

Medical Applications Of Machine Learning - Finale Project

Yuval Brunshtein, yuvalbrunshtain@gmail.com,

June 24, 2025

Abstract

In this paper which we write as a finale project document to describe our project done under the course Medical Applications of Machine Learning at the specialty of Data Engineering in the department of Industrial Engineering at Ariel University which is lectured by Dr Orit Raphaeli. In this work we provide statistical analysis, supervised analysis, unsupervised analysis and a mixture of the two learning methods of medical data. Specifically we take the MIMIC dataset which describes data such as: Vital signs, Lab results, length of stay in ICU, age, death and other parameters and draw medical conclusions and other conclusions from our analysis of the data. We also formulate a research question backed by a literature review for our work.

1 Research Question

Can distinct subgroups be identified within the ICU patient population using unsupervised learning, and how can this information contribute to improving 30-day mortality prediction?

1.1 Background

ICU patients represent a highly heterogeneous population – including young trauma cases, elderly patients with sepsis, oncology patients, and post-operative cases. This diversity complicates accurate clinical decision-making, especially regarding prognosis and resource allocation. Unsupervised learning, particularly clustering methods, provides tools for identifying natural latent subgroups within the patient population, without the need for mortality labels or diagnoses. For example, Mayne et al. (2024) applied clustering based on medical resource needs and found distinct subgroups with unique profiles – such as younger patients with favorable prognosis and critically ill patients with high mortality rates. Lintu et al. (2023) explored a semi-supervised approach – where clustering is influenced by survival outcomes – and found that this method improved the accuracy of deep learning-based predictive models. Thus, there is growing recognition that identifying subgroups not only enables better characterization of ICU populations, but also supports the development of personalized prognosis models.

1.2 Clinical and Research Justification

From a clinical perspective, identifying ICU subgroups serves not only to predict mortality but also to tailor treatment to individual patient profiles. For example, an elderly patient with high mortality risk may benefit less from aggressive interventions. Additionally, early identification of patients at risk of deterioration enables more efficient allocation of hospital resources, including ICU beds, treatments, and staffing. Moreover, subgroup identification can support coordinated care and ethically informed decisions (such as around DNR – Do Not Resuscitate), based on clinical status and expected life expectancy. The ability to assign patients to subgroups with particularly high expected mortality (such as cluster 3 in Mayne’s study, with a 76

From a research perspective, Mayne et al.(2024) demonstrated that ICU subgroups identified in one hospital system do not necessarily replicate in other datasets, indicating a need for personalized clustering approaches. However, there remains a lack of studies connecting clustering directly to short-term mortality prediction. Lintu et al.(2023) advanced the field by applying semi-supervised learning to improve mortality prediction based on patient subgroups, showing that deep learning models perform better when aligned

with these clusters. Hou et al. (2020) focused on developing a 30-day mortality prediction model for sepsis patients using XGBoost and showed improved performance compared to classical models – though their work did not include a clustering component, leaving a gap in population stratification. Therefore, our study fills an important research gap: It not only explores whether distinct patient subgroups can be identified in a heterogeneous ICU population, but also whether this information improves mortality prediction.

1.3 Goal

The objective of this project is to investigate whether distinct subgroups can be identified within the intensive care unit (ICU) patient population using supervised, unsupervised, and hybrid machine learning methods, and to evaluate whether such a risk stratification contributes to improved prediction of 30-day mortality. To achieve this, we will apply both unsupervised clustering techniques (such as K-means and DBSCAN) and semi-supervised to a large clinical dataset (e.g., MIMIC-III) in order to uncover latent patterns within the patient population. Subsequently, we will conduct statistical analyses and compare the performance of predictive models (such as XGBoost, Logistic Regression, and Random Forest) for 30-day mortality prediction, both within the identified subgroups and across the entire unsegmented population. We would like to examine whether stratifying ICU patients into clinically homogeneous subgroups enhances prognostic accuracy—potentially offering new insights to support personalized decision-making in critical care settings.

2 Data Pre-processing

Before using our data in order to analyze its attributes using statistical, supervised and unsupervised methodology it requires pre-processing where each methodology requires a different kind of pre processing of the data.

Using the subsections below and the conditions and considerations described in each sub-section we create multiple datasets for multiple purposes each customized for its own purpose.

2.1 Data Description

Our dataset is a MIMIC data set or formally, MIMIC - Medical Information Mart for Intensive Care, MIMIC are specific types of datasets which provide extensive information about patients. More specifically it contains information about a patient's: Length of Stay in the ICU, Death, Vital signs, Lab results, Ethnicity, Diagnoses, age and gender. Specifically our data set contains 4559 rows and 72 columns. Some important columns which we will discuss extensively in our project are:

Thirty Day Expire Flag - denotes a death of a person 30 days from being emitted into the ICU.

ICU los - Denotes the length of stay a patient stays in the ICU.

Sofa - denotes the sofa score, Sofa scores which means Sequential Organ Failure Assessment Score this score ranges from 0 to 24 with higher scores indicating severe organ dysfunction.

There are obviously multiple more parameters which we will use and explain when necessary.

2.2 Removing Unnecessary Columns

For all type of methodologies there are columns which we won't need and will only do harm throughout the analysis. Such as:

icustay_id, hadm_id, subject_id, is_male, race_white, race_black, race_hispanic, race_other

These columns are removed due to multiple reasons:

1. Data complexity, when training machine learning models we want our data to be as dimension redundant as possible while still maintaining the important information which is needed to accurately build models and provide accurate analysis.

2. The content of the columns - The aforementioned columns aren't describing useful information or describing repeating information for example, the *id* columns won't have or shouldn't have a connection to a

death of a person, the *race* columns are already contained in the *ethnicity* column and the *is_male* column is already contained within the *gender* column.

Out of the described reasons we can effectively say that the removal of the aforementioned columns won't damage the quality of our analysis and won't cause information removal.

Note: The *race* columns are removed in all analysis except statistical analysis.

2.3 Missing Values

The missing values in the data set are an important factor and consideration, in the one hand missing values can cause models to not learn properly and therefore need to be effectively filled or removed in the other hand we aren't doctors and even if we were we don't have extensive extra information about a patient to know exactly what to fill in instead.

We need to analyze the missing values in the data set to understand exactly where the missing values are and we do so and achieve the following results:

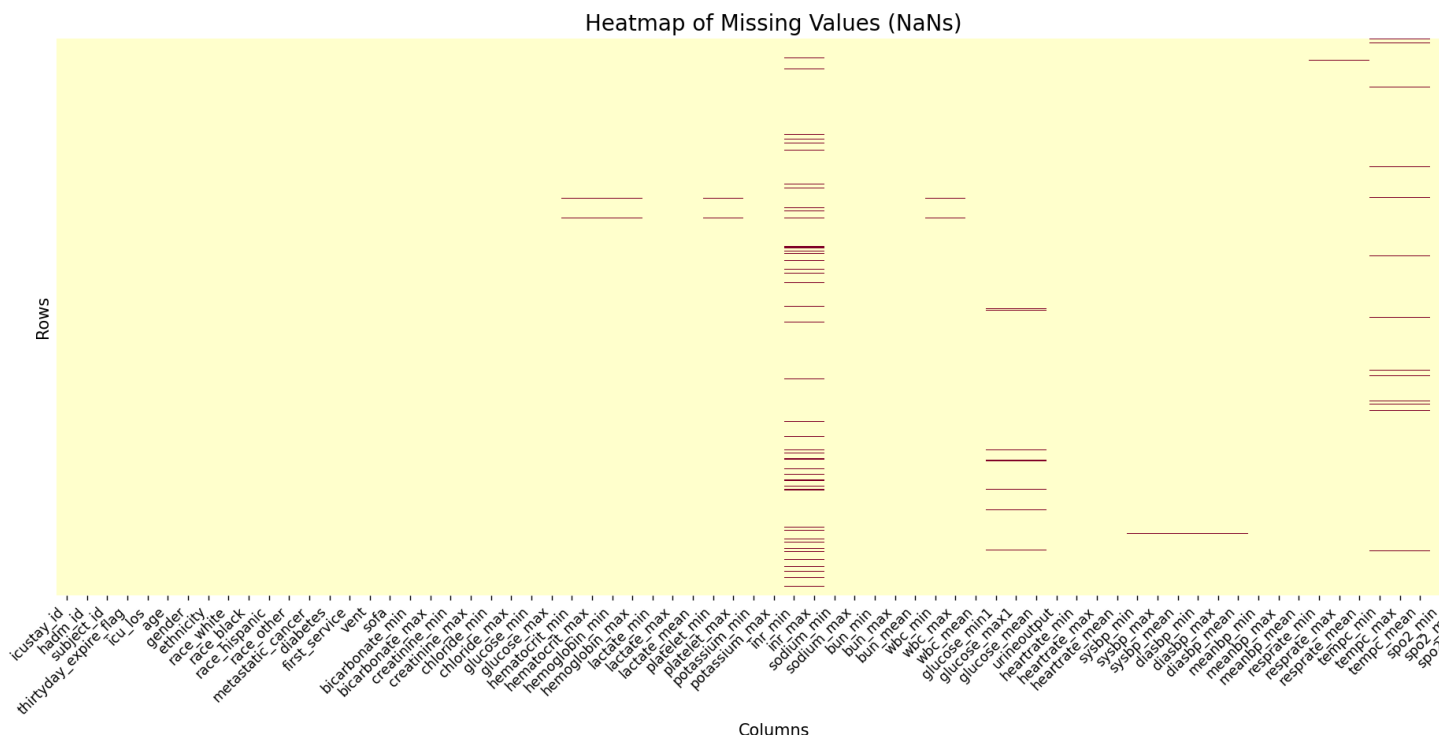


Figure 1.1 Analysis of Missing Values In The

Out of the analysis we see that mostly the data which is missing are in the following specific columns: inr columns, Glucose columns and tempc columns. Due to the fact there are mostly numerical values which are singular blood tests or vital signs and we see later that they do not possess high correlation to the death rate neccessairly then we make the assumption that we can fill in the missing values in all columns using the mean value of the column.

2.4 Normalizing Data

In normalizing the data there is also a lot of thinking to be done, for instance tree based models such as Boosting models, Random Forest and Decision Trees don't require numerical normalization but models such as Neural Networks, Logistic Regression, Clustering methods and others do require said normalization.

When deciding exactl what to do in this section we have devoted a lot of thought, some of us said rightly so

that medical data might have specific numerical representations that need to be properly fed into the model otherwise it might not represent what's actually happening clearly while keeping values as they are might bring high variance in the analysis. The way the data is numerically distributed before normalizing:

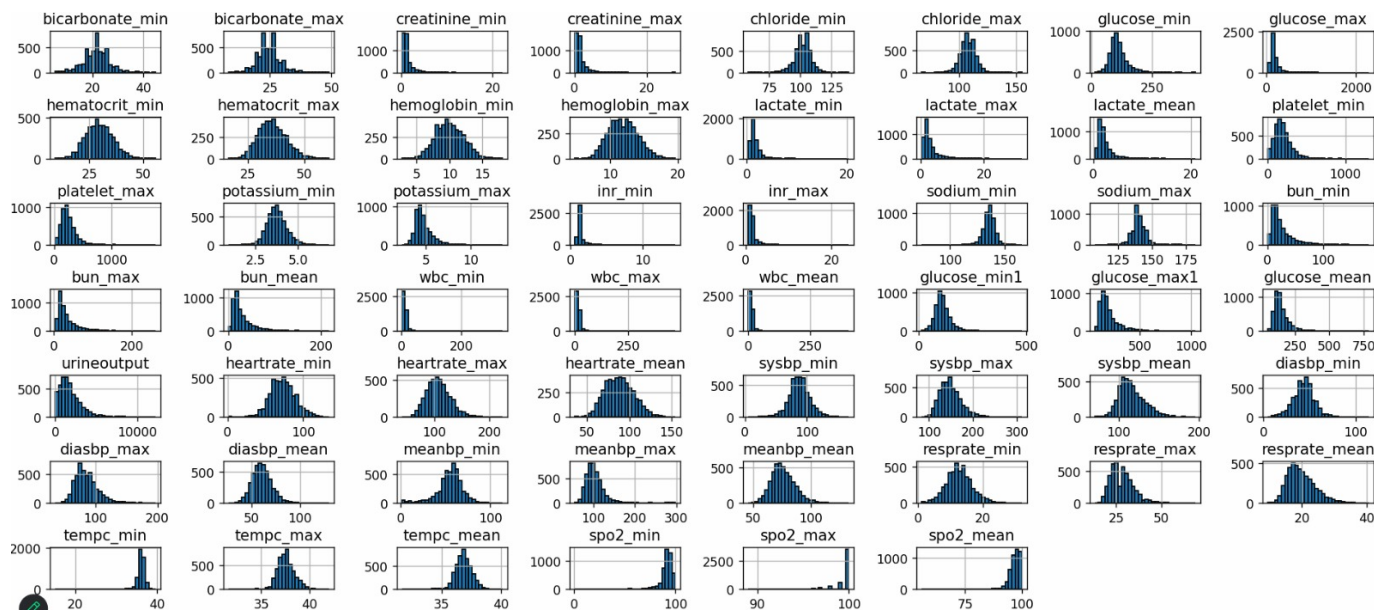


Figure 1.2 Analysis of Numerical Values of The Data

In this graph which we produced we see that values vary widely in ranges while some features like *urineoutput* reach values like 10,000 other values like *creatinine_max* barely reaches 20.

Out of this analysis we made a decision, for statistical analysis and tree based models we do not provide normalization because we want the data to be mostly as it is and to see numbers which are more readable to us in the statistical analysis. But in other models such as Logistic Regression, Clustering models and neural networks we provide a Z-score normalization method where basically what it does is the following:

$z = \frac{x - \mu}{\sigma}$; where, x - original value, μ - Mean of the column, σ - standard deviation of the column, and so what it does essentially is the brings the mean of each column to zero and ensuring that the standard deviation becomes 1. This makes sure all numerical columns are the same scale and therefore in methods like clustering or logistic regression it won't throw off the variance of the model and won't overly represent a feature because of it's previously high values. In gradient based methods it helps with convergence speed and numerical stability.

2.5 One-Hot-Encoding

When talking about one-hot-encoding we use this in order to represent the data in a way which would be suitable for machine learning models to be trained on.

Here too we make a two different datasets, for statistical analysis we do not one-hot-encode because we do want said representation in a categorical way.

But in the models including XGBoost, Random Forest, Logistic Regression, Neural Network and clustering algorithms we do provide a dataset which we have been one-hot-encoded.

Brief recap, one-hot-encoding creates a column for each category in a categorical column and in a row where said category exists then the created column in that specific row is assigned 1 value while when the category is absent a 0 value is assigned in the said category column.

3 Statistical Analysis

In this section we provide visual and statistical analysis. We provide different graphs to describe our data, provide thorough statistical tests and draw conclusion from said statistical analysis.

3.1 Variable Distribution

To understand the way the data behaves better we plot the distribution of most of our data.

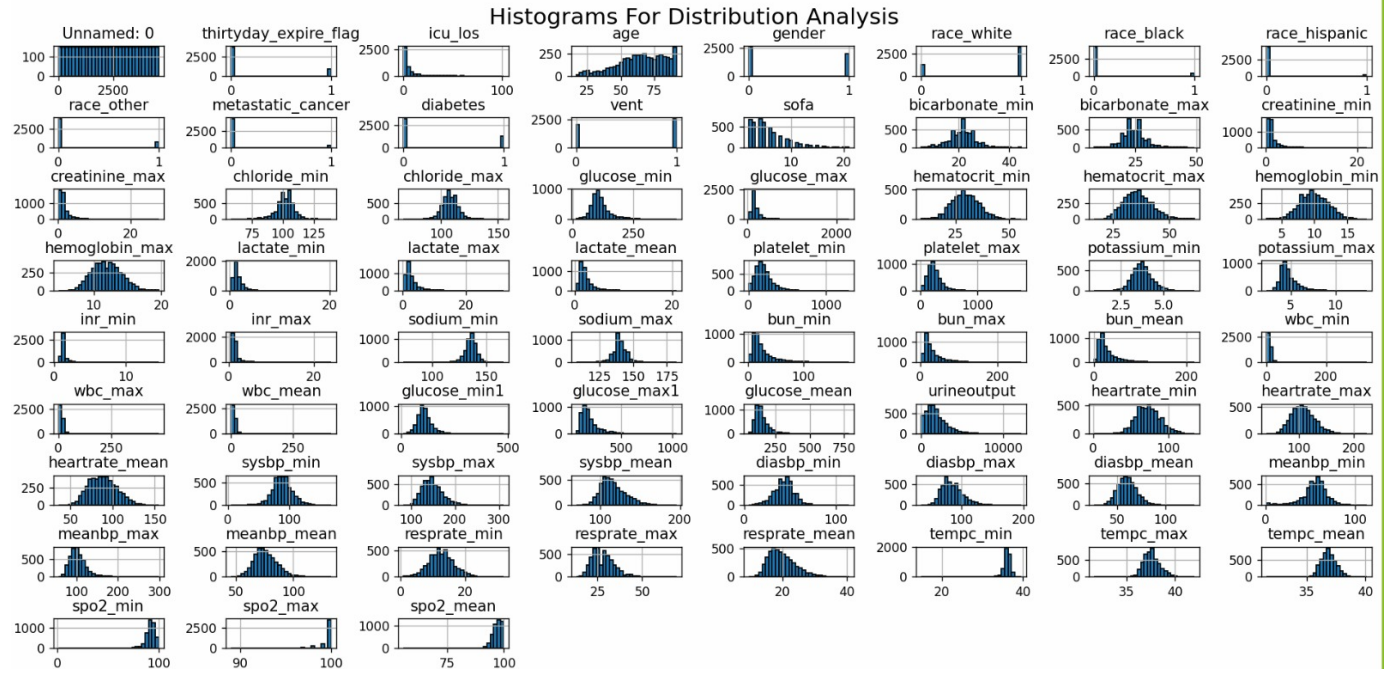


Figure 1.3 Analysis of Distribution of Features

Out of this plot we can draw all kinds of conclusions for example that the population of older people is more dominant in the dataset. We can also see how the distributions of the different data are mainly, Positive Skewed(Right skewed) Distribution and Normal Distribution. The meaning of the Positive Skewed distribution is mostly that most patients in the clinic are stable while some patients, less patients are at harsher condition we see this in the Sofa score distribution for example where most Sofa values are well below 10 but rightly skewed towards 20 meaning there is a low amount of people with $10 \leq Sofa \leq 20$ and this is something we would expect.

3.2 Different Plots, Graphs & Results

We test to see the correlation between the Sofa scores and the thirty day death feature and we get the following graph:

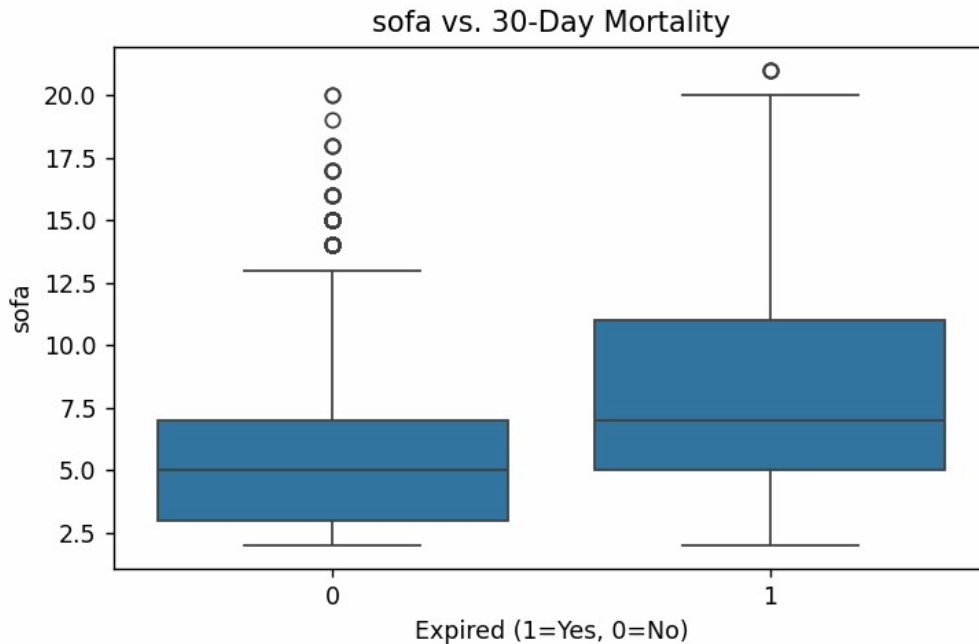


Figure 1.4 Sofa VS Thirty Day Expire Flag

We see exactly what we would expect where higher Sofa scores indicate higher mortality. We now plot a correlation matrix and the main goal is to check strong correlations to the thirty day expire flag this will also help us draw conclusions in the unsupervised and supervised analysis part.

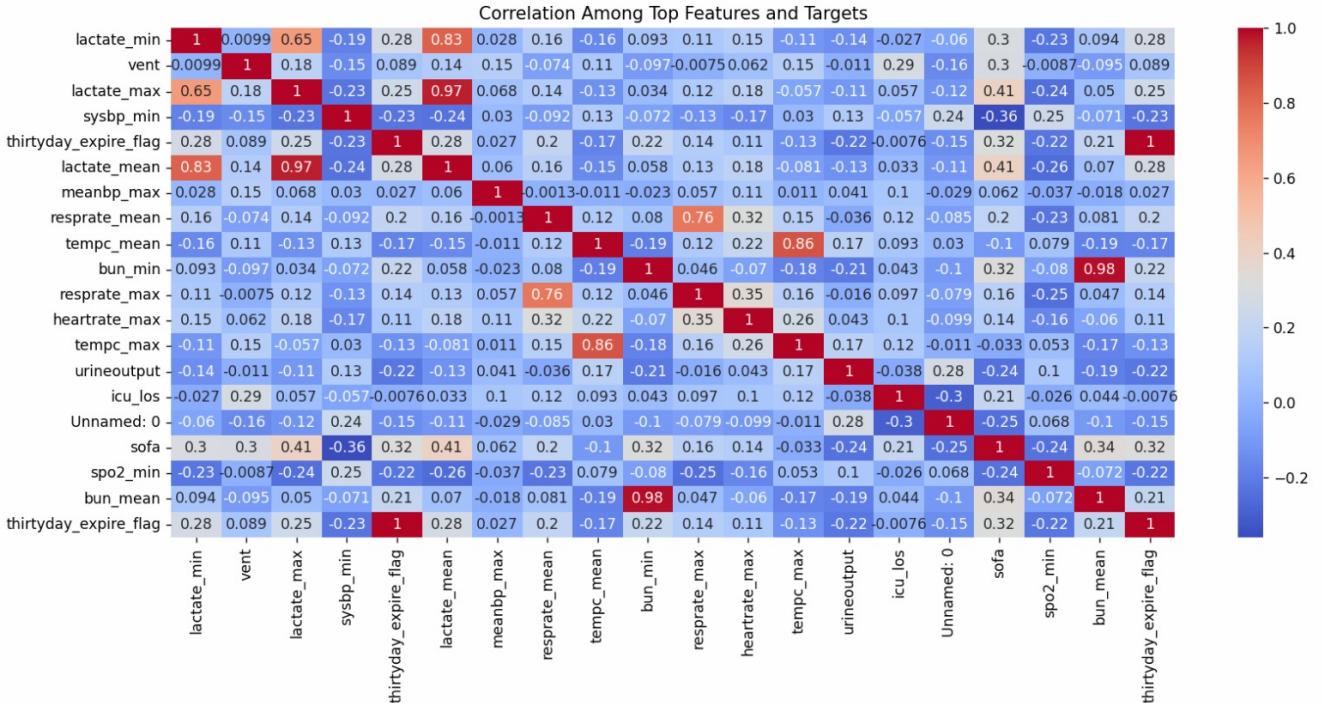


Figure 1.4 Correlation Matrix

What we see is that for the thirty day expire flag there is a relatively higher correlation between that feature and the sofa score 0.32 to be exact we also see some interesting other relatively high correlation that will come into play later on in the analysis such as: *lactate_min/max, /mean*.

We can also see that vent also has relatively higher correlation to Sofa, mainly a lot of features have high correlation to Sofa this likely means that the said Sofa feature has a direct connection and meaning to a lot of the other features which is to be expected since Sofa describes the overall's body condition.

We decide to also examine the connection between patients who have been assigned a ventilation machine to the thirty day expire flag (death) and we achieve the following:

| <i>Vent/Death</i> | 0 | 1 |
|-------------------|------|-----|
| 0 | 1711 | 316 |
| 1 | 1957 | 573 |

and so out of this table and in general we can calculate the death rate for people who did have ventilation and people who didn't have ventilation. We get that people who weren't assigned ventilation meaning vent is 0 have a death rate of 0.1558 while people who were assigned ventilation meaning vent is 1 the death rate is 0.2264.

We can thus conclude that people who were assigned ventilation have higher mortality rate this is of no surprise because typically people that need ventilation are typically at a worse medical condition. To further emphasize this connect we also plot the ventilation in relation to the mortality rate and achieve the following graph:

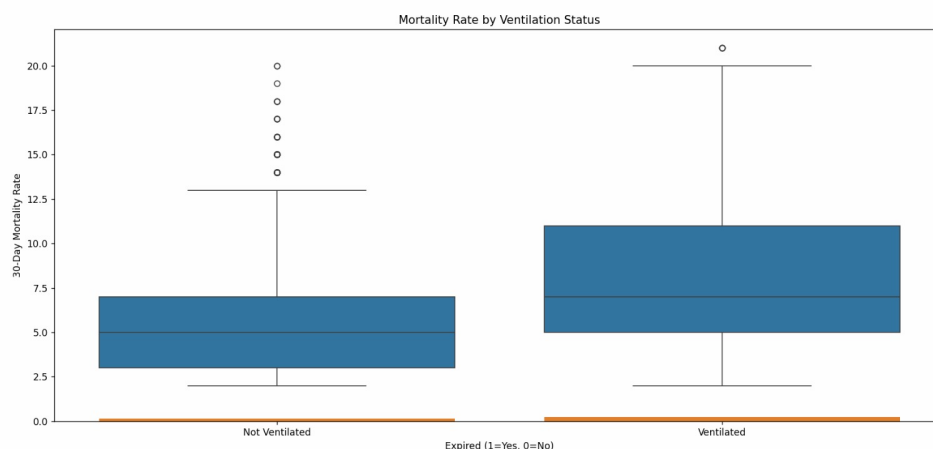


Figure 1.5 Ventilation VS Thirty Day Expire Flag

The last graph we would like to show is a graph which is important for our models evaluation. The following graph emphasizes the class imbalance which exists in the data:

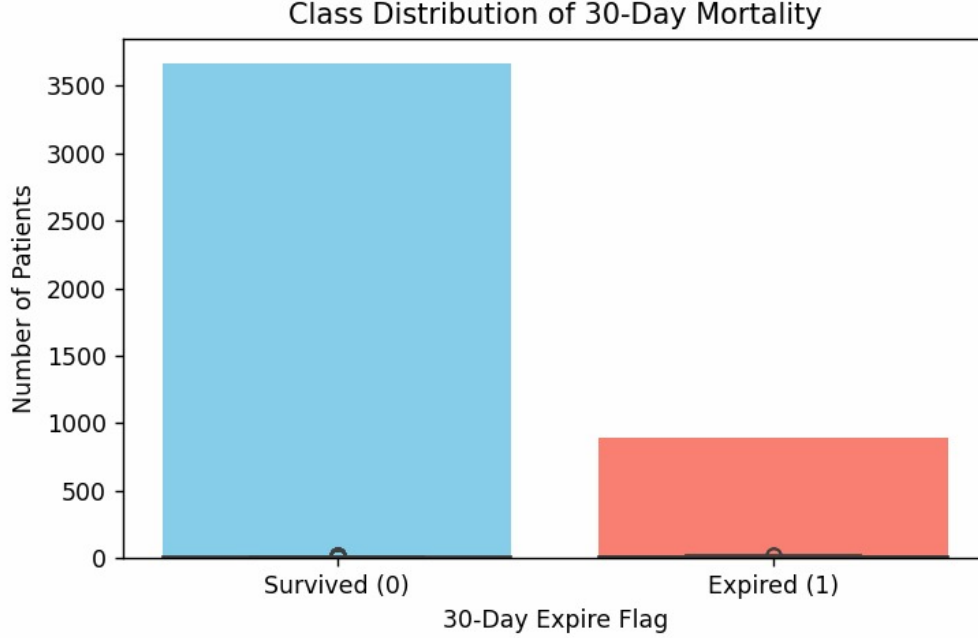


Figure 1.6 Class Imbalance

When training the models it's very important to understand the class imbalance which exists in the data, it will be an explanatory factor of why a model might perform worse when predicting death rather when predicting survival.

3.3 Statistical Tests

In order to fully confirm the relationship we have found between said features we want to conduct statistical tests.

3.3.1 Sofa Vs Thirty Day Expire Flag, Mann-Whitney U Test

Sofa is a numerical feature therefore we need a statistical test which can handle numerical features and at the same time not assume Sofa is normally distributed because it's not.

Thus we use the Mann-Whitney U test which is defined as followed:

Note: We calculate both tests in python obviously using a library but we do want to justify our calculations by explaining how the calculations actually happen.

Two groups are defined Group A which denotes Sofa scores for survived individuals (thirty day expire flag 0) and group B which denotes sofa scores for people who died (thirty day expire flag 1). n_A, n_B are the number of samples of A, B respectively we then calculate the sum of ranks R_A, R_B for each group and then use the formulas:

$U_1 = n_A \cdot n_B + \frac{n_A \cdot (n_A + 1)}{2} - R_A$ and $U_2 = n_A \cdot n_B - U_1$ then we denote $U = \min(U_1, U_2)$ and we solve the Z - score for $Z = \frac{U - \mu_U}{\sigma_U}$ where $\mu_U = \frac{n_A \cdot n_B}{2}$, $\sigma_U = \sqrt{\frac{n_A \cdot n_B (n_A + n_B + 1)}{12}}$, then we go to the table and extract the p-value.

For this test using python we achieve a p-value of $p\text{-value} = 2.4 \times 10^{-73}$ which makes sense given we have so much data then we can achieve a very high Z-score and thus achieve a $p\text{-value} \approx 0$. We thus conclude there is a highly significance statistical difference between groups A and B thus confirming what we have already suspected and claimed above.

3.3.2 Ventilation Vs Thirty Day Expire Flag, χ^2 -test

Ventilation can be looked at as a binary categorical variable therefore we use a categorical statistical test, χ^2 - test, as I mentioned above we are trying to prove that there is a statistically significance difference between the group of people who used ventilation and the group of people who didn't. In order to do that we built something called a contingency table which is the same table as above:

| <i>Vent/Death</i> | 0 | 1 |
|-------------------|------|-----|
| 0 | 1711 | 316 |
| 1 | 1957 | 573 |

Then we calculate, $E_{i,j} = \frac{\text{row_total} \times \text{column_total}}{\text{grand_total}}$ for each cell and we compute $\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$ we then calculate the degrees of freedom and then the p-value. Finally using the python library which uses the mathematics described above we find a p-value of 2.88×10^{-9} which also confirms our original hypothesis and thus the difference between people who received and didn't receive ventilation in correspondence to thirty-day-expire-flag is a statistically significance difference.

3.4 Summary

In this section we provided a thorough statistical analysis of our data, from distributions of our features to correlations between different features to statistical tests we provide a well founded statistical analysis of the data and we make important medical conclusions and important conclusions about our data.

4 Unsupervised Learning Analysis

In this section we provide an unsupervised learning analysis of our data we use distance based methods for the build of clusters, we use K-Means, DBSCAN and PCA for different purposes such as clustering and plotting.

4.1 Clustering Algorithms

In order to optimally cluster the data we examine two different algorithms, K-means and DBSCAN.

4.2 K-Means Clustering Algorithm

We use K-means to effectively cluster our data, K-means is an unsupervised learning clustering algorithm that works as following:

Given an initiate set of k means denoted as $m_1^{(1)}, m_2^{(1)}, \dots, m_k^{(1)}$ (first iteration).

Then the algorithm iteratively updates the following way:

1. We assign each data point vector x_p to the cluster with the nearest mean measured using the Euclidean metric.
2. We then recalculate the mean of each cluster will be denoted as the cluster's centroid.

We then simply iteratively perform 1 and 2. We have here an objective function WCSS and the algorithm stops when WCSS stops improving or simply when WCSS has become stable. WCSS is an objective function which attempts to minimize the variance within the clusters.

In order to build the optimal K-means cluster we calculate the mean Silhoutte score for every test of k we test a k for $k = 2, 3, \dots, 10$. Silhoutte score is a score given to each individual data point, essentially the Silhoutte score is calculated using the following formula: $s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$, where $b(i)$ — distance between data point x_i and it's nearest next cluster centroid and $a(i)$ — average distance between x_i and all of the other points within in it's own cluster. In other words the Silhoutte score denotes how well a data point fits in it's cluster.

As we mentioned earlier, we take iterate through different k values to find the optimal one according to the Silhoutte score.

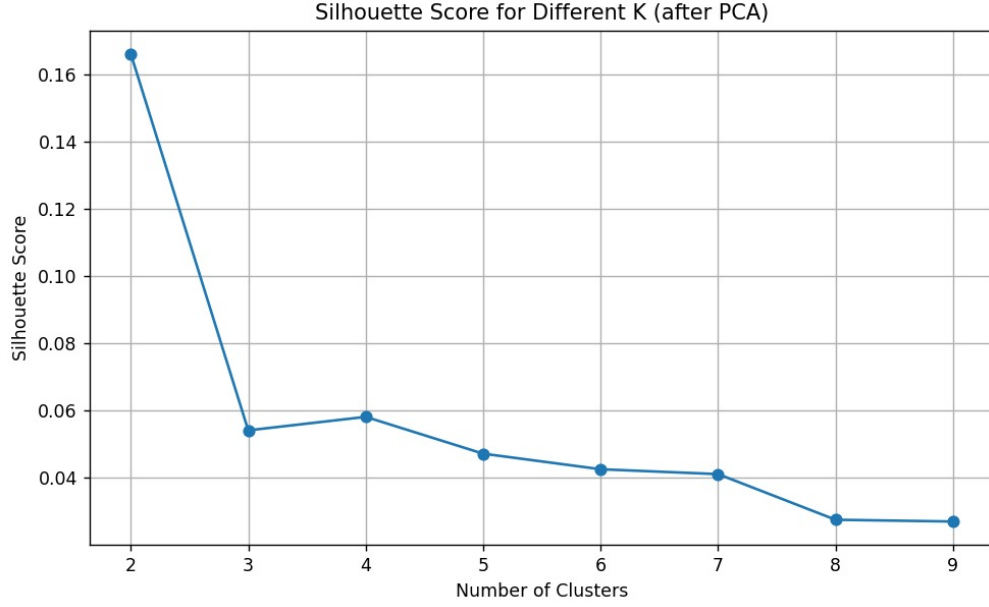


Figure 1.6 K Iterations

What we realize out of this graph is that we see that we don't really achieve high Silhouette score but after many tests we come to the conclusion that it's better to determine the quality of our clusters by actual examination of the feature values within the clusters.

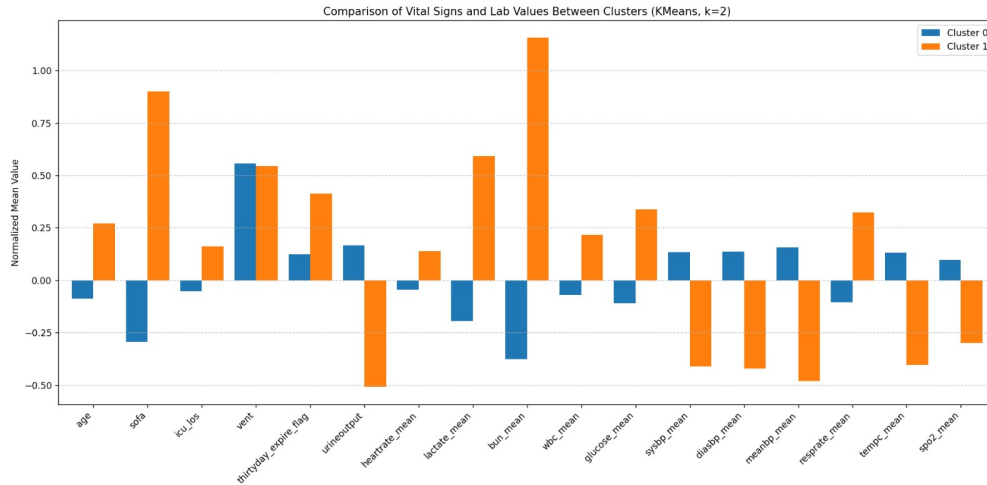


Figure 1.8 Feature Differences Between Clusters

This is an analysis of the features within the clusters of the normalized data, according to these results we see that in Cluster 0 Where we have most data we get that the sofa score is under the mean value, the age is under the mean value, the length of stay in the icu is under the mean value and we see that the expire flag is also significantly lower than in the other cluster where we see almost opposite effect where in the other cluster the age is above the mean the sofa is above the mean as well as the length of stay and as I said the expire flag is also higher. What this tells us is that the clusters actually seem to describe two completely different groups which represent certain populations within the dataset. Essentially this tells us that despite the low Silhouette Score it should be taken into consideration that each data point x_p here is a 116 – *dimensional* vector as this is post one-hot-encoding and such the Silhouette score doesn't actually represent the quality of the clusters here. We use PCA which is a dimension reduction method in order to plot the data. PCA done correctly keeps the variance of the data therefore we use it plot the clusters. It should be noted that our plot is 2 – *Dimensional* and even though PCA does keep some variance reducing

the dimensions from 116 to 2 is obviously not representative of the actual features of the clusters which is why we use it for plotting purposes only.

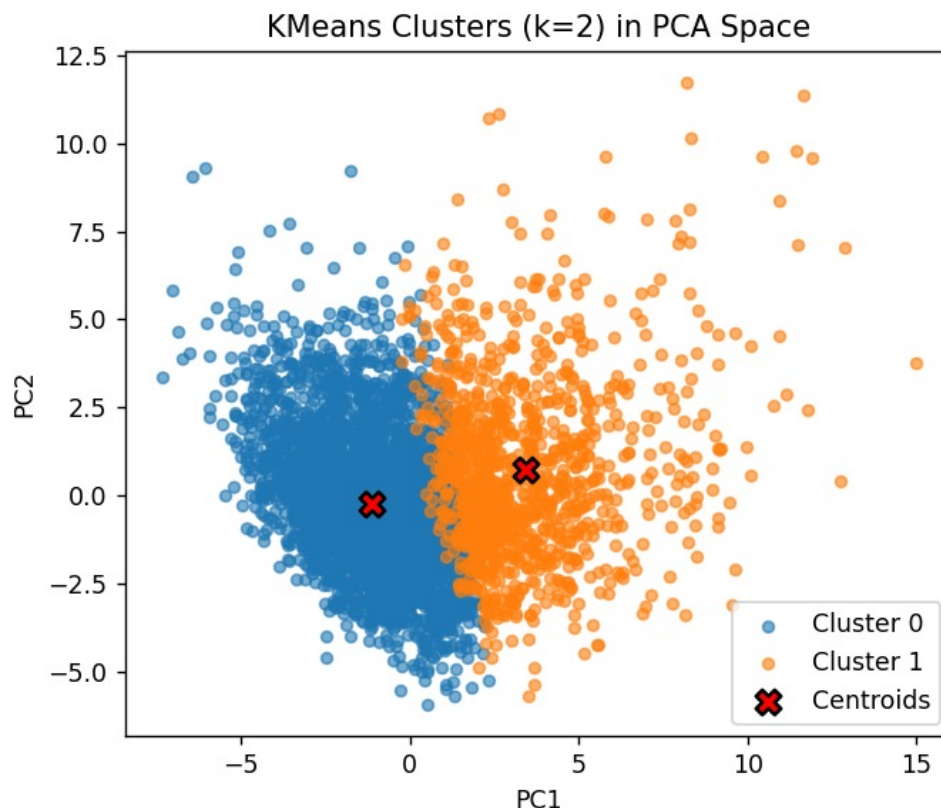


Figure 1.9 Plotted Clusters

4.3 DBSCAN Clustering Algorithms

Similarly, we use DBSCAN, (Density-based spatial clustering of applications with noise) in order to cluster our data, DBSCAN works in the following way:

We define distance ϵ which defines the neighborhood distance of every point. Clusters are based on core points, core points are points with more than a minimum defined number of neighbors. The algorithm goes as follows:

1. Find points in the ϵ neighborhood of every point, identify core points.
2. Find connected components of core points on the neighbor graph.
3. Assign each non-core point to a nearby cluster if the point's distance is ϵ from a nearby cluster otherwise it's noise.

The optimization criterion for this algorithm is actually the minimization the number of clusters.

We construct a similar build to the k-means here and we test the DBSCAN algorithm for different parameters of ϵ and minimum points for a core points and our results vary.

We show the following results because it emphasizes the difference in outcomes and what's relevant to the continuation of the work.

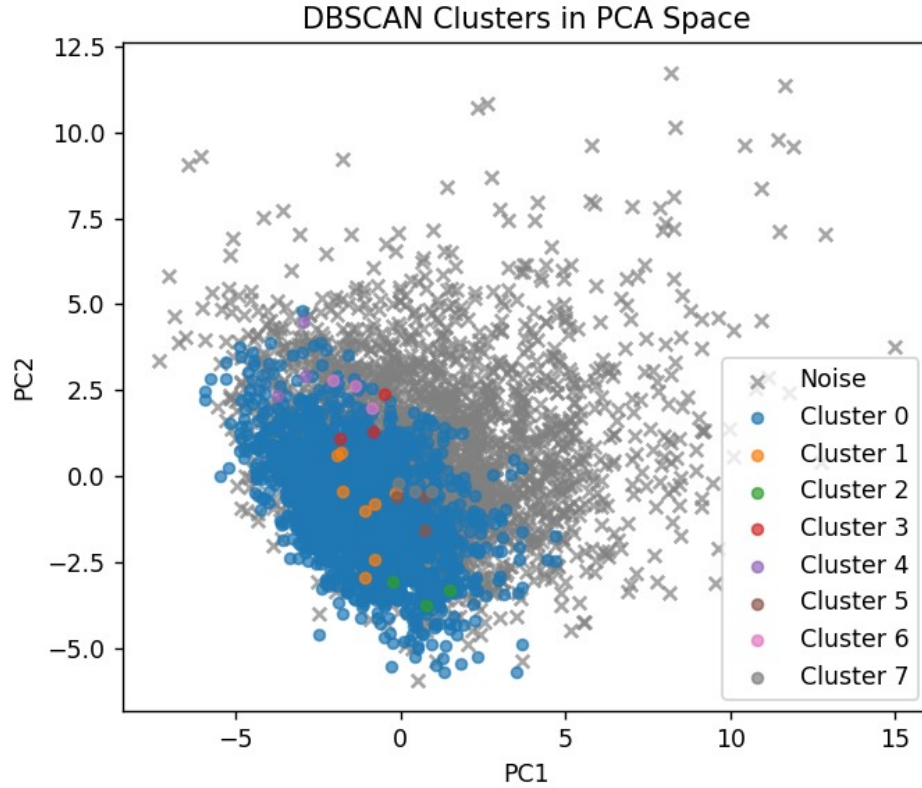


Figure 2.0 Plotted Clusters of DBSCAN, $\epsilon = 5$, $min_points = 3$

Basically what we see is that we achieve 8 clusters and a whole lot of noise what this essentially means which makes sense given the methodology that DBSCAN works with is that DBSCAN captures less general representations and relations inside the data while K-means captures the more general idea of the data.

Note: To plot the graph PCA was used and so the same assumptions of information loss from the section of k-means is relevant here as well.

4.4 Conclusions

To conclude our work in this section, we perform thorough unsupervised learning analysis creating clusters using 2 different algorithms K-means and DBSCAN and plot said clusters using PCA. We achieve optimal hyper parameters of said models according to our own goals and purposes. In the continuation of our work we will use K-means as we feel it represents the more general idea of the data.

It should be noted for future work that using DBSCAN for semi-supervised learning in this case is also a possibility and a more thorough analyzing of DBSCAN was possible but our results from K-means met our satisfaction for this work.

5 Supervised Learning Analysis

In this section we train and test different models mostly tree based models to predict the thirty day expire flag basically predicting the death feature.

5.1 Our Models

Under the work requirements we were required to use XGBoost, Random Forest and Logistic Regression.

5.1.1 XGBoost - Model

XGBoost which stands for eXtreme Gradient Boosting this is a decision tree based model. Gradient boosting is a methodology where we sequentially train decision trees and put weights on other decision trees error and continue to create better and better decision trees and then you make predictions based on the average of all decision tree predictions.

Note: The explanation above is a shallow explanation of boosting and what is XGBoost.

Training

We train the XGBoost model according to the following hyper parameters:

Train test split of 20% – *test*, 80% – *train*

n_estimators = 1000, *max_depth* = 6, *learning_rate* = 0.01, *loss_function* = *logloss*,

n_estimator– This hyper parameter represents the maximum number of decision trees used.

max_depth - Maximum depth of each decision tree(The depth of the decision tree depicts the amount of nodes in the tree in the following way: $num_of_leaves \leq 2^{max_length}$)

learning_rate– Effects the speed of learning of the model, when updating the model it will shrink the contribution of each specific tree (making the prediction more based).

logloss– loss function defined as follows: $Logloss = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{p}_i) + (1 - y_i)(\log(1 - \hat{p}_i))]$, $y_i \in \{0, 1\}$ (*truelabel*), $\hat{p}_i \in [0, 1]$ – predicted probability of positive class.

The definitions above will escort us across all models. Also, the hyper parameters were achieved after different runs of different hyper parameters and reaching the conclusion that those hyper parameters are the optimal ones.

Testing

Post training we get the following results:

Discrimination:

Note: Our main metrics for success are Accuracy, Precision and Recall which are defined as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$$

```
The test's accuracy is: 0.8541666666666666
The test's precision is: 0.8
The test's F1-Score is: 0.4743083003952569
The test's ROC AUC is: 0.8553179438508404
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.86 | 0.98 | 0.92 | 734 |
| 1 | 0.80 | 0.34 | 0.47 | 178 |
| accuracy | | | 0.85 | 912 |
| macro avg | 0.83 | 0.66 | 0.69 | 912 |
| weighted avg | 0.85 | 0.85 | 0.83 | 912 |

Figure 2.1 Metrics Results XGBoost

Our main focus here will be the results of the accuracy, precision and recall. We see that it's especially hard for the model get high recall for predicting the death of patients this is obviously due to to the class imbalance shown in statistical analysis one of our goals will be to improve said recall of predicting deaths.

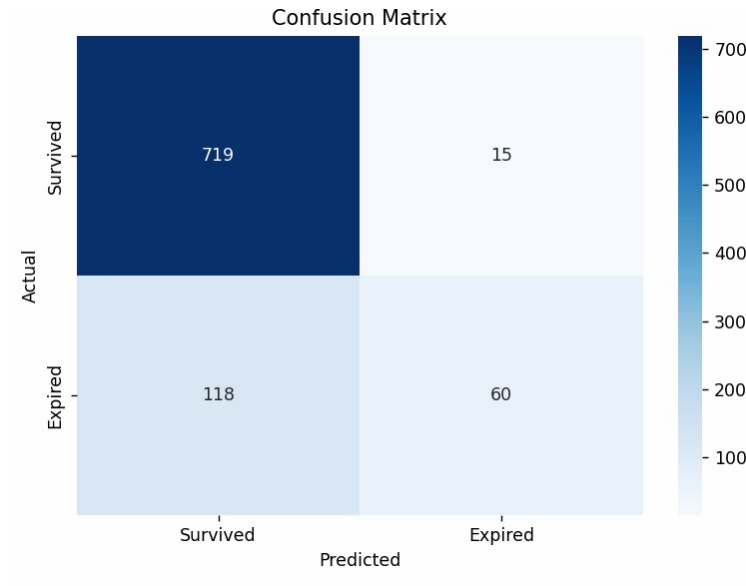


Figure 2.2 Confusion Matrix XGBoost

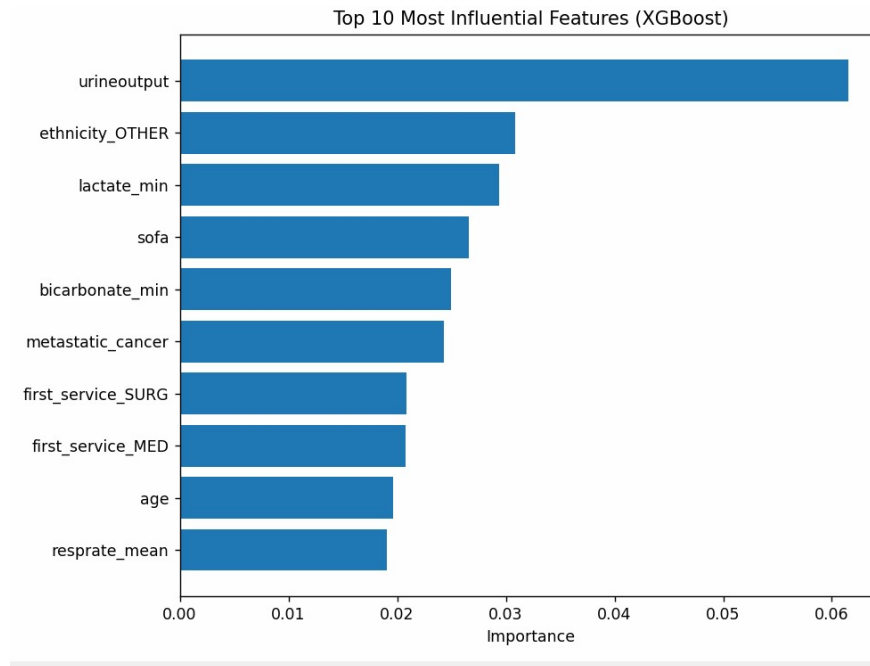


Figure 2.3 Feature Importance XGBoost

We see basically that urineoutput column has high impact on predicting the thirty day flag expire we see could have expected this because when we created our clusters urineoutput had a significant difference between the clusters. We also see that Sofa also has some high importance as well as *lactate_min* which is also expected because in the correlation table in Statistical Analysis we see relatively high correlation between thirty day expire flag and the lactate related features.

Calibration:

When testing for calibration we would like to see if the model for example predicts it's correct 30% of the time is it actually correct 30% of the time and this is the analysis we do here and in the following calibrations of Random Forest and Logistic Regression. We will also use the following methods: (a) Brier

Score (b) Calibration Plot

(a) Brier spot will tell us the accuracy of the probabilistic predictions it's defined in the following way almost like MSE: $BS = \frac{1}{N} \cdot \sum_{i=1}^N (f_i - o_i)^2$ where f_i is the predicted probability and the o_i is the actual outcome also, $BS \in [0, 1]$ and as it approaches 0 it signifies a better score. For XGBoost we achieve a Brier Score of 0.107 which is a good score indicates that XGBoost's predicted probabilities are relatively accurate.

(b) The calibration curve visually show how close the predicted probabilities are to the actual frequencies. $x-axis$ - Average predicted probability $y-axis$ - Actual observed fraction of positives (We haven't explicitly said so but a positive case means a mortality). For the XGBoost we get the following Calibration Curve:

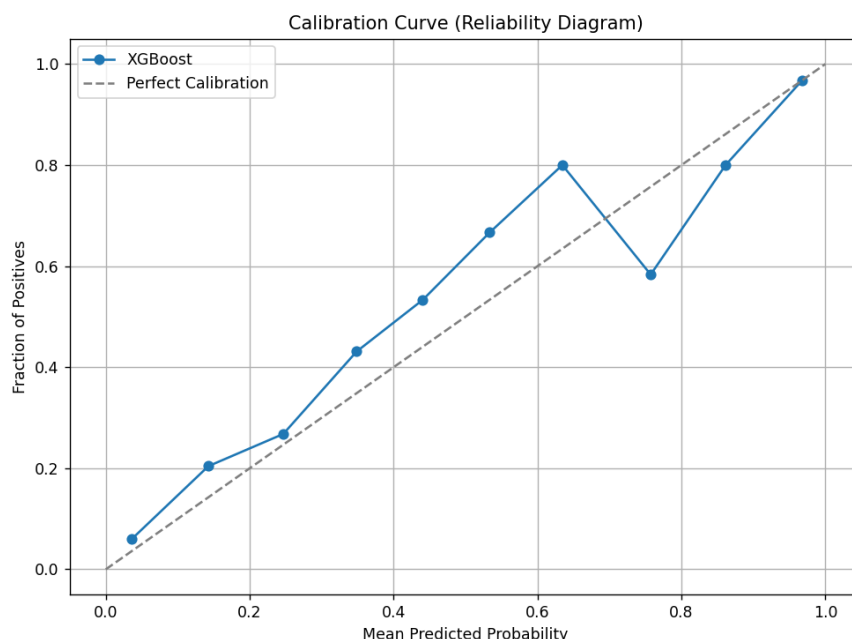


Figure 2.4 Calibration Curve XGBoost

We see that for predictions $\in [0.1, 0.4]$ the XGBoost tends to underestimate the true mortality rate (slight difference) when we go above 0.5 there is also a slight over estimation of the probabilities but overall it seems XGBoost is very well calibrated both by the brier score and by the calibration curve.

5.1.2 Random Forest - Model

Random Forest is again, a decision tree based model. Basically what it does is builds multiple decision trees and merges their results to improve performance thus justifying its name a forest (joke haha). In a categorical case like ours it can be thought as a vote of all trees in the forest and the majority vote is the prediction.

Training

We train the Random Forest model according to the following hyper parameters:

Train test split of 20% – test, 80% – train

$n_estimators = 1000$, $max_depth = 10$, $loss_function = logloss$,

Out of these training parameters and the training of the model we achieve the following results:

Testing

Discrimination


```

Accuracy: 0.8355263157894737
Precision: 0.6666666666666666
F1 Score: 0.42748091603053434
ROC AUC: 0.829960505771056

```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.85 | 0.96 | 0.90 | 734 |
| 1 | 0.67 | 0.31 | 0.43 | 178 |
| accuracy | | | 0.84 | 912 |
| macro avg | 0.76 | 0.64 | 0.67 | 912 |
| weighted avg | 0.82 | 0.84 | 0.81 | 912 |

Figure 2.5 Evaluation Metrics of Random Forest

In this model we also see relatively fine accuracy but with lower precision and even lower recall for the death prediction.

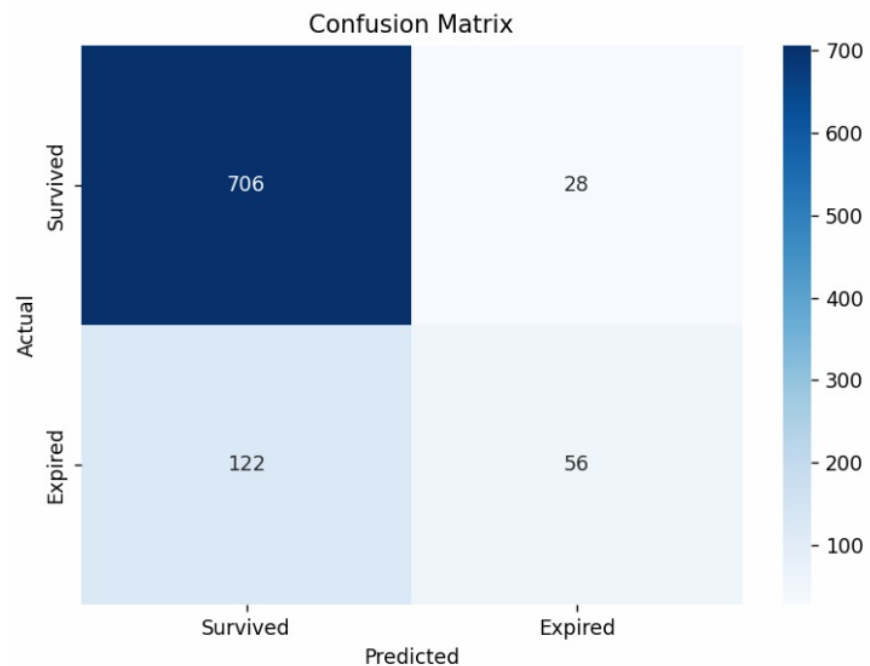


Figure 2.6 Confusion Matrix of Random Forest

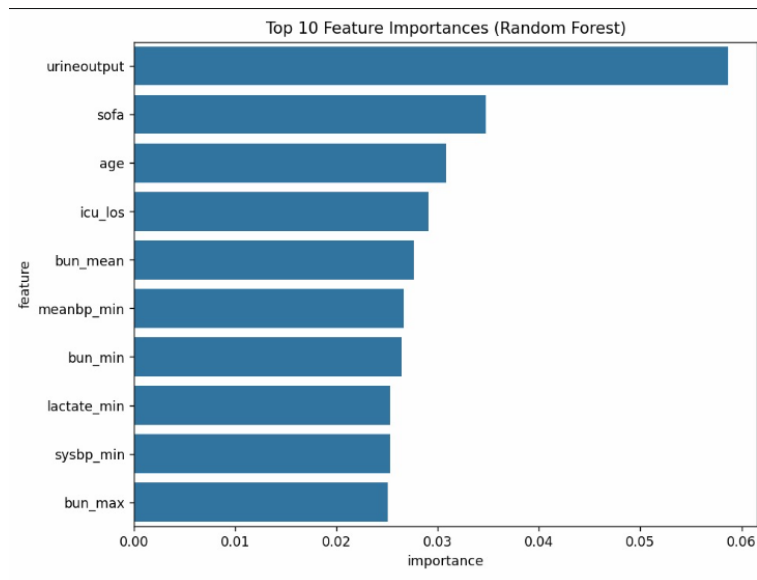


Figure 2.7 Feature Importance of Random Forest

Here we also see that *urineoutput* has significance weight in the random forest model here, *sofa*, *age* and *icu_los* get higher feature importance and contribute more to the finale prediction.

Calibration

For this model we achieve a brier score of 0.123 which also signifies a good calibration. We also achieve the following calibration curve:

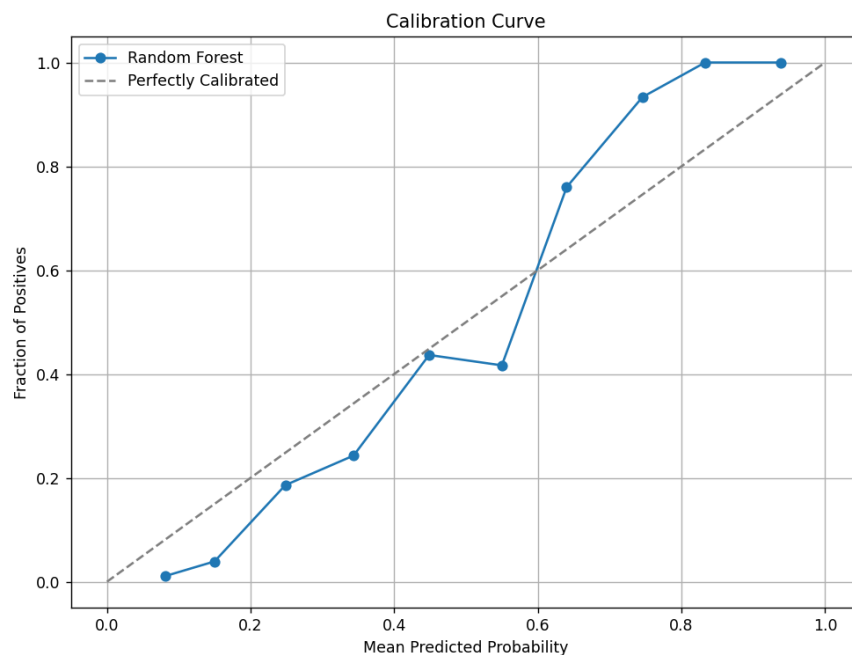


Figure 2.8 Calibration Curve Random Forest

Here we see that in:

1. $[0, 0.4]$ probability range the curve is below the diagonal line indicating that it predicts the probability to be less than it actually is for mortality.

2. (0.4, 0.6] It seems there is some level of uncertainty for the model in those that range indicating the model isn't predicting probabilities in that range very well.
3. [0.7, 1] The model over estimates the probability for a patient to decease.

5.1.3 Logistic Regression - Model

Logistic Regression is a model specifically used for classification thus appropriate for our usage, Logistic Regression essentially starts like linear regression (meaning linear combination of features) then a sigmoid function is applied (sigmoid is defined as: $\hat{p} = \frac{1}{1+e^{-z}}$) and so we get a probability out come which represent the probability of the positive class.

Note: Up till this point no normalization of the data was required since decision tree based models don't require normalization of numerical features but since logistic regression is based on linear regression normalization is needed and so the data used here is normalized.

Training

We train the Logistic Regression model according to the following hyper parameters:

Train test split of 20% – *test*, 80% – *train*

L2 - regularization technique that adds a term to the loss function meaning: $Loss = Logloss + \lambda \sum w_i^2$ this makes sure there aren't exactly large coefficients and thus prevents overfitting.

max_iter = 1000 - Maximum iterations

We add a class imbalance handling which goes as follows: $w_j = \frac{n_{samples}}{n_{classes} \times n_j}$.

After training according to these hyper parameters we achieve the following results:

Testing

Discrimination

| | | | | |
|-------------------------------|-----------|--------|----------|---------|
| Accuracy: 0.7675438596491229 | | | | |
| Precision: 0.4413793103448276 | | | | |
| F1 Score: 0.5470085470085471 | | | | |
| ROC AUC: 0.8380430456479809 | | | | |
| | precision | recall | f1-score | support |
| 0 | 0.92 | 0.78 | 0.84 | 734 |
| 1 | 0.44 | 0.72 | 0.55 | 178 |
| accuracy | | | 0.77 | 912 |
| macro avg | 0.68 | 0.75 | 0.70 | 912 |
| weighted avg | 0.83 | 0.77 | 0.79 | 912 |

Figure 2.9 Metrics for Logistic Regression Test Result

In this model we have an interesting outcome, for once we see that the accuracy and precision is worse than any other model but there is an important exception, unlike any other model we get a high recall for the death classification indicating a better handling of the class imbalance.

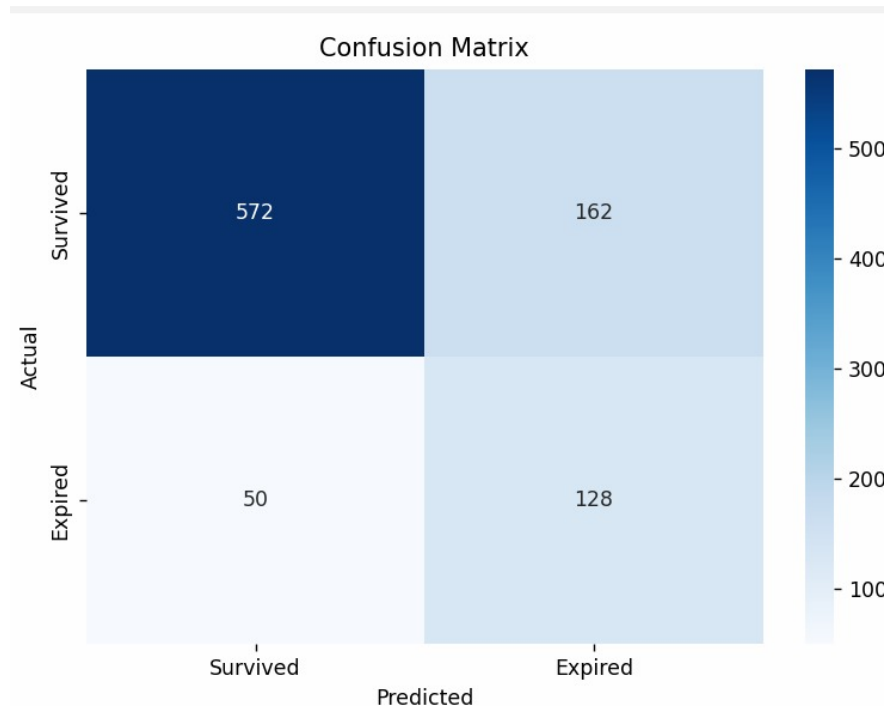


Figure 3.0 Confusion Matrix for Logistic Regression Test Result

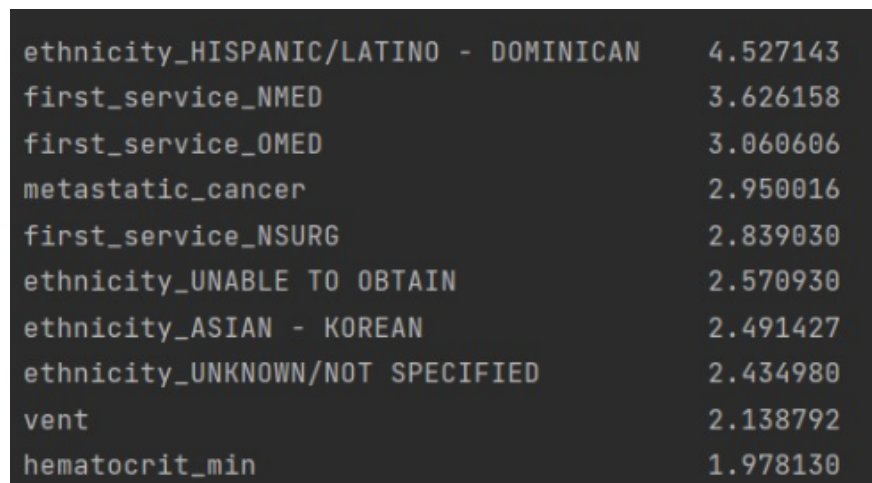


Figure 3.1 Feature Importance for Logistic Regression

Here we also see something interesting but not positive, it seems that the features which lead to the prediction are features which didn't show high correspondence before with predicting or correlating to thirty day expire flag, indicating that the way Logistic Regression handled the class imbalance changed fundamental relationships in the data.

Calibration

For calibration we achieve a brier score of: 0.156 which is also considered a good score in general. We also achieve the following calibration curve:

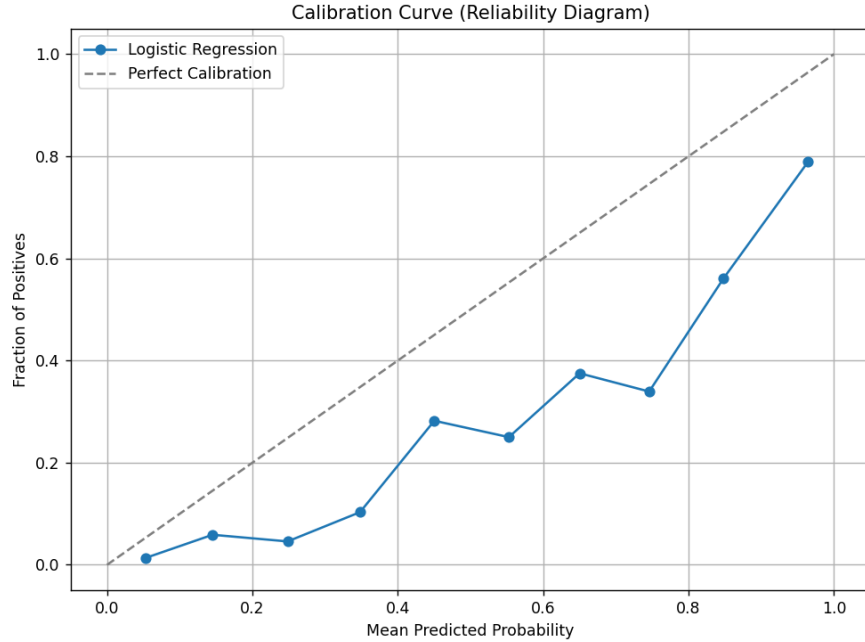


Figure 3.2 Calibration Curve of Logistic Regression

For this calibration plot we see that the curve is under confident in it's probabilistic predictions.

5.2 Model Comparison & Results

When comparing the model we want to mainly take into account in our opinion Accuracy, Precision and Recall we would like to specifically talk also about the class imbalance.

| <i>Class_Model/Metric</i> | <i>Recall</i> | <i>Precision</i> | <i>Accuracy</i> |
|---------------------------|---------------|------------------|-----------------|
| 0_XGBoost | 0.98 | 0.86 | — |
| 1_XGBoost | 0.34 | 0.8 | — |
| Total_XGBoost | 0.66 | 0.83 | 0.854 |
| 0_RF | 0.96 | 0.86 | — |
| 1_RF | 0.31 | 0.67 | — |
| Total_RF | 0.64 | 0.76 | 0.835 |
| 0_LR | 0.78 | 0.92 | — |
| 1_LR | 0.72 | 0.44 | — |
| Total_LR | 0.75 | 0.68 | 0.767 |

Basically what the table means is that, XGBoost is good at identifying people who survived (class 0) but much worse at identifying people who died at the same time when it does predict someone died it's correct for 80% of the time and holds a relatively high accuracy. Random Forest is also very good at identifying people who survived yet it's worse than XGBoost when identifying people who died it's also not as good as being right at predicting someone who died and holds accuracy which is less than XGBoost's. Logistic Regression holds the best record of identifying expired patients but makes more false positives in that class the cost of identifying said expired patients come at the accuracy's expense where it holds the lowest accuracy.

To better sum up the models, XGBoost is the overall best model, Random Forest has strong accuracy but doesn't innovate anything, Logistic Regression while holding a lower overall accuracy holds the best record for identifying deceased patients which is very useful.

The conclusions from the calibration plots are similar, XGBoost out performs the two other models both in the case of the brier score and in the case of the calibration curve. The overall calibration of all models is good but XGBoost's calibration results are the best out of all models and across all ranges of probabilities.

6 Mixing Unsupervised & Supervised Learning

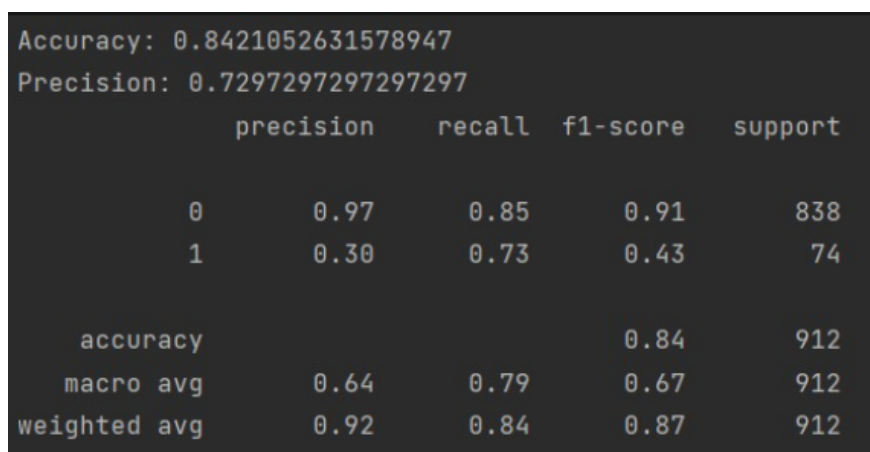
In this section we provide a some what semi-supervised learning methodology, we take the clusters established by k-means in the unsupervised learning part of the project and add them to our dataset. We then train two supervised learning models using the new added feature and achieve new results. We train an XGBoost model and a normal Feed-Forward Neural Network model. We also answer the question of can we use data about clusters to improve the model's ability to predict the thirty day expire flag feature.

6.1 XGBoost

We use the XGBoost as before with the same hyper parameters but here we have a new column essentially a new feature which is the clustering parameter. The results are the following:

Results

We achieve the following results:



```
Accuracy: 0.8421052631578947
Precision: 0.7297297297297297
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.97 | 0.85 | 0.91 | 838 |
| 1 | 0.30 | 0.73 | 0.43 | 74 |
| accuracy | | | 0.84 | 912 |
| macro avg | 0.64 | 0.79 | 0.67 | 912 |
| weighted avg | 0.92 | 0.84 | 0.87 | 912 |

Figure 3.0 Metrics for XGBoost + K-means Column

What we see is that the accuracy actually becomes worse but at the same time the recall improves dramatically, the model improves identifying in people who died but it's worse at the actual prediction of the deceased patients.

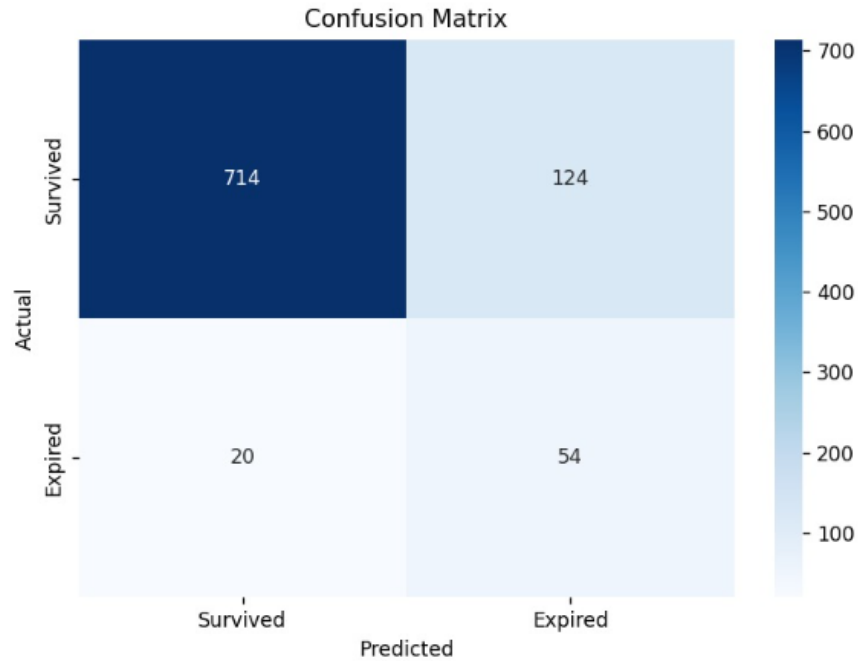


Figure 3.1 Confusion Matrix XGBoost + K-means Column

We now want to examine the feature importance in the model:

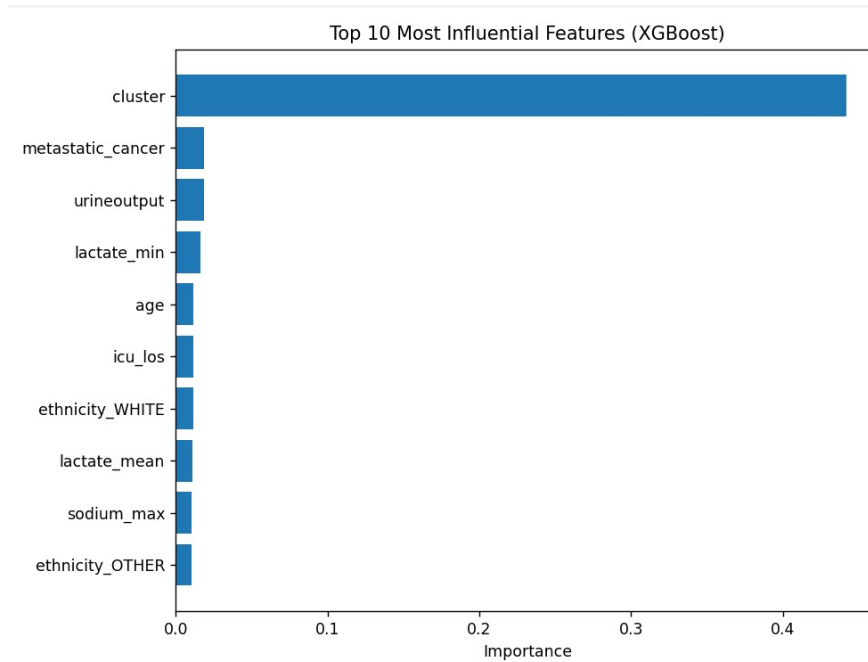


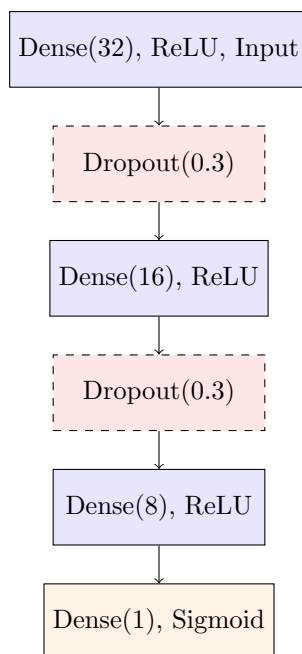
Figure 3.2 Feature Importance XGBoost + K-means Column

What we see here is that the clustering column is overwhelming the model become the main and most significant factor in the prediction process of the XGBoost.

Note: The reason we actually add a FF Neural Network is because a neural network when training gives divides the features in a more even when when training and therefore such a situation where the prediction happens mostly because of one situation is much less likely in a neural network.

6.2 Feed-Forward Neural Network

A neural network is model that works using back propagation, it updates the weights using back propagation iteratively using gradient descent. You can simply say that each edge in the neural network will have a weight and each neuron a bias factor and so it works by linear sums and activation functions such as reLu which adds a non linear functionality into the model. The dense layers of a neural network basically represent a fully connected layer so when we say dense(x) we mean a fully connected layer with x neurons and so our neural network architecture which is based on our trial and error of the model is the following:



Training: We train the neural network model according to the following hyper parameters: train test split of 20% test 16% validation and 64% train data.

We use an adam optimizer and logloss like loss function to train the model. The training process graphs as follows:

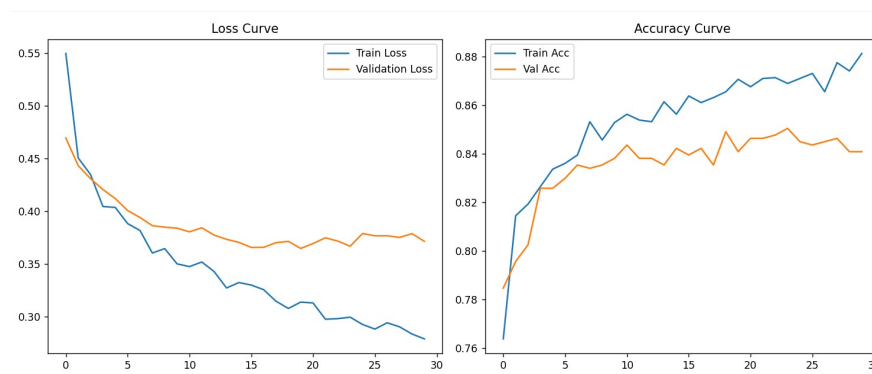


Figure 3.3 Neural Network Convergence

Testing:

After training we achieve the following results:

| Classification Report: | | | | |
|------------------------------|-----------|--------|----------|---------|
| | precision | recall | f1-score | support |
| 0 | 0.87 | 0.96 | 0.91 | 734 |
| 1 | 0.70 | 0.40 | 0.51 | 178 |
| accuracy | | | 0.85 | 912 |
| macro avg | 0.78 | 0.68 | 0.71 | 912 |
| weighted avg | 0.84 | 0.85 | 0.83 | 912 |
| Accuracy: 0.8497807017543859 | | | | |

Figure 3.4 Neural Network Results

What we see here is that the overall results of the neural network are more based and balanced than the XGBoost's results we get a somewhat similar accuracy and it seems like in the general picture it provides a more balanced predictability of the deceased rate of patients.

6.3 Model Comparison

When comparing the results from the two models we see that XGBoost's predictions are overly achieved by the clustering feature but at the same time that it does drastically improve the model's ability to identify deceased patients however at a cost of being right in the detection of said deceased patients. When we compare it to the neural network we see that the results of the neural network are more balanced and seem to have a more balanced outcome of metrics, it does identify deceased people more poorly than the XGBoost but when it does identify them it has a higher success rate at correctly identifying them while still maintaining an identify success rate higher than the XGBoost without the cluster feature.

6.4 The Original Question

In the first part of this section we present a question of whether or not adding an unsupervised learning method to our supervised learning method will help to predict the thirty day expire flag feature. What we're essentially asked is whether creating a hybrid model will improve predicting our target feature. In our case using this method improved the results in some aspects like increasing the recall but decreased the precision and overall accuracy even though by a negligible amount. So to sum up our answer the answer is yes mixing supervised and unsupervised learning into an hybrid model gives additional results that we haven't achieved beforehand.

7 Conclusions & Clinical Applications

Our key findings during this work are the following:

1. XGBoost- XGBoost showed the best overall performance with the highest accuracy and excellent recall for identifying non-deceased patients, it also had strong precision for predicting death, however it had low sensitivity to actual deaths and this may lead to missing a lot of high-risk patients.
2. Logistic Regression - Logistic Regression while being with the lowest accuracy had high sensitivity for identifying risk of death and therefore it can be used to identify high-risk patients better.

We used clustering to identify groups in the data and then used it as a feature, when used as a feature the clustering feature improved drastically the ability of the model to identify deaths but it showed dominance over other features something that may hurt the meaning of the actual prediction. Finally, we can use all of our models and techniques and apply them to the following medical categories:

1. Patient Risk Management - We can use XGBoost as a decision-support tool to classify patients into mortality risks at different scales using their vital signs, lab results, ICU length of stay, sofa score and more.
2. Resource Allocation - We can use the different models and techniques in order to categorize patients into different levels of estimated risk and doing so we can allocate our resources better especially in emergency times.
3. Real-Time-Alerts - The model can be used in the hospital's system constantly tracking patients vital signs, lab results and other features and thus provide real time alert for patients who's vital signs for example have deteriorated and thus their probability of death increases.

8 Summary

To conclude our work, we defined a clear research question grounded in a literature review, then processed and prepared our medical dataset for a range of analyses. We applied statistical methods, supervised learning models, unsupervised clustering algorithms and a hybrid approach, all aimed at predicting 30 day mortality in the ICU patients and identifying meaningful subgroups within our dataset.

Our findings allowed us to draw clinically relevant insights regarding the importance of specific features such as Sofa score, Urine output and ventilation status. We also proposed potential real world applications including patient risk management tool, Resource Allocation decision-tool and Real-Time alert tool, all this with the ultimate goal of improving patient outcomes.

Throughout this project, we gained deep exposure to the structure and complexity of clinical data, we recognized how modern machine learning methods with their capacity to handle large high dimensional datasets can contribute significantly to medical practice. Our main conclusion is that machine learning is not only capable of extracting knowledge from medical data, but holds immense potential as a life-saving tool in clinical decision-making.

9 Bibliography

References

- [1] H. Mayne, G. Parsons, and A. Mahdi, “Unsupervised learning approaches for identifying ICU patient subgroups: Do results generalise?” *arXiv preprint arXiv:2403.02945*, 2024.
- [2] M. K. Lintu, D. R. Micheal, and A. Kamath, “Mortality prediction on unsupervised and semi-supervised clusters of medical intensive care unit patients based on MIMIC-II database,” *Informatics in Medicine Unlocked*, vol. 39, 2023.
- [3] N. Hou *et al.*, “Predicting 30-days mortality for MIMIC-III patients with sepsis-3: a machine learning approach using XGboost,” *Journal of Translational Medicine*, vol. 18, no. 462, 2020.