Using Natural Language Processing to Predict Mortality in ICU Patients
Capstone 2 Milestone Report

Background

In-hospital mortality has long been used as a measure of hospital quality. As the world copes with the Covid-19 pandemic, hospital mortality rates - particularly those involving ICU stays, have come to the forefront.The intensive care unit (ICU) has the highest mortality rate of all the units in a hospital. Of the approximately four million ICU admissions per year in the United States, the average mortality rate ranges from 8-19%, or about 500,000 deaths annually[1]. The ICU mortality rate of Covid-19 was found to be much higher, at 41.6%, across international studies as of July 2020[2]. Fortunately, the mortality rate of Covid-19 appears to be trending downward in the last few months[11]. However, as the pandemic begins its predicted resurgence this fall and winter, decisions regarding the proper rationing of beds and mechanical vents will become issues that many ICUs throughout the US will, more likely than not, be dealing with again as they did earlier in the pandemic.

High ICU death rates lead to unnecessarily high financial costs and poor patient experiences at the end of life. The average cost of an end-of-life ICU stay in the US is $39,300[3]. This amounts to $19.65 Billion per year. Costs of prompt palliative care can reduce hospital costs by $3237[4] per patient according to a study from the Icahn School of Medicine at Mount Sinai and Trinity College Dublin. If hospice care is chosen as the end of life care option (in place of in-hospital palliative care), the average cost per patient is even less, at $11,820[5], reducing average costs by $27,480 per patient. If patients nearing the end of life could be identified soon after hospital admission, they could be given prompt palliative care or discharged to their home or other facility for hospice, resulting in a substantially reduced financial burden and a better patient experience by removing unnecessary, burdensome and unwanted procedures.

One possible way to help patient care providers and hospitals predict patients who are at the end of life is to analyze the data present in caregiver notes. Natural language processing (NLP) is a field of artificial intelligence that allows machines to understand the human language. Using NLP, computers can process words and phrases from text, like *sepsis* and *status worsening,* into a form that can be included in predictive models. The objective of this project is to use NLP to predict in-hospital mortality for ICU patients.

Client

The client is a healthcare system looking to reduce in-hospital mortality rates in order to improve patient experiences, outcomes and quality of care metrics.

Data

This project was completed using the MIMIC-III database from MIT. MIMIC-III is a large, freely-available database comprising de-identified health-related data associated with over 50,000 patients who stayed in ICUs of the Beth Israel Deaconess Medical Center between 2001 and 2012. The database includes information such as caregiver notes, demographics, vital sign measurements made at the bedside, laboratory test results, procedures, medications, imaging reports, and mortality. The database is accessible (after completing an online credentialing process) via: https://mimic.physionet.org/.

Data was obtained from three datasets in MIMIC-III: ADMISSIONS, NOTEEVENTS, and ICUSTAYS. ADMISSIONS contains the target variable, HOSPITAL_EXPIRE_FLAG (i.e., whether or not the patient died during hospitalization) and unique identifiers HADM_ID (hospital admission ID) and SUBJECT_ID. NOTEEVENTS contains TEXT (caregiver notes) and the unique identifiers HADM_ID and SUBJECT_ID. ICUSTAYS included LOS (length of stay in the ICU). The following is a full list of database variables included in the analysis:

ADMISSIONS:
- HOSPITAL_EXPIRE_FLAG
- SUBJECT_ID
- HADM_ID
- ADMITTIME
- DEATHTIME
- ADMISSION_TYPE

NOTEEVENTS:
- SUBJECT_ID
- HADM_ID
- TEXT
- CHARTDATE
- CATEGORY

ICUSTAYS:
- HADM_ID
- INTIME
- OUTTIME
- LOS

Target Variable

HOSPITAL_EXPIRE_FLAG was selected as the outcome variable (1=death, 0=no death).

Data Preparation

Preparation of the data began with data exploration and cleaning. Python pandas, numpy, and matplotlib were used for data wrangling and exploration. Natural Language Toolkit (NLTK) was used for the Bag-of-Words model of converting text into numbers that can be processed for prediction

Reading in the Data

Each dataset was provided in a GZIP compressed CSV format. The datasets were imported directly into individual pandas dataframes. Each dataset was explored and modified as needed before merging with other datasets.

Data Exploration

Initial data exploration of the ICUSTAYS, ADMISSIONS and NOTEEVENTS tables found, respectively, 61,532 rows/12 columns, 58,976 rows/19 columns and 2,083,180 rows/ 11 columns.

Feature Engineering and Data Cleaning

- All of the features started out as 'object' and 'int' data types. Date and time variables were converted to pandas datetime and timedelta format.
- Within ADMISSIONS, the column DEATHTIME contained many null values, which were retained since this reflected the patients who did not expire during hospitalization. No other null values were noted.
- New columns were created:
    - ICU: a boolean column that identified patients who stayed in the ICU one or more times during a each hospitalization
    - ENDTIME: a datetime column that, based on ADMITTIME, determined when to complete note collection for analysis (i.e., notes taken within 24 hours of admission).
    - LOS_HOSP: an integer column that indicated length of stay of hospitalization
- A new dataframe was created:
    - 'grouped_text': a dataframe that aggregated caregiver notes for each individual admission into a single row in the column TEXT.
- Irrelevant data was removed, including:
    - Patients who had no ICU stays (ICU==False)

- ○ Patients with a length of stay less than 24 hours (LOS_HOSP<24 hours)
- ○ Notes tagged as documented in error (ISERROR==1)
- ○ Patients with DEATHTIME within 24 hours of admission (DEATHTIME< ENDTIME)
- ○ Duplicate rows

Merging of Datasets

ADMISSIONS was updated to include only patients with ICU stays by performing a left merge with ICUSTAYS on HADM_ID. The resulting dataframe was then left-merged with NOTEEVENTS. This full dataset, which included 1,830,890 rows and 12 columns, was reduced to a final dataset of 53,976 rows and 10 columns once the caregiver notes were aggregated (as mentioned above in *Feature Engineering and Data Cleaning*).

The final dataset was now ready for processing and saved as a CSV file.

Splitting the Data into Training, Validation and Test Sets

In order to apply the NLP model, and for eventual use in machine learning, the data was split into training, validation, and test sets.

Text Preprocessing

To allow the machine learning model to accept the data, caregiver text from the training set was preprocessed using the NLP Bag-of-Words (BoW) model. Bag-of-Words is a simple way to represent text with numbers. Basically, it breaks up a text (here, each caregiver note) into individual words (or 'tokens') and then counts how often each word occurs. Each note is represented as a bag of words vector (i.e., a string of numbers).

BoW was implemented as follows:

Step 1: To free up valuable processing time, numbers, punctuation, and irrelevant characters, such as \n, were removed from the text and a pre-made list of stop words from the NLTK library was used to remove irrelevant words (such as "a", "the" "and"). The most frequent words were noted, and a few stop words were added to the list (Figures 1 and 2).

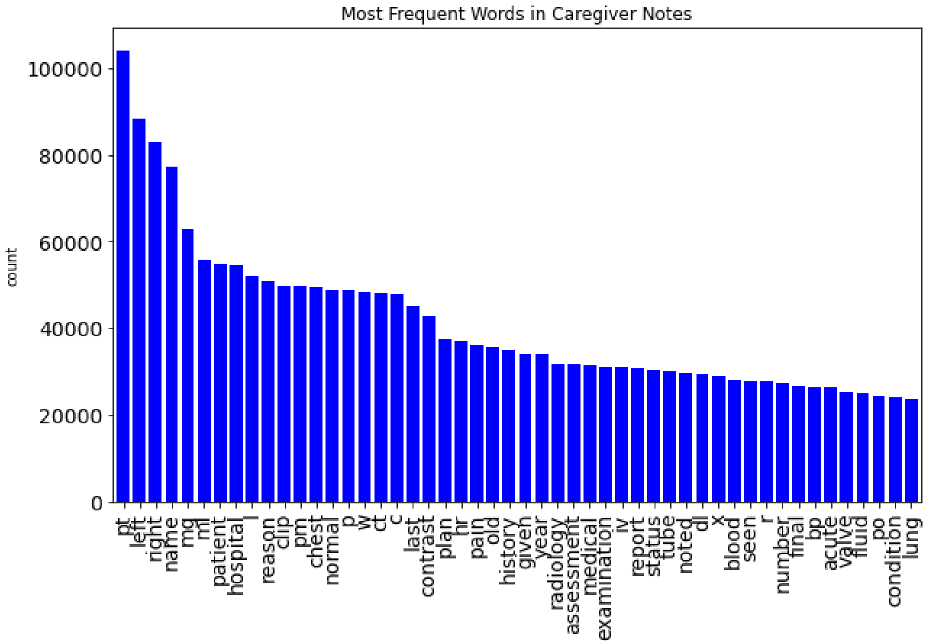

Figure 1. Most common words in caregiver notes.

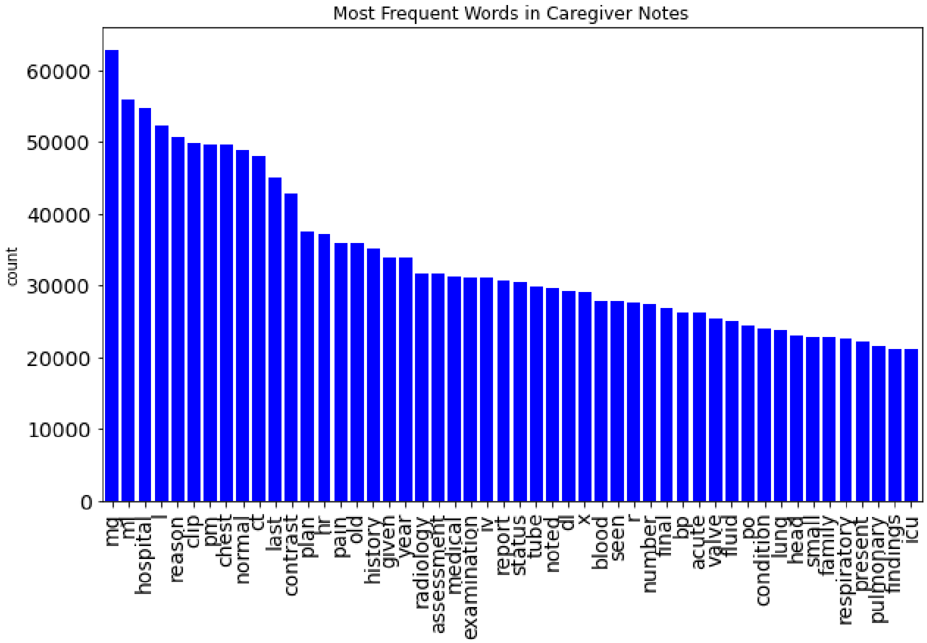

Figure 2. Most common words in caregiver notes (after additional clinical stop words filtered)

Step 2: A vocabulary of 'tokens' was built from all the unique words that were left in the caregiver notes.

Step 3: Finally, each of the tokens was fed into a vectorizer, where it was converted into a feature (variable) that represented the different columns of the dataset. One column (vector), was created for each unique token in the corpus (dataset). For a given text sample, each time a word occurred, the value of 1 was added to its column, if not, the value added would be 0.

Now the text was ready for use in a machine learning model.

Prediction Using Machine Learning

Model Selection

Since the target variable, mortality, is categorical and binary, a supervised learning model that would work well with a large, sparse matrix dataset was needed. Simple logistic regression and Naive Bayes models were applied to the data and performance metrics were compared between the two models on both the training and validation data (Figures 3 and 4).

The following metrics were examined:

- *Accuracy*: Provides an overview of each model's performance. Out of all patients in the database, the fraction of patients who were correctly predicted as dying or not dying.
- *Recall/Sensitivity:* The proportion of all patients deaths that were correctly predicted to be deaths.
- *Specificity*: The proportion of all survived, that were correctly predicted to survive.
- *Precision*: The proportion of all predicted deaths that were *correctly predicted* to be deaths.

In all the indicators examined, Logistic Regression, Random Forest, and SVM performed better than Naive Bayes. While Random Forest scored highest in nearly all metrics for the training data, this was due to over-fitting, so this model was eliminated from the running. Logistic Regression had the next greatest performance, with an AUC-ROC of 0.978 for the training set and 0.781 for the validation set. It also scored best in Specificity, which is important for this data, since it would be desirable to find a model that is less likely to predict false positives (i.e., incorrectly predict death).

Performance Metrics				
	Logistic Regression	Naive Bayes	Random Forest	SVM
AUC	Train:0.978 Valid:0.781	Train: 0.743 Valid: 0.699	Train: 1.000 Valid: 0.814	Train:0.955 Valid: 0.814
Accuracy	Train:0.917 Valid:0.717	Train: 0.594 Valid: 0.830	Train: 1.000 Valid: 0.661	Train: 0.876 Valid: 0.671
Recall	Train:0.917 Valid:0.718	Train: 0.283 Valid: 0.248	Train: 1.00 Valid: 0.804	Train: 0.859 Valid: 0.788
Precision	Train:0.917 Valid:0.187	Train: 0.749 Valid: 0.161	Train: 1.000 Valid: 0.180	Train: 0.895 Valid: 0.177
Specificity	Train:0.917 Valid:0.717	Train: 0.905 Valid: 0.883	Train: 1.000 Valid: 0.648	Train: 0.892 Valid: 0.660

Figure 3. Performance metrics - Logistic Regression and Naive Bayes.

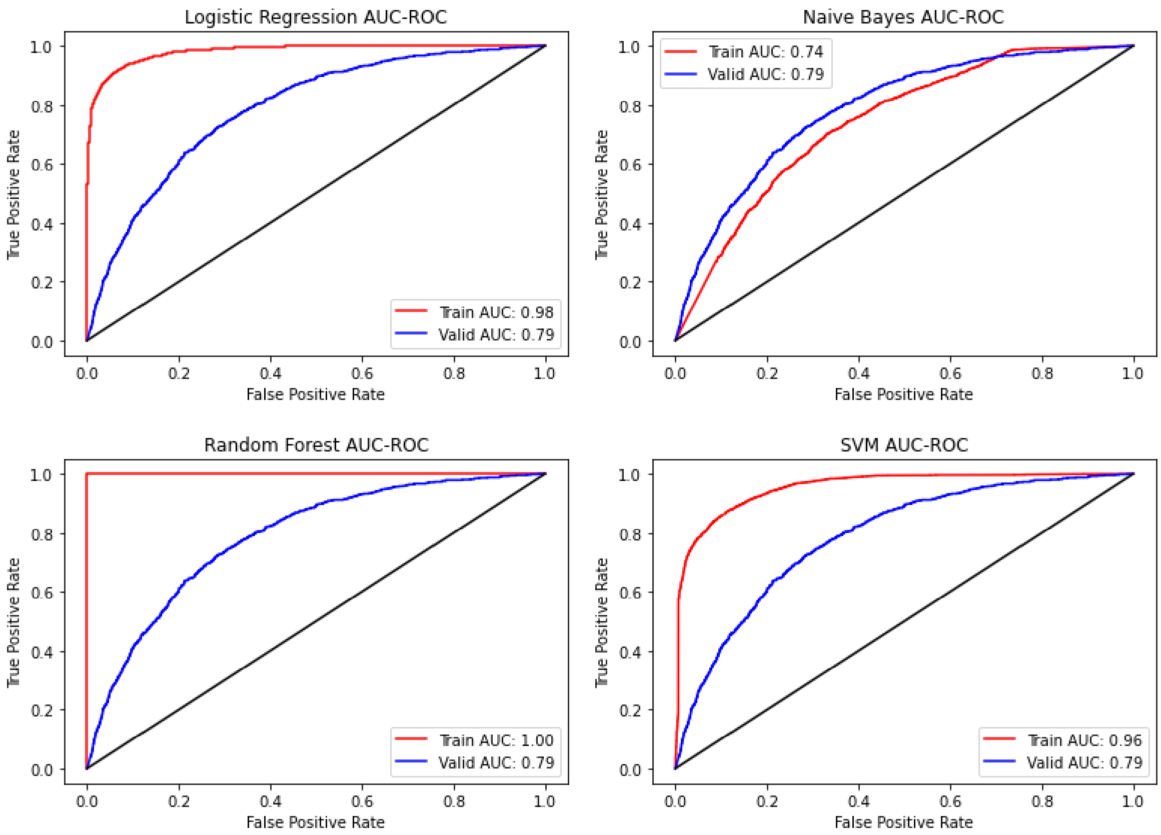

Figure 4. AUC-ROC for Logistic Regression, Naive Bayes, Random Forest and SVM.

Model Optimization

In the previous step, logistic regression was chosen as the best performing model. Next, the logistic regression model was ready to be explored further so that its parameters could be optimized to create an even more powerful algorithm based on the training and validation data. Feature importance, sample size, number of features, and hyperparameters were all examined with the intent to adjust parameters and hyperparameters as needed to create a final, best-performing model to be implemented on the test data.

Feature Importance:

First the features that the classifier is using to make decisions were visualized (Figure 5). Features with the highest coefficients (‘positive’ features) predict death and the lowest coefficients ‘negative features) predict no death. Top ‘Positive’ (implying imminent death) flagged words made logical sense, describing severe infections, chronic severe medical conditions, and sepsis (endocarditis, flagyl (treatment for sepsis), lymphoma), and possile signs of imminent death ('pupil', 'prognosis', 'dnr' (do not resuscitate'). The highest ranked ‘Negative’ flagged word was 'extubated'. This also made sense, as patients who have been extubated (taken off vent) are typically improving in health. Many of the subsequent negative flagged words were not clearly descriptive of improved health status.

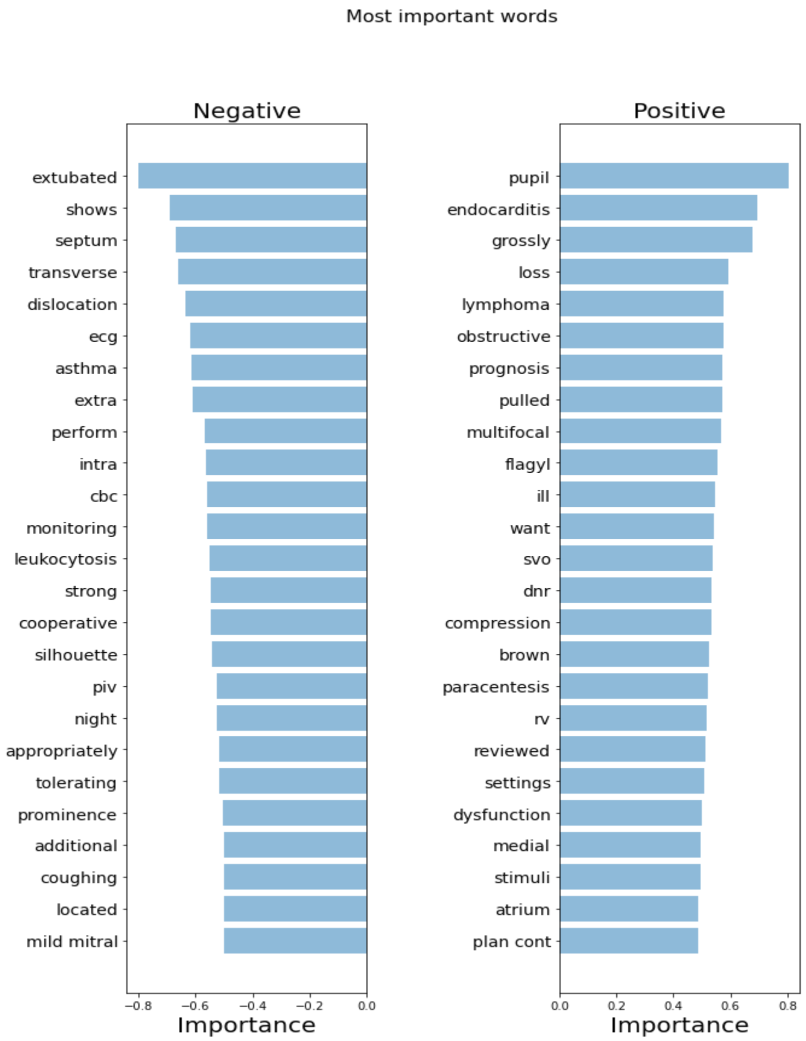

Figure 5. Feature Importance

Sample Size

Next, a learning curve was plotted to examine whether enough samples were available for the model to make good predictions (Figure 6). The AUC of the training set was high, and the Cross-validation score was also high, and began to approach the training score curve as the number of samples increased. This indicated that the model was learning well with the number of samples provided; more data would not necessarily make the model better. Some overfitting is indicated since there is a gap between the training and cross-validation curves.

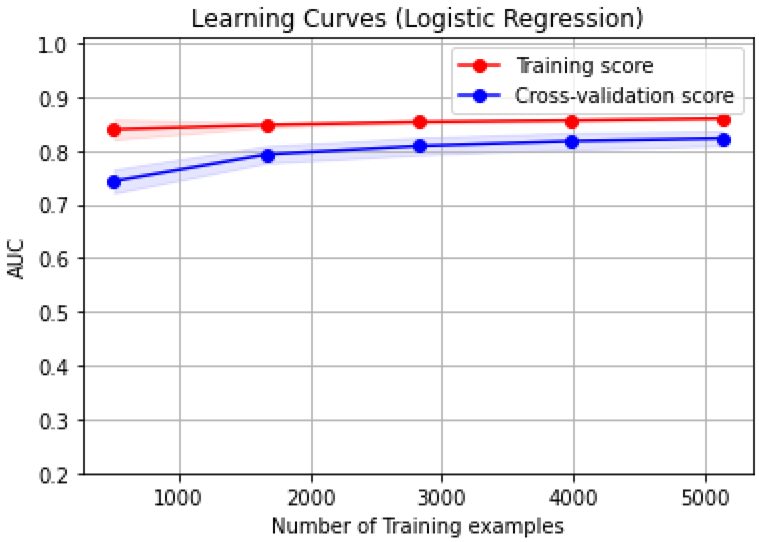

Figure 6. Sample Size

Number of Features

The maximum number of features input for CountVectorizer were examined to determine this effect on the model's results (Figure 7). As the maximum number of features increased, overfitting occurred. The existing max_features value of 3000 was deemed appropriate.

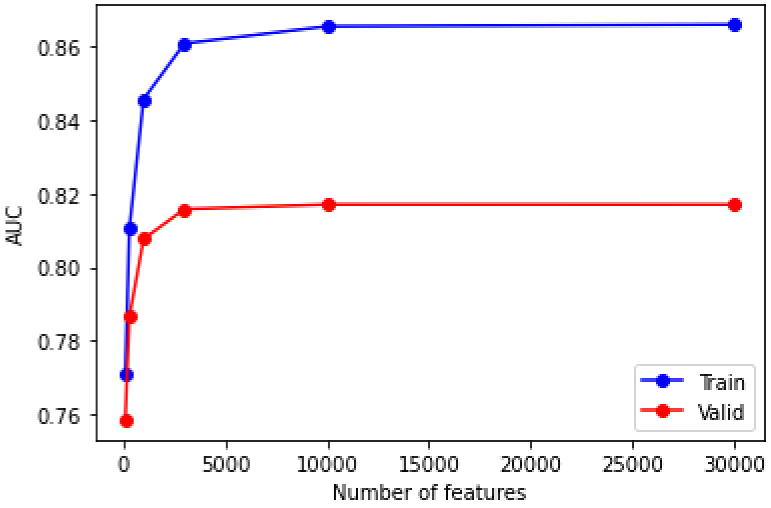

Figure 7. Maximum number of features.

Hyperparameter Optimization - Regularization Strength

To further optimize the model, the effect of C on performance was explored (Figure 8). C is a parameter that determines the strength of regularization; higher values of C correspond to less regularization. The best value of C based on this analysis was between 0.0001 and 0.001.

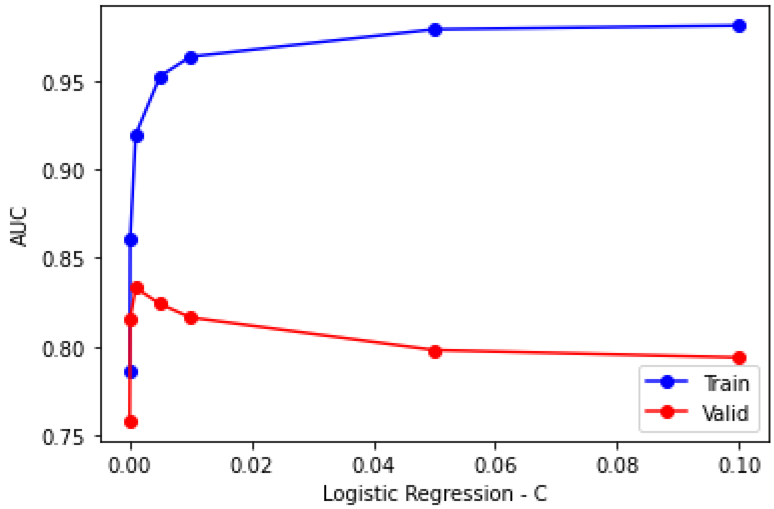

Figure 8. The effect of C on ROC-AUC.

Hyperparameter Optimization - GridSearch

Finally, a GridSearch was performed to help determine the best combination of logistic regression hyperparameters in an exhaustive search. GridSearch determined the best performing parameters to be: 'C': 0.0001, 'max_iter': 100, 'penalty': 'l2', 'solver': 'sag'. Gridsearch results were visualized with an AUC-ROC curve (Figure 9) and performance metrics were calculated (Figure 10).

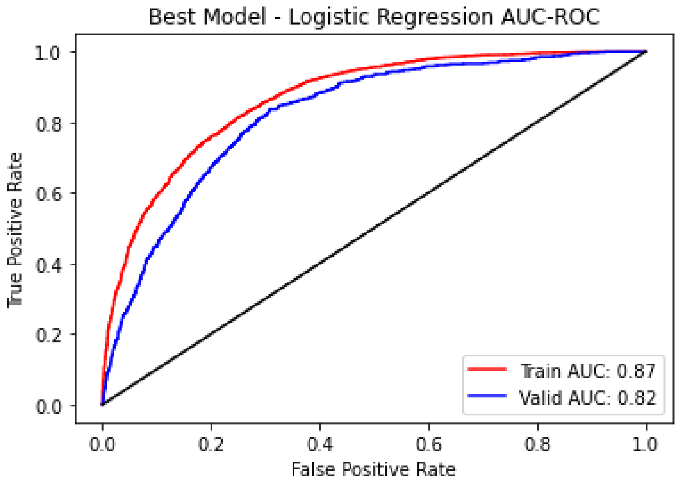

Figure 9. ROC-AUC for best logistic regression model.

Performance Metrics - Logistic Regression		
	Basic Model	Best Model
AUC	Train:0.978 Valid:0.781	Train:0.868 Valid:0.822
Accuracy	Train:0.917	Train:0.781

	Valid:0.717	Valid:0.703
Recall	Train:0.917 Valid:0.718	Train:0.829 Valid:0.837
Precision	Train:0.917 Valid:0.187	Train:0.757 Valid:0.197
Specificity	Train:0.917 Valid:0.717	Train:0.734 Valid:0.691

Figure 10. Performance metrics for best logistic regression model.

Results

Once the model was finalized, test data was processed using the optimized algorithm. A confusion matrix was created as a visualization of the model's performance, describing the quantity of patients who were true positives (actual deaths) and true negatives (actual survivors) with those who were predicted positives and predicted negatives (Figure 11). Performance metrics (Figure 12) were based on values in the confusion matrix.

Confusion Matrix - Best Model Test Data			
	Predicted +	Predicted -	Total
+ death	587	148	735
- death	2910	4451	7361
total	3497	4599	8096

Figure 11. Confusion Matrix

Performance metrics were calculated for the best model:

- Accuracy: Aggregation of model's performance. Out of all patients in the database, 71% of patients' +/- mortality was predicted correctly.
- Recall/Sensitivity: 80% of actual deaths were predicted correctly.
- Specificity: 70% of patients who survived were correctly predicted to survive.
- Precision: 21.1% of all predicted deaths were correctly predicted to be death. This means that although the model captured 80% of the patients who were truly dying, many of those predicted as deaths actually survived.

Performance Metrics: Train, Validation, and Test Data			
	Train	**Validation**	**Test**
AUC	0.868	0.822	0.828

Accuracy	0.781	0.703	0.711
Recall	0.829	0.837	0.799
Precision	0.757	0.197	0.211
Specificity	0.734	0.691	0.702

Figure 12. Performance metrics using best model on Train ,Validation, and Test Data

An AUC-ROC curve was also plotted to visualize the model's performance on the test data when compared with the training and validation data (Figure 13). There remained evidence of over-fitting (as shown by the space between the training curve and the validation and test curves, but the model is much improved in that respect compared with the baseline logistic regression model.

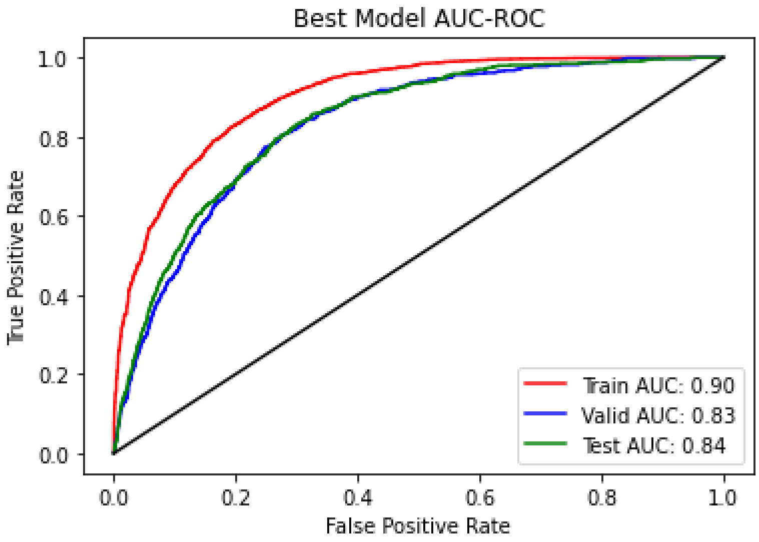

Figure 13. Comparison of ROC-AUC for Training, Validation, and Test data

Interpretation of Model Results and Implications

Using the results from the optimized model's confusion matrix, outcomes at an imaginary average community hospital in the US were projected. Based on data from the American Hospital Association[6], the average community hospital in the US has approximately 6589 admissions per year. ICU stays accounted for 27% of these admissions[7]. Therefore, it was inferred that 1779 hospital stays would include at least one ICU stay. The confusion matrix, indicating counts of actual and predicted deaths and survivals, is applied to average hospital data in Figure 14.

Confusion Matrix - Best Model

Average US Hospital			
	Predicted +	predicted -	Total
+ death	129	33	162
- death	639	978	1618
total	768	1011	1779

Figure 14. Confusion matrix extrapolated to average hospital ICU admissions and deaths

The cost savings analysis* was based on two possible actions a health system could take:

- Option 1: No intervention; costs remain the same.
- Option 2: Employ machine learning model. Patients whose charts are flagged as ‘high risk’ of death are given extra attention within 24 hours of admission. Per physician’s discretion, consideration is given for discharge to home, hospice or palliative care.

Assumptions:

- Average US hospital costs per year from ICU hospital stays resulting in death: $5,310,550[6,7,8,9]..
- The average length of hospital stay for patients who spent time in the ICU 10.5 days, with 5.25 days in the ICU. [7]
- Average cost of hospital stay that includes ICU admission: $32,879[8,9]
- Cost of hospice care in place of hospitalization: $11,820[5]
- If Option 2 used, 75% of those truly at end of life would be discharged to hospice 24 hours after admission.

Savings Analysis*		
	Option 1	Option 2
Cost of hospital stays	$5,310,550	$5,310,550
Savings	$0	$2,336,273
Hospice care costs	$0	$1,143,534
Total Savings	$0	$1,192,739
Percent Savings	0.00%	22.46%

Figure 15. Savings Analysis*

* Savings analysis details and calculations are in the Appendix.

If the machine learning model were used, an average savings of 22.5% (approximately $1.19 million) could be achieved. (Figure 15). Additionally, 80% of the patients who are at the end of life would be flagged after 24 hours of hospital admission. If this model was applied to all 5198 community hospitals[6] within the US, a savings of nearly $6.2 billion per year could be achieved.

Limitations and Opportunities

There are many opportunities for further exploration and refinement of the dataset and model. First, the dataset that this analysis was based on included any and all caregiver notes written within 24 hours of a patient's admission. To potentially improve the model's power and accuracy and allow for a quicker assessment after admission (less than 24 hours), only the admission note could be collected. Another alternative, if sufficient data was available from patients with specific high-risk medical conditions, would be to implement separate versions of this model within a specific diagnosis categories (such as 'liver disease' or 'cancer'). This could isolate commonly-documented features within the diagnosis categories that could significantly improve accuracy and reduce the proportion of false positives.

In addition to dataset adjustments, other options for feature engineering, optimization, and even choice of machine learning model could be explored. NLTK's regular expressions, using TF-IDF in place of countvectorizer, and normalizing and grouping the tokens with stemming and lemmatization could be implemented to determine the effect on outcomes. Additionally, since the results of logistic regression and SVM were quite close, an optimized SVM model could be tested and compared with the optimized linear regression model.

As an alternative to the traditional machine learning models of logistic regression and SVM, the preprocessed data could be fed into a deep learning framework which could take the sequence of words into account, creating a more sophisticated model that may be more accurate in its predictions. Python Tensorflow, HuggingFace Transformer, Keras, and spaCy are some of the libraries that could be utilized to create a deep learning model.

Lastly, a stacked ensemble model could be created using the best NLP model along with a model that predicts mortality based on other features included in the MIMIC III dataset, such as laboratory data, vitals, and other patient data. The combination of the two models into a meta-model would likely create an even stronger algorithm for predicting mortality.

Conclusion

Based on this analysis, it was found that using caregiver notes from the first 24 hours of hospital admission, a better-than-random model can be built to predict mortality in ICU patients. A predictive model such as this could be used to support physicians in a high-stress environment who are often pressed for time. Such a model would be beneficial to both patients and healthcare organizations, helping to reduce unnecessary and intrusive procedures and substantially reducing financial costs.

It is important to emphasize that the model's predictions are meant to be used to flag patients who are at high risk of mortality based on caregiver notes, so that a timely discharge is considered. Many surviving patients were, in fact, predicted to die with the model. It is up to the physician and care team to determine the patient's risk of death. A stacked ensemble model could be created, which would allow for a stronger algorithm to predict mortality in ICU patients, and this would likely reduce the proportion of false positives and improve model accuracy.

References

1) ICU Outcomes - UCSF
2) Outcomes From Intensive Care in COVID-19 Patients
3) Patterns of Cost for Patients Dying in the Intensive Care Unit and Implications for Cost Savings of Palliative Care Interventions
4) Long Term Care Market Size, Share & Trends Analysis Report by Service (Home Healthcare, Hospice, Nursing Care, Assisted Living Facilities), by Region (North America, Europe, APAC, Latin America, MEA), and Segment Forecasts, 2020 - 2027
5) Better care of sickest patients can save hospitals money, says largest study of its kind
6) Fast Facts on U.S. Hospitals, 2020
7) ICU Occupancy and mechanical ventilator use in the United States
8) Average hospital expenses per inpatient day across 50 states
9) Critical Care Statistics - ICU costs per stay
10) Introduction to Clinical Natural Language Processing: Predicting Hospital Readmission with Discharge Summaries
11) Studies Point to Big Drop In Covid-19 Death Rates

*Appendix can be found in the attached google sheet, ‘Appendix_ICU_Mortality’.