Mortality Prediction in ICU
CSE 6250: Final Report

Team 03: Divya Paduvalli, Shubhangi Waghere, Manuel Novoa, and Winnie Philip
Georgia Institute of Technology

Abstract
We have selected Mortality Prediction in ICU as our project topic. In recent years, there has been a concerted move towards the adoption of digital health record systems in hospitals. In the United States, the number of non-federal acute care hospitals with basic digital systems increased from 9.4% to 75.5% over a 7-year period between 2008 to 2014 (1). The percentage increase of the hospital's digital information will be huge in the coming future and to analyze those datasets, we will have to use advanced big data tools. The combination of medical judgment and an algorithmic diagnostic tool based on extensive medical records is, in the best sense, the future of medical diagnosis and treatment. We have used the MIMIC III database for our study (2). In this report, we will be briefly going through why Mortality Prediction in ICU is interesting and the approach we have taken to analyze and solve the given problem.

Motivation
Patients who are typically admitted to an intensive care unit (ICU) suffer from serious emergency illness and are at a high risk of dying. Hence, they need to be constantly treated and monitored to ensure that they recover. In a modern-day ICU, there are several types of devices such as heart monitors, pulse oximeter, ventilators, catheter, arterial lines, etc. that continuously keep track of a patient's health records. If there are any irregular fluctuations, the nurse or the doctor is immediately automatically notified. Aggregating data from these equipments and from several patients will be very useful in predicting the risk of death. In healthcare, estimating the risk of mortality can be very crucial in terms of sorting and allocating hospital resources in determining appropriate levels of care needed. The doctors can also have an early discussion with the patient's family around the expected outcomes. In this study, we will explore the MIMIC III database to find used cases for various machine learning models and techniques to detect diagnosis of patients admitted to ICU with cardiovascular disease. Although the study is conducted on a specific case, the analysis results can be generalized and applied to a broader domain of illnesses.

Literature Survey
Various machine-learning models have been proposed by several researchers to predict mortality. One such research included the use of an ensemble technique called the Super Learner (SL) to improve the performance of mortality prediction. Super Learner uses cross-validation to estimate the performance of various machine-learning models and creates an optimal weighted average. The results of this research had a prediction testing accuracy score of around 95% (3). In another research study, personalized mortality prediction was done by analyzing similar past patients. Death count among similar patients taken first and then logistic regression and a decision tree applied. Using logistic regression, the results of AUROC and AUPRC was 0.830 and 0.474 respectively. Using a decision tree, the results of AUROC and AUPRC was 0.753 and 0.347 respectively (4). We intend on using a very similar approach by clearly categorizing between similar and non-similar patients by using their feature values. Multivariate logistic regression was applied in another research and the results were AUCROC = 0.85, SMR = 1.25 (5). In multivariate logistic regression, multiple independent variables are taken into consideration to predict the dependent variable. Typically, for mortality prediction, severity of illness (SOI) scores are used. SOI refers to the physiological decompensation (organ system failures) of a patient (11). SOI scores tend to underperform on the validation set. In another research study, a neural network was utilized to predict mortality. Neural network resulted in a superior performance with an AUCROC score of 0.8445 for 30 days mortality and 0.86 for hospital mortality (6). Some studies have evaluated the benefit of nonparametric approaches, namely based on neural networks or data mining, to predict hospital mortality in ICU patients (10). Technologically speaking, speed, high availability, stability, and scale are required characteristics of the technology stack to be used, given the fact that software for health care systems should inherent these out of the box. One of the most promising big data analytics engine for data science professionals that have emerged with those qualities is Spark. The true power and value of Apache Spark lies in its ability to execute data science tasks with speed and accuracy. Spark's selling point is that it combines ETL, batch analytics, real-time stream analysis, machine learning, graph processing, and visualizations (7). On the other hand and given strong support from the scientific community, Python became the first choice for the programming language. The language represents the perfect bridge between the selected data processing tool and the ML framework selected. Libraries like NumPy, Pandas, pyspark, with the proper tooling for the development, made the process faster and more intuitive (8).

Our Approach
Most models stated in the literature survey were designed for at least 24 hours or 48 hours after ICU admission to provide real-time or retrospective prediction on patient mortality. In this study, we propose a two-phase model framework to address the task of predicting the mortality and also death hours in the early stage of ICU stay timeframe. If a patient is predicted dead in the first phase, our models would further provide an estimate of death hours since ICU admission in the second phase. Our goal is to identify high-risk patients who might be dead within hours or days since ICU admission. Data was pre-processed and extracted for the 24 hour period since ICU admission for each ICU stay from MIMIC-III database. Multiple models were trained on the extracted features of the study population for the specified timeframe. The model results are compared and discussed thoroughly in this report.

Data Structure
The original dataset in the MIMIC-III database consists of 61,532 distinct ICU stays of 46,520 unique patients. To form our study population, we have included all ICU stays to analyze the data of unusual short stays and only consider all patients. The final study population covered 49,632 ICU stays of 36,343 patients. Multiple machine learning models in this study have been trained and evaluated based on this study population. Table 1 provides summary statistics of the study population.

Variables	Statistics
Age	Mean 62.61
Gender	Male 57.79%
Ethnicity	White 71%
Admission type	Emergency 82.31%
Number of ICU stays	Mean 1.37
In-hospital mortality ratio	11.62%

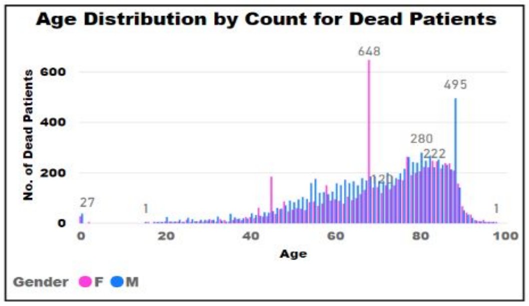
Table 1: Summary statistics of the study population

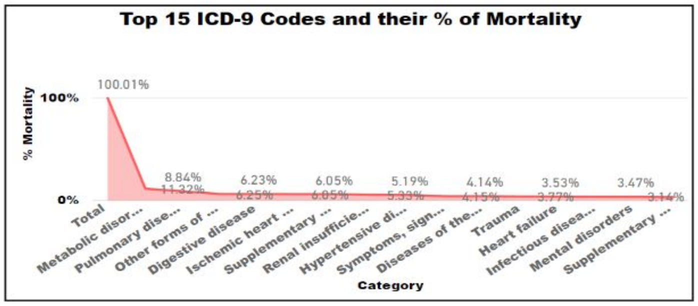
Among 49,633 ICU stays, there are 5,766 in-hospital mortality. After filtering out the ICU stays with negative death time since ICU admission (which is likely an administrative error resulting in an incorrect ICU admission or incorrect death time), 5,718 in-hospital mortality were resulted. The average death time since ICU admission is 9.57 days, maximum death time is 206.38 days and minimum death time is 0 day. Below line and clustered chart (Age Distribution by Count for Dead Patients) shows perfect distribution of deaths by age and as per the trends in the graph, most of the deaths happen between ages 55-85. Age groups 20-40 has less death rate but still the number is in the range of 100-200.

Category	Code Range	Number of Codes	Death %
Metabolic disorder	240-279	23927	11.3%
Pulmonary disease	460-519	18694	8.8%
Other forms of heart disease	420-429	13203	6.3%
Digestive disease	520-579	13181	6.2%
Supplementary classification of factors influencing health status and contact with health services	V01-V86	12801	6.1%
Ischemic heart disease	410-414	12794	6.1%
Renal insufficiency	580-629	11262	5.3%
Hypertensive disease	401-405	10970	5.2%
Symptoms, signs, and ill-defined conditions	780-799	8777	4.2%
Diseases of the blood and blood forming organs	280-289	8744	4.1%
Trauma	800-959	7975	3.8%
Heart failure	428	7470	3.5%
Infectious diseases	001-139	7388	3.5%
Mental disorders	290-319	7337	3.5%
Supplementary classification of external causes of injury and poisoning	E800-E999	6632	3.1%
Arteries and veins	440-459	5828	2.8%
Neoplasms	140-239	5403	2.6%
Neurologic disease	320-389	5140	2.4%
Diseases of the musculoskeletal system & connective tissue	710-739	3632	1.7%
Other complications of procedures, NEC	998	2936	1.4%
Cerebrovascular disease	430-438	2849	1.4%
Diseases of the skin and subcutaneous tissue	680-709	2799	1.3%
Complications affecting specified body systems, not elsewhere classified	997	2551	1.2%
Complications peculiar to certain specified procedures	996	2403	1.1%
Other and unspecified effects of external causes	990-995	1941	0.9%
Chronic rheumatic heart disease	393-398	1438	0.7%
Diseases of pulmonary circulation	415-417	1233	0.6%
Congenital anomalies	740-759	673	0.3%
Complications of pregnancy, childbirth, and the puerperium	630-677	627	0.3%
Poisoning	960-989	590	0.3%
Complications of medical care, not elsewhere classified	999	213	0.1%
Acute Rheumatic fever	390-392	5	0.0%

Table 2. ICD9 codes and Death percentage

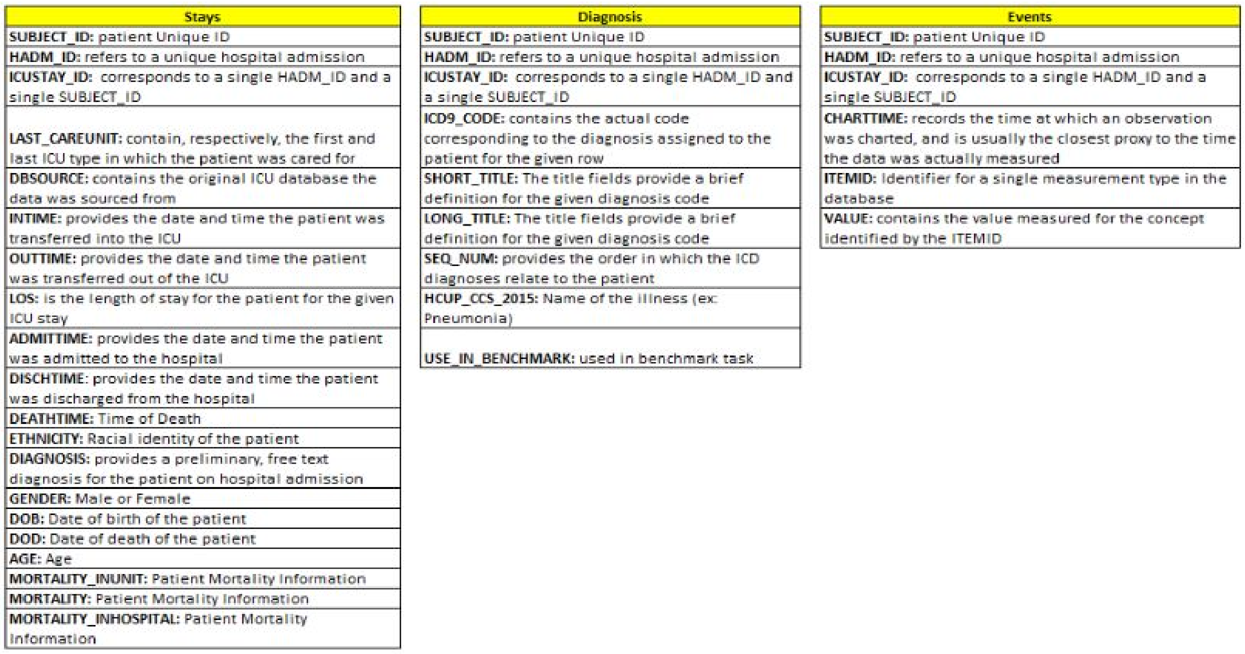
ICD9_CODE is one of the major feature for finding the mortality of the patients and it contains the actual code corresponding to the diagnosis assigned to the patient. Hence, we did more analysis of those codes and extracted the data from MIMIC III database on death percentage among the various ICD-9 codes. We extracted only dead patients from and their related ICD-9 codes and calculated the percentage death among them. The above graph describes top 15 ICD-9 codes by percentage of death for more visibility. After closely observing the graph we can clearly tell that metabolic disorder, pulmonary disease, heart disease, and digestive diseases are the common diseases for mortality resulting into death. Table 2 below describes the overall ICD-9 codes resulting into death.





Data Pre-Processing

We have a total of 26 datasets downloaded from the MIMIC III database (2). These datasets have a mixture of both categorical and continuous variables. Since the total size of the datasets is around 43.3 GB with over a million records, we used hadoop spark to automate the process of data extraction. Our algorithm was inspired by the homework assignments and the benchmarking techniques used in the Multitask Learning and Benchmarking with Clinical Time Series Data research paper (9). Despite using spark, the overall data extraction process was computationally very expensive. Each patient in the database are referred to as subjects and are given a unique subject ID. We created folders for each subject ID and in each folder, we stored specific patient information that will aid in predicting the mortality. Subject's hospital stays, diagnosis, and events information was stored in each of those folders. The below figure 1 shows the attributes from each of the csv files used and its descriptions. The attributes in the below figure are extracted from several datasets. Please click on this excel link to learn about from which specific datasets the below attributes are derived from. If you notice, the attributes are a mixture of demographic information (such as age, date of birth, etc.) and technical information (such as mortality, the name of the illness, etc.) about the illness that the patient was suffering from. Eventually, this will aid in our analysis to answer questions like, is there a specific risk for people belonging to a certain age or ethnicity to suffer from a particular illness? Does frequent visit to the hospitals or longer hospital stays indicates that the patient's health is deteriorating and may increase the chance of mortality?.



Stays
SUBJECT_ID: patient Unique ID
HADM_ID: refers to a unique hospital admission
ICUSTAY_ID: corresponds to a single HADM_ID and a single SUBJECT_ID
LAST_CAREUNIT: contain, respectively, the first and last ICU type in which the patient was cared for
DBSOURCE: contains the original ICU database the data was sourced from
INTIME: provides the date and time the patient was transferred into the ICU
OUTTIME: provides the date and time the patient was transferred out of the ICU
LOS: is the length of stay for the patient for the given ICU stay
ADMITTIME: provides the date and time the patient was admitted to the hospital
DISCHTIME: provides the date and time the patient was discharged from the hospital
DEATHTIME: Time of Death
ETHNICITY: Racial identity of the patient
DIAGNOSIS: provides a preliminary, free text diagnosis for the patient on hospital admission
GENDER: Male or Female
DOB: Date of birth of the patient
DOD: Date of death of the patient
AGE: Age
MORTALITY_INUNIT: Patient Mortality Information
MORTALITY: Patient Mortality Information
MORTALITY_INHOSPITAL: Patient Mortality Information

Diagnosis
SUBJECT_ID: patient Unique ID
HADM_ID: refers to a unique hospital admission
ICUSTAY_ID: corresponds to a single HADM_ID and a single SUBJECT_ID
ICD9_CODE: contains the actual code corresponding to the diagnosis assigned to the patient for the given row
SHORT_TITLE: The title fields provide a brief definition for the given diagnosis code
LONG_TITLE: The title fields provide a brief definition for the given diagnosis code
SEQ_NUM: provides the order in which the ICD diagnoses relate to the patient
HCUP_CCS_2015: Name of the illness (ex: Pneumonia)
USE_IN_BENCHMARK: used in benchmark task

Events
SUBJECT_ID: patient Unique ID
HADM_ID: refers to a unique hospital admission
ICUSTAY_ID: corresponds to a single HADM_ID and a single SUBJECT_ID
CHARTTIME: records the time at which an observation was charted, and is usually the closest proxy to the time the data was actually measured
ITEMID: Identifier for a single measurement type in the database
VALUE: contains the value measured for the concept identified by the ITEMID

Figure 1

The number of events prior to removing missing information was 253,116,833. Events with missing HADM_ID (5,162,703) and ICUSTAY_ID (15,735,688) information in the three csv files were removed and the total number of events after this step turned out to be 232,218,442. Every time a subject is admitted to the ICU, their ICU stay is represented with an ICU stay ID. A subject may have multiple ICU stay IDs which indicates that they have been admitted to the ICU multiple times. ICU stay medical information such as, capillary refill rate, diastolic blood pressure, fraction inspired oxygen, glascow coma scale eye opening, glascow coma scale motor response, glascow coma scale total, glascow coma scale verbal response, glucose heart rate, height, mean blood pressure, oxygen saturation, respiratory rate, systolic blood pressure, temperature, weight, and pH, was extracted for every ICU stay ID in different csv files (based on the number of ICU visits) for every subject ID.

All the subject level ICU stay information csv files were distributed to three folders namely, Train (60% ~14,681), Validation (20% ~ 3,222), and Test (20% ~ 3,236). In this process, three separate csv files were generated namely, train_listfile, val_listfile, and test_listfile. These 3 csv files contains the names of all the ICU stay csv files in the three folders and their corresponding mortality information represented by 0 (dead) and 1 (alive). We have ran our machine learning models on the full data set.

Model Architecture

In phase 1 and phase 2, we trained and experimented with feature rich MIMIC III data sets using a combination of classification and deep learning keras models namely: Logistic Regression, Random Forest, Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Long Short-Term Memory Network (LSTM). We finalized two keras deep learning frameworks in Python for our machine learning model evaluation in Phase 2. Keras is an open source machine learning API framework that can run on top of Theano, Tensorflow, or CNTK framework. Karas APIs are simple to use and to understand the machine learning problem and in addition, we get the full ability of its underlying powerful deep learning architecture. We chose to run Keras model on top of Tensorflow. The figure 2.b shows the popularity of Keras API. Keras Model has built in multi - GPU Parallelism support. We noticed that LSTM model training took 30 hours to complete in CPU. For the ANN/CNN Keras models, we tried 'binary_crossentropy' and 'mean_squired_error' as two loss functions, and 'adam', and sgd as two optimizers. We noticed that 'sgd' optimizer, and 'mean_squired_error' loss function yielded expected AUC/ROC, F1 score, and Accuracy, and we finalized our model based on this finding. We trained our models using train data size of 14681* 714, validated using 3222*714, and tested using test set of size 3236*714. The training dataset was of reasonable size to run on a local laptop MAC OS environment, therefore we chose to train model in Spyder/Anaconda Python 3.6 environment, and capture the results log in HTML form for further review. Figure 2.b shows deep learning frameworks ranking computed by Jeff Hale, based on 11 data sources across 7 categories.

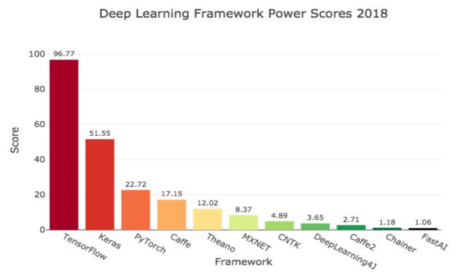


Figure 2.b Reference

System Specifications

Our study consists two stages of implementation: (i) feature engineering using Hadoop HDFS, and PySpark on Local Docker MAC OS environment (1 Master node and 2 worker nodes, each with 900 GB space, 14 GB RAM, and 4 processors), and (ii) Machine learning using Python 3.6 on a local cluster (500 GB space, 16GB RAM, 4 GB CPU and 4 processors). The decompressed dataset of MIMIC-III requires around 50 GB of space. We chose big data tools namely: Apache Hadoop/HDFS and PySpark to perform data preprocessing and feature engineering since the dataset is quite large. Data output in the first stage is then used as feature input for the model training in the second stage. We also used Python and packages such as Keras, Pandas and Scikit-learn for efficient model testing, hyperparameter tuning and model evaluation. All the dependencies for the execution of this project has been defined in the conda environment definition added in the root of the project (environment.yml). A set of reproducible code, detailed instructions of implementation (see README) and presentation slides have been provided along with the paper (also available at our **github link**).

Experimental Results

ANN Loss, Accuracy, and ROC Curve:

ANN model is trained and validated on the dataset for 10 epochs to give a quick view of the model's working. Figure 3 ANN LC states the training loss and validation loss which starts from ~0.13 The robustness of model increases along with the decrease of the value of loss function. Training Loss gradually decreases and stabilizes at ~0.04. Validation Loss also reduces with training loss. Loss curve states that the model is neither overfitting nor underfitting.
Accuracy Curves helps in notifying the score accuracy of prediction. Figure 4 of ANN accuracy denotes that the validation accuracy is between 0.882-0.884 i.e. model can predict even for the new unseen data with most accuracy. Figure 5 ANN ROC gives a smooth AUC curve which covers an area of about 0.85. It gives the notion of a positive learning model.

ANN Confusion Matrix:

True positive gives 0.95 and false positive is 0.05 for the confusion matrix for ANN which explains that the training is appropriate for predicting positive data. Even if false negative is greater than the true negative, the difference in prediction is much lower. F1-score was 0.49 which considers not even positive but also negative prediction values.

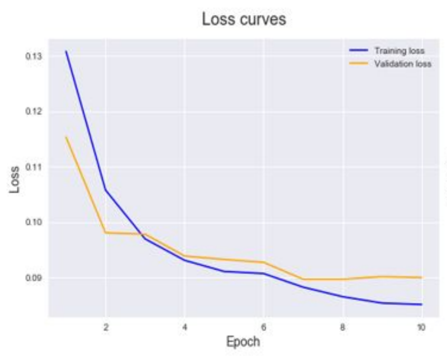


Figure 3: ANN LC

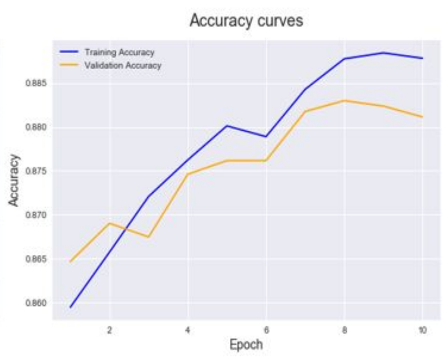


Figure 4: ANN AC

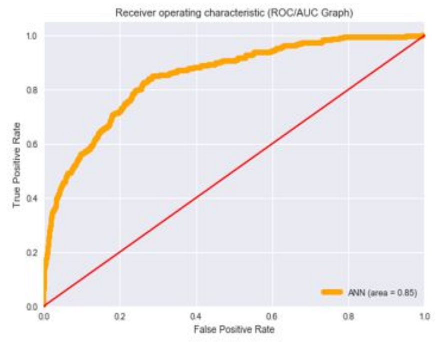


Figure 5: ANN ROC

CNN Loss, Accuracy, and ROC Curve:
Figure 6 CNN LC plots the Training Loss and Validation Loss. Model trains quickly with the training data as the training loss touches 0.08 before 40 epochs and goes on decreasing further. Validation loss goes down towards 0.09 and below. This gives a very fine-tuned binary classifier for mortality prediction. Training accuracy in figure 7 CNN AC reaches 0.895 before 40 epochs. Validation accuracy continuously increases and attains 0.885 which states the correct positive prediction. Area under the AUC/ROC Curve determines the working of the model. CNN model built with specific configurations which gives the area of 0.85 and curve almost touching the top-left corner of the graph.

CNN Confusion Matrix:
True positive scores 0.96 and false positive 0.04 for the confusion matrix in Fig CNN CM. It means the training works good on positive data. False Negative is greater than True Negative which explains that still the Negative prediction is not enough good and needs further tuning.

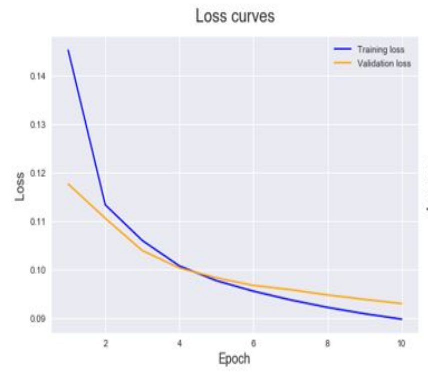


Figure 6: CNN LC

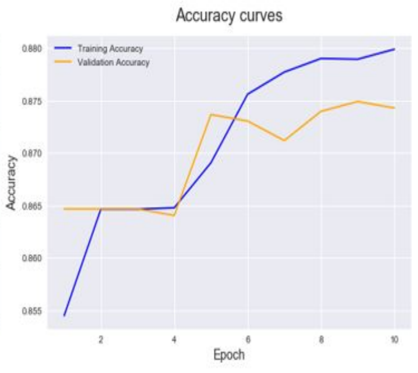


Figure 7: CNN AC

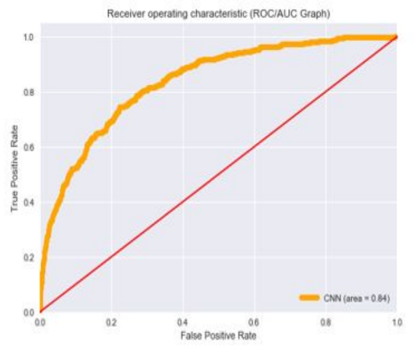


Figure 8: CNN ROC

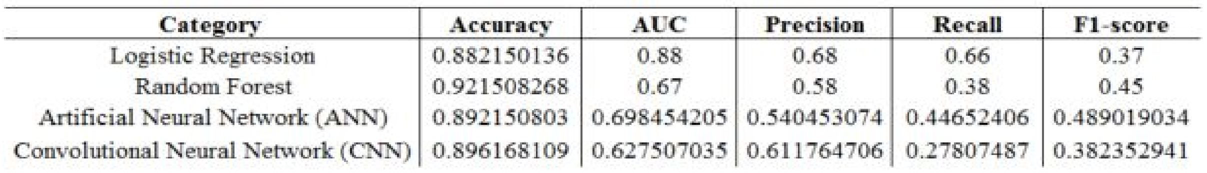
Random Forest Curve:
We produced AUC/ROC curve, learning curve, confusion matrix, accuracy, and performance metrics tables, and precision recall curves. It was interesting to notice that in python, producing these visualization was computationally time consuming when we tuned different parameters particularly for this model due its increased complexity as compared to logistic regression. Model gives the varying differences in metrics while training and testing the random forest model with specific parameters. In model accuracy results table, accuracies are almost 90%, which shows even for fewer parameters this model works great. An another reason for such a high accuracy score is due to the model's inherent ability to not overfit. AUC/ROC curve for the sample entry from the above table where the area is calculated as 0.60. Every test done gives a Confusion Matrix like the one in figure 4.3 which helps in calculating precision and recall metrics. For this Confusion Matrix, we have precision as 0.9846 and recall as 0.9099. In Performance Metrics result table, for varying parameters the Precision range was 0.55-0.61 while recall ranges from 0.34-0.38. Precision-Recall curve shows a good steady descent where AUC=0.494 and F1=0.344. Training score curve gradually decreases but the cross-validation score is almost steady which ensures the model is highly biased. It needs increasing the complexity of the model by engineering new and more relevant features to deliver the greatest impact and we will work on those enhancements in the next phase.

Team's Reflection on Experimental Results

We have shown that by combining static features such as patient demographic information and dynamic features such as physiological variables measured in ICU, we could train an effective model to predict in-hospital mortality in the early stage of ICU stay. The predictive performance of the model trained ICU data has competitive performance (AUROC 0.88) with the same model trained on 12-hour or 24-hour ICU data (AUROC 0.90 and 0.92 respectively). We then showed that the death time multiclass classifier trained different ICU stay data offers an effective base (micro-average AUROC 0.77) to provide a rough estimate of death hours since ICU admission. The result is competitive to the same model trained on 12-hour or 24-hour ICU data (micro-average AUROC 0.79 and 0.82 respectively). A natural question is that can we use regression instead of classifier to more precisely predict death time in Phase 2. In our proof-of-concept stage, we have fitted a random forest regressor using the same set for extracted features. However, the regressor overfitted and only achieved R = 0.20 on the test set. R is the coefficient of determination, which is a regression metric to measure how good the model explains the data. Best possible R is 1. A constant model that always predicts the expected value of target would get a R of 0. Due to the poor predictive performance, the trained regressor is not very useful in the ICU setting. Hence, it is better to formulate the death time prediction problem as a multiclass classification problem instead of a regression problem. Apart from this, our model framework provides a base for potential improvement. One possible way to enhance the performance is to fit the model with time-series data instead of aggregation of dynamic features. In our study, we have aggregated dynamic features over a specified timeframe using mean, maximum, minimum or sum. In fact, some features such as heart rate are highly temporal. Methods such as CNN and ANN, are types of Recurrent Neural Network, which is designed to handle sequence dependencies for time-series prediction problem, could be applied to handle temporal data. In our proof-of-concept stage, we have fitted ANNs and CNNs to dynamic features such as heart rate, blood pressure and respiratory rate, and ensembled them with random forest classifiers trained on other features. The initial result is in fact quite satisfactory. Fitting ANNs and random forest classifier without any tuning achieved AUC score of 0.8. It is possible that we could improve the model performance if we could find an effective algorithm to interpolate missing time-series data, construct a deeper CNN and tune the model. However, due to time and resource constraint, we have focused our study on constructing a single random forest classifier in phase 1 and ANN and CNN models in phase 2.

Model Comparison

Now, Comparative study is done referring the table 3 below. Accuracy is the ratio of number of correct predictions to the total number of input samples. Here, Random Forest gives the highest accuracy(0.92) compared to the others. ANN and CNN accuracy is almost the same. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. AUC is the area under the curve of plot of True Positive Rate(Sensitivity) against False Positive Rate(Specificity). Logistic Regression gives the highest AUC as 0.88, confirming it works well for a Positive result of a binary classifier. F1 Score is used to measure a test's accuracy. The greater the F1 Score, the better is the performance of our model. Precision corresponds to the number of correct positive results divided by the number of positive results predicted by the classifier. While Recall is the number of correct positive results divided by the number of all relevant(actual positive) samples. Only F1 score considers both precision and recall which helps in understanding how precise your classifier is (how many instances it classifies correctly), as well as how robust it is(it does not miss a significant number of instances). Comparing F1 Score of all the experimented ML models, it turns out that ANN has the highest score i.e. 0.489. Even if RF has 0.45 F1 Score, it classifies highly for one category than other. Hence, ANN model is the most robust and more accurate classifier for mortality prediction on given dataset. Apart from fine-tuning the model, more than one classifier can be adjoined for this mortality prediction problem to further increase the accuracy.



Category	Accuracy	AUC	Precision	Recall	F1-score
Logistic Regression	0.882150136	0.88	0.68	0.66	0.37
Random Forest	0.921508268	0.67	0.58	0.38	0.45
Artificial Neural Network (ANN)	0.892150803	0.698454205	0.540453074	0.44652406	0.489019034
Convolutional Neural Network (CNN)	0.896168109	0.627507035	0.611764706	0.27807487	0.382352941

Table 3: Model Performance Metrics

Challenges faced during Project Implementation

We spent a lot of time in reading numerous Literature Survey papers to understand the previous related work done by other researchers and in analyzing MIMIC III dataset to understand the tables and its underlying data structures. We did research on the Azure HDInsight as well as AWS EMR Big Data cloud services for identifying a cost effective way of loading, pre-processing and transforming our raw data set into valid training,validation and test set. We finally decided to use our local Hadoop Docker machine instance to run pre-processing job in PySpark, it took 48 hours to complete a run, that we could've avoided if we utilized GT GPU cluster instances. Another challenge that we faced is in tuning model hyper parameters. We spent good amount of time in fine tuning hyper parameter on LR/RF/CNN/ANN models, and analysing ROC/Loss curves. We didn't get enough time to fix Keras LSTM model issues. We used Keras deep learning models for the first time. This project was a great learning experience for our team to see a full life cycle of a Machine Learning problem implementation.

Conclusion

As a summary,the work for the entire project lifecycle was divided into two phases namely, phase 1 and phase 2. This final report is dedicated mainly for phase 2.The entire large dataset was processed using big data tools such as hadoop, spark, sql and cohorts based mainly on ICU related technical information for mortality prediction are created. This was the most time-consuming portion of phase 2. To understand the structure and the behavior of the datasets, three advanced neural network models namely: Artificial Neural Network, Convolutional Neural Network, and Recurrent Neural Network (LSTM) are used. Inspired by the homework assignments, an environment.yml file is created with a list of all python libraries and its specific version numbers and the experiments in Docker and Spyder IDE (the results were stored in our private group GitHub) were conducted. Two weekly checkpoint meetings (on every Monday and Thursday)were scheduled, which were dedicated mainly for strategic planning and code implementation working sessions.

Logistic Regression(LR) is a mathematical model used in statistics to estimate the probability of an event occurring having been given some previous data.Random Forest(RF) formulates some set of rules that can be used to perform predictions. LR gives the function value which classifies the data while RF branches to the specific decisive sub-trees which categorizes the input for multi-class classification. Neural Networks(ANN) are based on the working of a Human- brain. It learns from the input with self-adjusting weights and stores knowledge in the form of frozen weights. Convolutional Neural Network(CNN) is an Neural Network with the refinement of Deep Learning. Deep Learning is achieved by performing Optical Character Recognition(OCR), pooling and down-sampling. ANN and CNN both learns from the train data by storing weights and then classifying the input based on the tuned model. Only difference in CNN is that it processes more efficiently on even image and sound(represented as spectrogram)inputs. NN solves all types of ML problems i.e. clustering, classification and regression.

Timeline

All the team member have contributed equally throughout the project life cycle.

Week	**Tasks**	**Division of Labor/Time**
3/3/2019	Proposal	All (around 8 hours)
3/10/2019	Data Pre Processing	All (around 8 hours)
3/24/2019	Code Development Phase	All (around 8 hours)
4/7/2019	Project Draft	All (around 8 hours)
4/14/2019	Testing Phase	All (around 8 hours)
4/28/2019	Final Project (code+presentation+final paper)	All (around 8 hours)

References

1. http://www.ncmedicaljournal.com/content/77/2/112.full
2. https://mimic.physionet.org/
3. Pirracchio.Romain, Petersen. Maya L. “Mortality prediction in the ICU: can we do better? Results from the Super ICU Learner Algorithm (SICULA) project, a population-based study” NCBI, 24 Nov. 2014, ncbi.nlm.nih.gov/pmc/articles/PMC4321691/
4. Lee, Joon, et al. “Personalized Mortality Prediction Driven by Electronic Medical Data and a Patient Similarity Metric.” PLOS ONE, Public Library of Science, 2015, journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0127428.
5. Fika, Sofia, et al. “A Novel Mortality Prediction Model for the Current Population in an Adult Intensive Care Unit.” Heart and Lung, Jan. 2018, www.heartandlung.org/article/S0147-9563(17)30151-6/fulltext.
6. Zahid, M.A. “Mortality Prediction with Self Normalizing Neural Networks in Intensive Care Unit Patients.” IEE Xplore, Wiley-IEEE Press, 4 Mar. 2018, ieeexplore.ieee.org/abstract/document/8333410.
7. Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask Learning and Benchmarking with Clinical Time Series Data. arXiv:1703.07771
8. McKinney, Wes. Python for Data Analysis, 2nd Edition. O'Reilly Media, Inc. 2017
9. Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask Learning and Benchmarking with Clinical Time Series Data. arXiv:1703.07771
10. Dybowski R, Weller P, Chang R, Gant V (1996) Prediction of outcome in critically ill patients using artificial neural network synthesized by genetic algorithm. Lancet 347(9009):1146–1150. https://linkinghub.elsevier.com/retrieve/pii/S0140673696906091
11. M. Ghassemi, T. Naumann, F. Doshi-Velez, N. Brimmer, R. Joshi, A. Rumshisky, and P. Szolovits. Unfolding Physiological State: Mortality Modelling in Intensive Care Units. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 14, pages 7584, New York, NY, USA, 2014. ACM.
12. https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a