = labelencoder_X_15.fit_transform(X[:, 15])
```

```
labelencoder_X_16 = LabelEncoder()
```

```
X[:, 16] = labelencoder_X_16.fit_transform(X[:, 16])
```

```
labelencoder_X_17 = LabelEncoder()
```

```
X[:, 17] = labelencoder_X_17.fit_transform(X[:, 17])
```

```
labelencoder_X_18 = LabelEncoder()
```

```
X[:, 18] = labelencoder_X_18.fit_transform(X[:, 18])
```

```
labelencoder_X_19 = LabelEncoder()
```

```
X[:, 19] = labelencoder_X_9.fit_transform(X[:, 19])
```

```
labelencoder_X_20 = LabelEncoder()
```

```
X[:, 20] = labelencoder_X_20.fit_transform(X[:, 20])
```

```
labelencoder_X_21 = LabelEncoder()
```

```
X[:, 21] = labelencoder_X_21.fit_transform(X[:, 21])
```

```
labelencoder_X_22 = LabelEncoder()
```

```
X[:, 22] = labelencoder_X_22.fit_transform(X[:, 22])
```

```
X
```

```
labelencoder_X_23 = LabelEncoder()
```

```
X[:, 23] = labelencoder_X_23.fit_transform(X[:, 23]) X
```

```
labelencoder_X_24 = LabelEncoder()
```

```
X[:, 24] = labelencoder_X_24.fit_transform(X[:, 24])
```

```
labelencoder_X_25 = LabelEncoder()
X[:, 25] = labelencoder_X_25.fit_transform(X[:, 25])
```

```
labelencoder_X_26 = LabelEncoder()
X[:, 26] = labelencoder_X_26.fit_transform(X[:, 26])
```

```
labelencoder_X_27 = LabelEncoder()
X[:, 27] = labelencoder_X_27.fit_transform(X[:, 27])
```

```
labelencoder_X_28 = LabelEncoder()
X[:, 28] = labelencoder_X_28.fit_transform(X[:, 28])
```

```
labelencoder_X_29 = LabelEncoder()
X[:, 29] = labelencoder_X_29.fit_transform(X[:, 29])
```

```
X
labelencoder_X_30 = LabelEncoder()
X[:, 30] = labelencoder_X_30.fit_transform(X[:, 30])
```

```
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```

import keras from keras.models
import Sequential from keras.layers
import Dense

classifier = Sequential()

classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu', input_dim =
31)) classifier.add(Dense(output_dim = 6, init = 'uniform', activation = 'relu'))
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'sigmoid'))
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =
['accuracy']) classifier.fit(X_train, y_train, batch_size = 10,
nb_epoch = 100)

y_pred = classifier.predict(X_test) y_pred
= (y_pred > 0.5)

from sklearn.metrics import confusion_matrix from
sklearn.metrics import accuracy_score

cm = confusion_matrix(y_test, y_pred) print(cm)
print('Accuracy' + str(accuracy_score(y_test, y_pred)))

from sklearn.metrics import confusion_matrix cm
= confusion_matrix(y_test, y_pred)

import numpy as np import
pandas as pd import seaborn
as sns sns.set_palette('husl')
import matplotlib.pyplot as plt

```

```
tmp = dataset.drop('id', axis=1)
```

```
g = sns.violinplot(y='disposition', x='esi', data=dataset, inner='quartile') plt.show()
```

```
g = sns.violinplot(y='disposition', x='age', data=dataset, inner='quartile') plt.show()
```

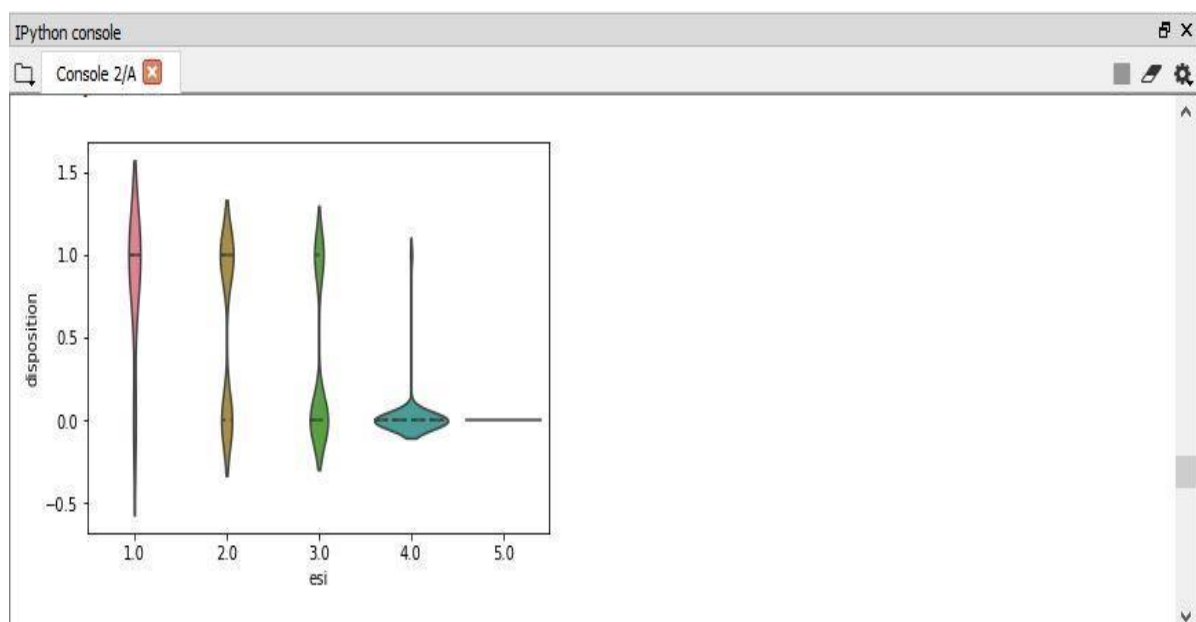
```
g = sns.violinplot(y='disposition', x='asthma', data=dataset, inner='quartile') plt.show()
```

```
g = sns.violinplot(y='disposition', x='allergy', data=dataset, inner='quartile') plt.show()
```

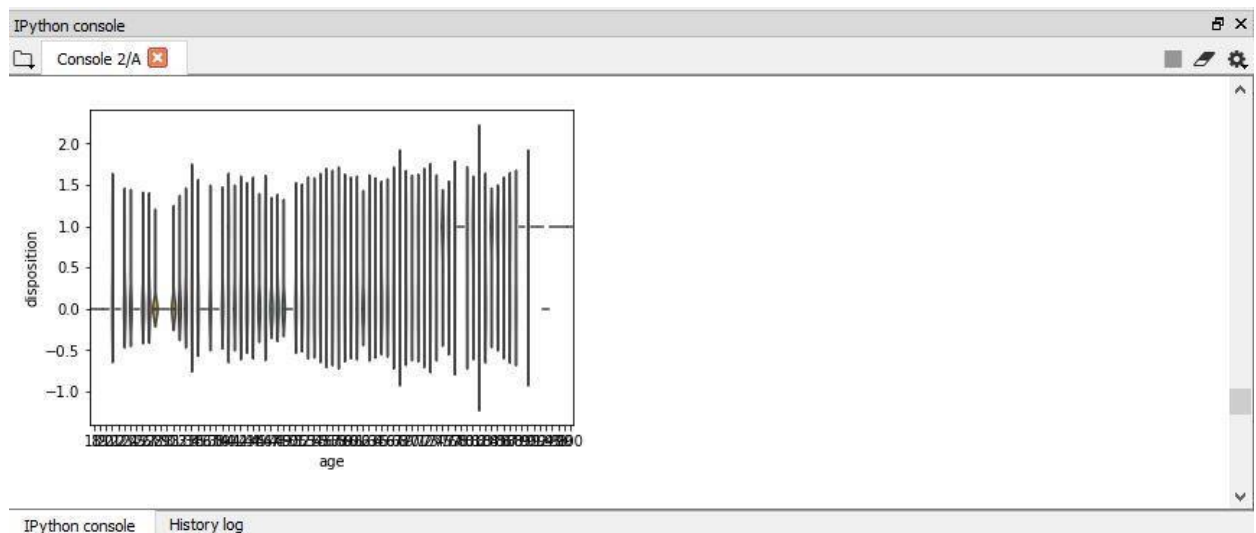
B.Screenshots

Variable explorer			
Name	Type	Size	Value
X	float64	(804, 31)	[[1. 40. 0. ... 0. 0. 0.] [1. 66. 0. ... 0. 0. 0.]
X_test	float64	(161, 31)	[[-0.92359403 -0.56151227 0.63078609 ... -0.2011295 -0.15974461 -0 ...
X_train	float64	(643, 31)	[[-0.92359403 -0.16874383 0.63078609 ... -0.2011295 -0.15974461 -0 ...
cm	int64	(2, 2)	[[94 14] [22 31]]
dataset	DataFrame	(804, 35)	Column names: id, dep_name, esi, age, gender, ethnicity, maritalstatus ...
tmp	DataFrame	(804, 34)	Column names: dep_name, esi, age, gender, ethnicity, maritalstatus, em ...
y	int64	(804,)	[0 0 0 ... 1 0 0]
y_pred	bool	(161, 1)	[[False] [True]
y_test	int64	(161,)	[0 0 1 ... 1 0 1]
y_train	int64	(643,)	[0 1 0 ... 0 0 0]

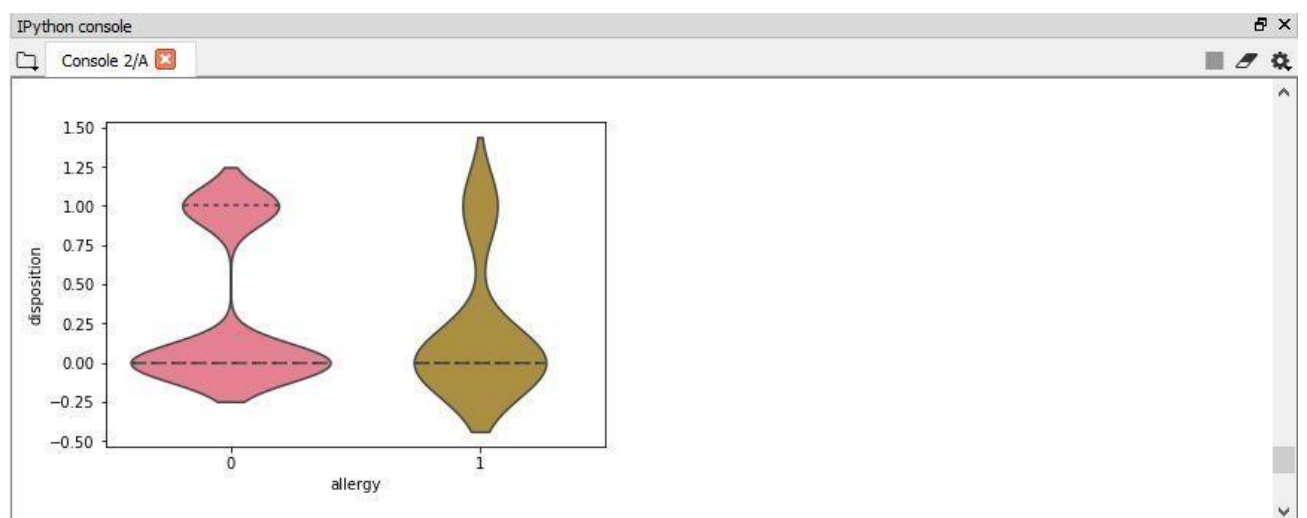
Splitting Of Data



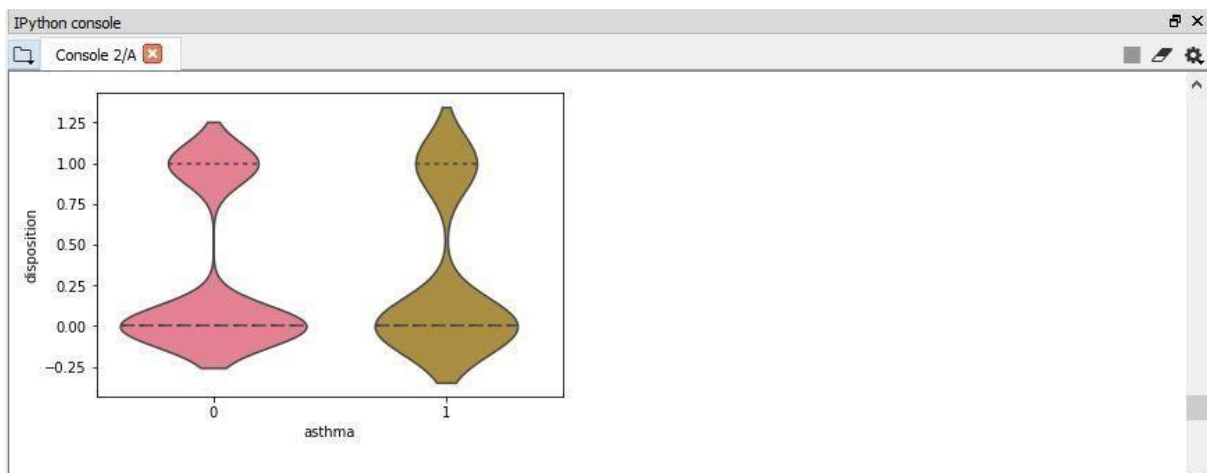
Esi Vs Disposition



Age Vs Disposition



Allergy Vs Disposition



Asthma Vs Disposition

This certifies that
Goutham kumar
has successfully completed the freeCodeCamp.org
Data Analysis with Python
Developer Certification, representing approximately 400 hours of coursework



Quincy Larson
Executive Director, freeCodeCamp.org

Verify this certification at <https://freecodecamp.org/certification/gouthamkumar/data-analysis-with-python-v7>