Deep Learning with MIMIC-III

Deep Learning Assisted Mortality Prediction Based on Prescription Information

Mehmet Demir
PhD Candidate Computer Science
Ryerson University
Toronto, Canada
mehmet.demir@ryerson.ca

***Abstract*—Healthcare science consists of science and the technology to diagnose the illnesses and treat the patients. It is one of the most important research areas of any emerging technologies. Newly discovered technologies and new ideas are almost immediately taken to healthcare sciences to see if there is a match or a value. Deep learning is what we are focusing on in this research. We are looking for improved results in one of the areas of healthcare by applying deep learning methodologies.**

One of the biggest challenges in the deep learning research is finding benchmark datasets to validate research. It is not easy to find publicly available, reliable and high quality benchmark datasets that we can easily access, process and compare. This challenge is only amplified further in the heath care technologies since the privacy concern is a big limitation for publishing datasets. Therefore finding a dataset to work was one of the first problems to address.

We demonstrate the deep learning experiment on Medical Information Mart for Intensive Care (MIMIC-III) database. In this database, there are several aspects of the Intensive Care Unit procedures recorded. Our study is targeting improvements in the accuracy of ICU processes. Any discovery of multi-dimensional relationships between different entities is very valuable. With this in our mind, we chose this database and decided to apply deep learning techniques.

We follow several steps of data science on this publicly available data store. We are targeting to use the computer aided decision-making systems to predict the outcome of healthcare delivery. This document summarizes the findings of the research project. The target of the project is to use the deep learning techniques and tools to understand applications of the deep learning.

Keywords—MIMIC-III, Keras, Python, Tensorflow, Deep Learning

I. INTRODUCTION

Each year, millions of patients are admitted to ICU units in the United States. About a quarter million patients are admitted to Canadian units. Numbers are increasing faster than the number of hospital admissions. There are problems about resource planning, capacity planning and quality [1].

Healthcare systems can be improved to prevent medical errors that are the leading causes of injury and death[2]. These preventable medical errors in medical care can be identified earlier with advanced data processing mechanisms that can predict the outcomes.

It is not a surprise to see the deep learning to be applied to health care domain. Health care is complex[10]. Data is big. Originator of the data is distributed geographically as well as the health care providers. Health care systems are also decentralized and data is in multiple locations/ databases/ systems. Several systems create and store the information. Multiple standards are used even in the same area in the same country due to diversity of the manufacturers.

Data format used in health care is very broad. Images are used as well as comments. Most data are stored in free form text format. Variety of the data had been an obstacle for reaching desired standards. Data is structured in some common fields and unstructured in most specific fields. This inconsistent standards in data structures create further complexities when the changing requirements and regulations are applied in independent schedules.

As a response to above mentioned issues, health care management decided to utilize electronic health records (EHR)[13] and reached for the big data technologies. This standardization enables researchers to apply machine learning techniques to the collected standard data. Researchers are targeting to build clinical decision support tools from the findings of the deep learning.

With all above reasons and criteria, we decided to focus on the heath care. We believe any significant step in above mentioned opportunities would be a great benefit to public health.

The structure of this report reflects the data science steps we took in order to process the MIMIC-III data[11]. This introduction summarizes the heath care industry, heath care data and its presented deep learning opportunity. We continue this report with relevant work and existing research. Our research target will be summarized in this title as well. Then, first section will introduce the data source. It will include the steps we took in order to prepare the data source. Second section will be on the preparation steps for the data and the extraction. This step will detail the Keras module we developed. And the last sections will evaluate the results and provide criticism on the work.

II. Related Work

A. Origins

Data is the heart of the deep learning research. Depending on the quality and structure of the data, learning and outcomes may change. Quality of the data has direct relationship with the quality of the research.

MIMIC-III (Medical Information Mart for Intensive Care III) is public available health-related database. Its data consists of the information collected and de-identified from thousands of patients. Our dataset indicate number of patients is more than fifty thousand. Data is collected from the critical care unit systems of the Beth Israel Deaconess Medical Center.

MIMIC-III database includes patient information such as demographics as well as some periodical measurements made at the bedside. Lab results and other healthcare information such as drugs, caregiver notes, imaging reports and mortality are included in the data.

"MIMIC-III is notable for three factors: It is freely available to researchers worldwide. It encompasses a diverse and very large population of ICU patients. It contains high temporal resolution data including lab results, electronic documentation, and bedside monitor trends and waveforms"[6].

B. Existing Research

MIMIC-III database is used in several studies. Data were used to research on illnesses such as obesity of the patients [12], data were also used to evaluate the data quality issues in clinical data [3].

Heart rate (HR) and blood pressure (BP) values are used in research for patient monitoring and outcome prediction[9]. They also included mortality prediction in their research.

In our research and in this report, we are focusing the impact of a variety of drugs in to the outcome such as mortality. There are more precise research focusing on only one drug (Heparin) which is identified by our study as 18th most prescribed drug in the MIMIC-III database[5].

And there are several other studies with this database that focuses on different medical areas such as red blood cell transfusion[4], fluid balance in ICU patients[8] and hypotension[7].

C. Target Research

In this report, we propose a deep learning application for a clinical problem: Mortality in ICU in relation to the prescriptions. Our application is built to work on the data provided by the MIMIC-III database. This data is suitable to work on several clinical condition problems as listed in above related work section.

Deep learning is known to provide regression and classification solutions. Beyond all the known applications of deep learning, analyzing the old data in variety of formats and provide regression of outcome also teach us about the complexity of the data. By including and excluding fields in the regression, we can see the impact of each field of data in learning. This also shows us about the impact of the field to the outcome. How much the regression can be improved with the additional data using the same deep learning model also give us information about the data. Knowing the quality of the data also help the heath care IT groups and IT system providers to improve their systems. It can lead to new clues about grouping and clustering their information.

The target of our research is to provide a deep learning application on the MIMIC-III data. Experience of the deep learning application development will be detailed in the report. Whether the learning will be successful or not, the deep learning data preparation, application development, application testing and results evaluation processes are the outcome of the research. The steps followed can be guide to future researchers who want to work on this database or a similar database.

III. Data

A. Recreation of the database

MIMIC-III database comes with downloadable CSV files. CSV files are good to store single dimensional data but having a multi-dimensional analysis on CSV files is not possible. In order to gather multi-dimensional data, we need to collect data from multiple CSV files that are representing a complete information.

Therefore, first step in my research was to load to data in to a database where I can run advanced queries. Since the data is relational and limited, the simplest solution appeared to be a relational database. And we started using the scripts coming with the MIMIC GitHub sources. The table structures DDL was given and creation of were mostly successful other than some partition logic. This partition DDL was for a non-functional requirement and was only related to performance, therefore we skipped this structure.

We continued the database creation with loading the CSV files in to the database. This activity took several days. Several glitches were encountered and hours of cleaning had to be done. This experience thought us an important lesson about the shared data. Quality of the data and the platform to use the data is not in industry quality. Industry practice for such sharing would be either virtual machines or Docker images. They would be downloaded and used without additional pain of recreating the database and reloading data.

After the data is loaded, We continued with creation of indices. Indices file provided were mostly successful for small tables. But for the tables with large amount of data, they were not successful. Overnight running of the index creation on some specific tables failed. 8-10 hours were not enough to create some of these indices. There were not much guidance in the MIMIC-III resources about alternatives to these issues.

Finally, we completed loading of the data in the desired structure with its several dimensions and entities.

IV. FINDING THE TARGET DATA

MIMIC-III database has several tables: ADMISSIONS, D_CPT, LABEVENTS, SERVICES, CALLOUT, D_ICD_DIAGNOSES, MICROBIOLOGYEVENTS, TRANSFERS, CAREGIVERS, D_ICD_PROCEDURES, NOTEEVENTSCHARTEVENTS, D_ITEMS, OUTPUTEVENTS, CPTEVENTS, D_LABITEMS, PATIENTS, DATETIMEEVENTS, ICUSTAYS, PRESCRIPTIONS, DIAGNOSES_ICD, INPUTEVENTS_CV, PROCEDUREEVENTS_MV, DRGCODES, INPUTEVENTS_MV and PROCEDURES_ICD

We started with discovering the tables in order to find a relevant regression or classification idea. ADMISSIONS table has several patient data such as DEATHTIME. Even the religion and ethnicity of the patient are recorded. Several tables like CAREGIVER and TRANSFERS were not very fruitful in deep learning as they have a lot of repeating information for several patients.

We decided to start working on the PRESCRIPTIONS table. We believe finding a prediction opportunity on the prescriptions that the patient use during the ICU visit can be most valuable. A significant learning from the above mentioned data can point out to best practices or mistakes in the ICU. If a prescription is dominant in the outcome of the mortality, this could be a significant finding. A combination of prescriptions also could indicate issues. Side effects of the drugs in combination are rarely tested in pharmaceutical studies. If the outcome would indicate a combination effect, this would be a significant outcome as well.

In order data to be ready for training tools, we remodeled the data. We created two new tables. The first table is for creating an identification for prescriptions. It also store the occurrences/frequency of prescriptions. This was necessary since we decided to use the prescriptions in deep learning and the prescriptions were not included in a table with codes or identifiers attached to them. Below is the DDL of this new table.

```
DROP TABLE M_PRESCRIPTIONS;
CREATE TABLE "M_PRESCRIPTIONS"
   (    "ROW_ID" NUMBER(10,0),
        "DRUG" VARCHAR2(80),
        "OCCURANCE" NUMBER(10,0)
   );
```

We loaded this table from the PRESCRIPTIONS table with the following SQL.

```
INSERT INTO M_PRESCRIPTIONS(ROW_ID, DRUG,
OCCURANCE)
SELECT M_SEQUENCE.NEXTVAL, DRUG, OCCURANCE
FROM (
     SELECT DRUG DRUG, COUNT (DRUG) OCCURANCE
     FROM PRESCRIPTIONS GROUP BY DRUG ORDER BY
     COUNT(DRUG) DESC
);
```

As a result of above operation, we could query which drugs are used by how many patients. We assigned a code for each drug. ROW_ID column is assigned to values starting from sequence number 3000, which happened to be start of the database sequence number we were using.

	ROW...	DRUG	OCCURANCE
1	3001	Potassium Chloride	192993
2	3002	Insulin	143465
3	3003	D5W	142241
4	3004	Furosemide	133122
5	3005	0.9% Sodium Chloride	130147
6	3006	NS	129731
7	3007	Magnesium Sulfate	90427
8	3008	Iso-Osmotic Dextrose	87005
9	3009	Sodium Chloride 0.9% Flush	83392
10	3010	Acetaminophen	78768
11	3011	Metoprolol	73986
12	3012	5% Dextrose	73829
13	3013	SW	72522
14	3014	Morphine Sulfate	62134
15	3015	Metoprolol Tartrate	59824
16	3016	Lorazepam	55352
17	3017	Calcium Gluconate	52110
18	3018	Heparin	52069
19	3019	Docusate Sodium	45694
20	3020	Vancomycin	42634
21	3021	Bisacodyl	41075
22	3022	Warfarin	39678

Main data modelling exercise was to collect the usable data for the deep learning tools. In order deep learning tools can use the data, best format is numeric format. Some fields were already numeric. Some other fields had to be processed in order to be numeric. For instance, categorization of the patients were needed in order to have a data whether they are a newborn, old person or a normal adult.

We created a table to represent all the data that we needed in this project. Collecting all the data in to a new table is key to create very simple deep learning networks. With a single table of denormalized data in the right format, all the data can be extracted and loaded to the neural network as is without further preparation. We did some ETL work and then finally, we had a table of the usable data.

```
DROP TABLE M_PATIENT_DENORM;
CREATE TABLE M_PATIENT_DENORM
   (    ROW_ID INT,
        "SUBJECT_ID" NUMBER(7,0),
        "HADM_ID" NUMBER(7,0),
        "ICUSTAY_ID" NUMBER(7,0),
        "AGE" NUMBER(5,2),
        "PREICULOS" NUMBER(5,2),
        "LENGTHOFSTAY" NUMBER(5,2),
        "HOSPITAL_EXPIRE_FLAG" NUMBER(1,0),
        "ADULT" NUMBER(1,0),
        "NEONATE" NUMBER(1,0),
        "OLDAGE" NUMBER(1,0),
"M3001" INT,"M3002" INT,"M3003" INT,"M3004"...
INT,"M3901" INT,"M3900" INT
    );
```

It is noticeable that the columns starting with M3001. These columns are created to store the prescriptions for a specific patient. We ran an ETL process in order to update these columns.

First we filled the patient data to this table. The code related to this is included in Appendix 1 – SQL for Loading The Data to Denormalized Table. We started with some of the SQL scripts from the MIMIC-III web site and customized them for specific fields and format. Important part of this script is to prepare the format of the data to be loaded in to the python.

Finally we created a java program to process the existing data and load the prescription usage information on this table. Related code used to load the occurrences of prescriptions in to the denormalized table are included in the Appendix 2 – Java Application For Loading The Prescription Data To Denormalized Table.

At this point the python loader was failing because of the non-numeric fields in the data. Therefore we updated the prescription data to be set to zero where it's value was null. Normally a data modification like this might be considering as changing and invalidating the data. But this change is meaningful as the null value means that this field is not used since there is no occurrence of the specific prescription for this patient. This operation helped the leaning tools. The code is included in the Appendix 3 – Java Application For Cleaning The Denormalized Data.

V. Extracting the Data

We chose a large list of fields. We loaded them in to the newly created M_PATIENT_DENORM table. When time came to extract the data, we decided to take the first 450 of the drugs beside the main information I had about the patient. The main reason to take the first 450 fields is that we believe as the drugs are used less, there would be less relationships with the data we wanted to do learning on.

At the extraction we had three output files. First file "export450Data.csv" has more than 450 columns. Fields are ("SUBJECT_ID", "HADM_ID", "ICUSTAY_ID", "AGE", "PREICULOS", "ADULT", "NEONATE", "OLDAGE", "M3001" … "M3450"). Other two files are for the labels. For instance, the file "export450HospitalExpire.csv" has field name "HOSPITAL_EXPIRE_FLAG". We also extracted another file named "export450LengthOfStay.csv" in order to label "LENGTHOFSTAY" field.

VI. Deep Learning

A. Tools

Main deep learning tool we used is Keras with TensorFlow backend and Python3. Detailed installation information of this suite is widely available on the internet.

The reason of using Keras is that programming is very easy with Keras. Creating a network an testing is generally less than 20 lines. There are easy to use libraries to define the network. There are keywords to specify the fine tuning details. There is no need to worry about gradient based learning details. All the implementation is part of the libraries and can be used with simply calling methods.

The output of the Keras application is also very verbose. Any level of details can be extracted. All the fine grained information can be monitored or reported at the end. With these properties, Keras is very suitable for research project such as this one.

B. Neural network selection

As we formatted the data in to independent columns of information and since the data is numeric, the data would enter to our learning system as an array of independent numeric values. Best approach to form a network to use this kind of data is to use a deep feedforward neural network.

We did not use other techniques such as convolution (CNN) or RNN/LSTM. Convolution is not suitable approach for the problem we are tackling since the columns do not have neighbor relationship with each other. Such neighbor relationship that we see in pixels of an image does not exist in the prescriptions we extracted. Prescriptions in general may have this behaviors as they can be given as a cocktail of medicines. But in our model, the neighbors are purely selected with the order of frequency. It is independent of the co-usage. Therefore no meaningful feature extraction can be done with convolution in our example.

RNN/LSTM is also not a suitable approach as the data we focused are not time series data. Carrying a memory to the next time frame would not benefit the outcome. Even the order of the prescription usage is not captured in this system and therefore RNN is not used. There are some other data in MIMIC-III that can be time series. Our research does not focus on those tables.

C. Code

We implemented the python code in order to represent the neural network configurations we wanted to try. Each trial was a change in code and consecutive testing.

The code is structured as following. First section of a python code is to define libraries. The next section would be the first functional section where we load the data. Here we introduce the files, define the structure and load them. Next section will be the definition and validation of the network. In Keras modules, network will be defined layer by later. Here we define how many inputs are there in the network. Then, for each layer, how many nodes/neurons there will be. How are they connected are defined. A dense layer for instance will be a simple type describing a feed forward layer that is fully connected to the next layer. At the end of the network configuration we validate and create the network with a compile statement. Following this section is where we run the learning algorithm with the fit command. This is where we give the learning set of data, test data, define number of epochs and batches.

Further details of the network are discussed in below results section. Our experiment included trying different configuration of the network in order to see the impact and improvements. Final code of the deep learning application is included below in Appendix 4 – Deep Learning Python Application Using Keras. We tried several alternative configurations. Some of these alternatives are also detailed below. Numerous unsuccessful tests are not included in this report.

VII. RESULTS

A. Single layer network

We used several different configurations for the feed forward network. First significant configuration we experimented consisted of only one layer that consist of hundreds of nodes. This single layer had an accuracy of only 0.1. This result proven that the network had to be more deep. This prove that a simple logic cannot solve this problem. Is the solution would be simple such as a coin toss, a single layer could have solve the problem.

```
mimic.py:44: UserWarning: Update your `Dense` call to the
Keras 2 API: `Dense(1, input_dim=410,
activation="sigmoid", kernel_initializer="uniform")`
  model.add(Dense(1, input_dim=410, init='uniform',
activation='sigmoid'))
Epoch 1/5
61522/61522
[==============================] - 9s
153us/step - loss: 14.2300 - acc: 0.1074
```

B. Multi Layer Deep network

After several tries, We finally found below network to be most efficient in the target task.

a) Input layer consists of 458 input. (450 drugs and 8 other information fields) outputting 400.

b) Second layer has the 400 input and 100 output.

c) Third layer is for 100 input and 1 output.

This multi-layer, feed forward, fully connected network got best results. These results are listed below.

```
mimic.py:27: UserWarning: Update your `Dense` call to the
Keras 2 API: `Dense(400, input_dim=458, activation="tanh",
kernel_initializer="uniform")`
  model.add(Dense(400, input_dim=458, init='uniform',
activation='tanh'))
mimic.py:29: UserWarning: Update your `Dense` call to the
Keras 2 API: `Dense(100, activation="tanh",
kernel_initializer="uniform")`
  model.add(Dense(100, init='uniform', activation='tanh'))
mimic.py:31: UserWarning: Update your `Dense` call to the
Keras 2 API: `Dense(1, activation="sigmoid",
kernel_initializer="uniform")`
  model.add(Dense(1, init='uniform', activation='sigmoid'))
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
50000/50000
[==============================] - 39s
782us/step - loss: 1.0704 - acc: 0.8891 - val_loss: 0.3332 -
val_acc: 0.9043
```

When the configuration was changed to different numbers of neurons in each layer, the benefit was not significant. Therefore this configuration is accepted as final.

With an accuracy of 0.9043 this network is an improvement. The learning is successful.

In the first layer there are several weights that are 0. Which shows that the system learned to ignore these input. Some of the prescriptions are proven to be not related to the death information we have in the tables. This also proves that model is successful.

We experimented similar methodology with other information that was in the MIMIC-III database. Results are proving that some information are not related and cannot be used for deep learning of regression. Length of stay with its minimal accuracy is a proof of that.

```
Epoch 1/10 -  10000/10000
[==============================] - 2s
231us/step - loss: -49.2647 - acc: 0.0041 - val_loss: -62.9234
- val_acc: 0.0048
```


ACKNOWLEDGMENT

Thanks to Professor Alireza Sadeghian for his help in authorizing access to the MIMIC-III database.

REFERENCES

[1] *Care in Canadian ICUs*. (2016, 08). Retrieved 12 1, 2017, from Canadian Institute of Health Information: https://secure.cihi.ca/free_products/ICU_Report_EN.pdf

[2] Cohan, A., Fong, A., Ratwani, R. M., & Goharian, N. Identifying Harm Events in Clinical Care through Medical Narratives. *8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics (ACM-BCB '17)* (pp. 52-59). New York: ACM.

[3] Combi, C., Mantovani, M., & Sala, P. (2017). Discovering Quantitative Temporal Functional Dependencies on Clinical Data. *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, (pp. 248-257). Park City.

[4] Dejam, A., Malley, B. E., Feng, M., Cismondi, F., Park, S., Samani, S., et al. (2014). The effect of age and clinical circumstances on the outcome of red blood cell transfusion in critically ill patients. *Critical Care* .

[5] Ghassemi, M. M., Richter, S. E., Eche, I. M., Chen, T. W., Danziger, J., & Celi, L. A. (2014). A data-driven approach to optimized medication dosing: a focus on heparin. *Intensive Care Medicine , 40* (9), 1332--1339.

[6] Johnson, A., Pollard, T., Shen, L., Lehman, L., Feng, M., Ghassemi, M., et al. (n.d.). MIMIC-III, a freely accessible critical care database. *Scientific Data (2016)* . DOI: 10.1038/sdata.2016.35.

[7] Lee, J., Kothari, R., Ladapo, J., Scott, D., & Celi, L. A. (2012). Interrogating a clinical database to study treatment of hypotension in the critically ill. *BMJ Open* .

[8] Lee, J., Louw, E. d., Niemi, M., Nelson, R., Mark, R., Celi, L., et al. (2015). Association between fluid balance and survival in critically ill patients. *J Intern Med.* , 468–477.

[9] Lehman, L. w. A Physiological Time Series Dynamics-Based Approach to Patient Monitoring and Outcome Prediction. *IEEE Journal of Biomedical and Health Informatics*, *19*, pp. 1068-1076. A Physiological

Time Series Dynamics-Based Approach to Patient Monitoring and Outcome Prediction.

[10] LeSueur, D. (2017). *5 Reasons Healthcare Data Is Unique and Difficult to Measure*. Retrieved 11 20, 2017, from Heath Catalyst: https://www.healthcatalyst.com/5-reasons-healthcare-data-is-difficult-to-measure

[11] *MIMIC-III*. (n.d.). Retrieved from Physionet: https://mimic.physionet.org/about/mimic/

[12] Pan, J., Shaffer, R., Sinno, Z., Tyler, M., & Ghosh, J. (2017). The obesity paradox in ICU patients. *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, (pp. 3360-3364). Seogwipo.

[13] Sha, Y., & Wang, M. D. (2017). Interpretable Predictions of Clinical Outcomes with An Attention-based Recurrent Neural Network. *8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics (ACM-BCB '17)* (pp. 233-240). New York: ACM.

VIII. Appendix 1 – PEND for Loading The Data to Denormalized Table

```
INSERT INTO M_PATIENT_DENORM (SUBJECT_ID, HADM_ID, ICUSTAY_ID, AGE, PREICULOS, LENGTHOFSTAY,
NEONATE, OLDAGE, ADULT, HOSPITAL_EXPIRE_FLAG)
SELECT IE.SUBJECT_ID SUBJECT_ID, IE.HADM_ID HADM_ID, IE.ICUSTAY_ID ICUSTAY_ID,
    ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) AS AGE,
    ROUND((CAST(IE.INTIME AS DATE) - CAST(ADM.ADMITTIME AS DATE))/365.242, 2) AS PREICULOS,
    ROUND((CAST(IE.OUTTIME AS DATE) - CAST(IE.INTIME AS DATE)), 2) AS LENGTHOFSTAY,
    CASE
        WHEN ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) <= 1
            THEN 1
            ELSE 0
        END AS NEONATE,
    CASE
        WHEN ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) > 100
            THEN 1
            ELSE 0
        END AS OLDAGE,
    CASE
        WHEN ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) <= 1
            THEN 0
        WHEN ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) <= 14
            THEN 0
      WHEN ROUND((CAST(IE.INTIME AS DATE) - CAST(PAT.DOB AS DATE))/365.242, 2) > 100
            THEN 0
        ELSE 1
        END AS ADULT,
CASE
        WHEN ADM.HOSPITAL_EXPIRE_FLAG = 1 THEN 1
    ELSE 0
    END AS HOSPITAL_EXPIRE_FLAG

FROM ICUSTAYS IE
INNER JOIN PATIENTS PAT
ON IE.SUBJECT_ID = PAT.SUBJECT_ID
INNER JOIN ADMISSIONS ADM
ON IE.HADM_ID = ADM.HADM_ID ;
```

IX. Appendix 2 – Pava Application For Loading The Prescription Data To Denormalized Table

```
String sql ="select pre.hadm_id, pre.icustay_id, mpre.row_id, count(pre.drug) as NumOfTimes from
prescriptions pre, m_prescriptions mpre  WHERE  pre.icustay_id is not null and
pre.drug=mpre.drug AND mpre.row_id>3000 AND mpre.row_id<3900 group by pre.hadm_id,
pre.icustay_id, mpre.row_id order by mpre.row_id";
PreparedStatement preStatement = connection.prepareStatement(sql);

ResultSet rs = preStatement.executeQuery();
while (rs.next()) {
    String p1 = rs.getString("HADM_ID");
    String p2 = rs.getString("ICUSTAY_ID");
    String p3 = rs.getString("ROW_ID");
    String p4 = rs.getString("NUMOFTIMES");
    String updatesql ="update M_PATIENT_DENORM "
           + " SET M"+ p3 +" = "+p4
           + " WHERE HADM_ID="+p1+" AND ICUSTAY_ID="+p2
           + "";
    PreparedStatement preUStatement = connection.prepareStatement(updatesql);
    int resultUpdate= preUStatement.executeUpdate();
    preUStatement.close();
    System.out.println(p1+"-"+p2+"-"+p3+"-"+p4 +" updated:"+resultUpdate);
```

I. Appendix 3 – Pava Application For Cleaning The Denormalized Data

```
for (int i = 3001; i < 3900; i++) {
     String updatesql="Update M_PATIENT_DENORM "
     " SET M"+ i +"=0 "
     + " WHERE M"+ i +" IS NULL ";
     PreparedStatement preUStatement = connection.prepareStatement(updatesql);
     int resultUpdate= preUStatement.executeUpdate();
     preUStatement.close();
     System.out.println("row:"+i +" updated:"+resultUpdate);
     }
```

I. Appendix 4 – P eep Learning Python Application Using Keras

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
import numpy
import sys
seed = 8
numpy.random.seed(seed)

print('Starting to load the data')
dataset_data = numpy.loadtxt("export450Data.csv", delimiter=",", dtype=numpy.float32)
dataset_hospitalExpire = numpy.loadtxt("export450HospitalExpire.csv", delimiter=",",
dtype=numpy.float32)
print('Finished to load the data')

Y_train = dataset_hospitalExpire[0:50000]
X_train = dataset_data[0:50000,0:459]

Y_test = dataset_hospitalExpire[50001:60001]
X_test = dataset_data[50001:60001,0:459]

# create model
model = Sequential()
model.add(Dense(400, input_dim=458, init='uniform', activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(100, init='uniform', activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(1, init='uniform', activation=' sigmoid '))
model.add(Dropout(0.5))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(X_train, Y_train, validation_data=(X_test,Y_test), epochs=10, batch_size=10,
verbose=1)
```