Foundations of Machine Learning Final Project

Mortality Prediction of Patients in Intensive Care

Yasmina Hobeika
Master's in DSBA
B00804284
yasmina.hobeika@student-cs.fr

Victor Hong
Master's in DSBA
B00802726
victor.hong@student-cs.fr

Ali Najem
Master's in DSBA
B00806315
ali.najem@student-cs.fr

Arindam Roy
Master's in DSBA
B00802759
arindam.roy@student-cs.fr

Introduction

Hospitals, particularly Intensive Care Units (ICU), are essential resources for treating critically ill patients. However, these resources can be scarce and in emergency situations, ICU capacity can become stretched. When this happens, difficult decisions need to be made on how to allocate these resources to the patients who need them most. One such decision that may need to be made is whether to discharge existing patients who have been awarded in lieu of incoming patients who require urgent medical attention.

In emergency situations, it is crucial that hospitals have a system in place to prioritize patients based on their acuity level and likelihood of recovery. The triage process can help to identify patients who are stable enough to be discharged and those who are in urgent need of ICU care. This process can be complex and requires the expertise of a multidisciplinary team, including doctors, nurses, and other healthcare professionals.

In cases where ICU capacity is stretched, it may be necessary to discharge patients who have made significant progress in their recovery and no longer require the level of care provided by the ICU. This can free up beds and resources for incoming patients who are in urgent need of care. However, it is important that patients are only discharged when it is safe to do so and that they are properly transitioned to a lower level of care before they leave the hospital.
In conclusion, resources in hospitals, particularly ICU's, can be scarce and in emergency situations, difficult decisions need to be made on how to allocate them. The triage process is an essential tool to help prioritize patients and ensure that resources are used in the most efficient and effective way possible. At the same time, it is vital that the care of the patient remains the top priority and that decisions are made with their best interest in mind.

Problem Statement

The project that is being proposed aims to utilize data collected from patients in order to predict their chances of survival while being treated in an intensive care unit (ICU). The data collected includes a variety of demographic information, as well as body vital signs and information about any pre-existing medical conditions that the patient may have. This data is collected at hospitals and is continuously monitored throughout the patient's stay in the ICU. By utilizing machine learning models, the goal is to be able to accurately predict the survival rate of patients, based on the information that has been collected.

These applications have the potential to greatly benefit the healthcare industry. One benefit is the ability to prioritize patients based on their predicted survival rate. This could potentially save lives by ensuring that those who are most likely to survive are treated first. Additionally, this project could also be used to make more informed decisions about healthcare infrastructure and hospital operations; if the data shows that certain medical conditions or demographics are more likely to result in a higher survival rate, resources could be allocated accordingly to support such patients. Another potential benefit of this project is its ability to be used as a prognostic tool. By predicting the survival rate of a patient, healthcare professionals can better prepare for potential outcomes and make more informed decisions about the patient's treatment.

Literature Review

First Review: Mortality prediction of patients in intensive care units using machine learning algorithms based on electronic health records (03 May 2022) [1]

This study explored the use of machine learning algorithms in tandem with conventional scoring models and concluded that this approach did provide useful information for predicting the prognosis of critically ill patients in Korea. The authors posited that conventional scoring models (Assessment and Chronic Health Evaluation (APACHE), Simplified Acute Physiology Score (SAPS), Mortality Probability Model (MPM), and Sequential Oran Failure Assessment (SOFA)) are moderately accurate in predicting individual mortality, but vary in results across characteristics of hospitals they are applied (due to varying demographic compositions). Additionally, conventional scoring systems are prone to temporal and social changes as they were evaluated at specific points of time. These shortcomings could be overcome by machine learning algorithms coupled with the advent of abundance of electronic health records to predict patient survivability.

The authors modeled and cross-validated predictions (XGBoost, LightGBM, K-Nearest Neighbours (KNN), Random Forest (RF) and Support Vector Machines (SVM) in two hospitals in Korea based on 70 features capturing demographic features, presence of comorbidities, where missing values were imputed using the median. The results scored high F1-scores for in-sample predictions (>85%) but deteriorated when the samples were switched. Nevertheless, it was possible in both to discern in both samples top predictors that influenced in-hospitality mortality prediction. Another observation was that various ML algorithms, ANN, RF and XGBoost were superior to conventional models in terms of F1 scores.

The authors' conclusion was that in-patient mortality prediction is highly specific to the hospital, and certain models fared better in prediction ability than conventional models, which are better suited to standardized comparisons. The authors also concluded that the results of this study suggest combining conventional scoring models with ML algorithms has practical application, but are only specific to the hospital in question and is limited in effectiveness if generally across other hospitals nationwide, much less globally.

Second Review: Early hospital mortality prediction using vital signals [2]

Early hospital mortality prediction is critical as intensivists strive to make efficient medical decisions about the severely ill patients staying in intensive care units (ICUs). In this paper, they propose a novel method to predict mortality using features extracted from the heart signals of patients within the first hour of ICU admission. In order to predict the risk, quantitative features have been computed based on the heart rate signals of ICU patients suffering cardiovascular diseases. These are the steps:
1- Preprocessing: Remove noise and use FIR filter to interpolate new samples to resample the signals with a lower sampling rate.
2- Feature Extraction: Each signal is described in terms of 12 statistical and signal-based features which were extracted from the patient's ECG signal
3- Classification: They used eight classifiers: decision tree, linear discriminant, logistic regression, support vector machine (SVM), random forest, boosted trees, Gaussian SVM, and K-nearest neighborhood (K-NN). The decision tree may provide the best choice as a tradeoff between transparency and accuracy.
Early hospital risk of mortality prediction in CCU units is critical due to the need for quick and accurate medical decisions. This paper proposes a new signal-based model for early mortality prediction, leveraging the benefits of statistical and signal-based features. Their method is a clinical decision support system which focuses on using only the heart rate signal instead of other health variables such as physical state or presence of chronic diseases.

Third Review: Prediction of mortality in an intensive care unit using logistic regression and a hidden Markov model [3]

This study proposes an algorithm for predicting in-hospital mortality in Intensive Care Unit (ICU) patients that is based on logistic regression and Hidden-Markov model, using commonly available vital signs, laboratory values, and fluid measurements. The algorithm was trained on 4000 ICU patient records and was tested on two sets of 4000 ICU patient records each, obtained from the PhysionNet/CinC Challenge 2012 (prediction of mortality in ICU). Two different metrics, Event1 and Event2, were used to assess the algorithm's performance, and the results were compared to the Simplified Acute Physiology Score (SAPS-I). The proposed algorithm performed better in both Event1 and Event2 score, and since it uses instantaneous values of vital signs and laboratory values, it could be used as a continuous, real-time patient-specific indicator of mortality risk. The proposed algorithm could improve resource allocation and prioritize patient care in ICU.

Fourth Review: Prediction of hospital mortality in intensive care unit patients from clinical and laboratory data: A machine learning approach [4]

The study was conducted to develop a binary classifier for the outcome of death in ICU patients based on clinical and laboratory parameters, formed by 1087 subjects and 50 features. The data was divided into 80% training and 20% testing, and the training set was used to train a model based on the Random Forest (RF) algorithm, and the test set was used to evaluate the predictive effectiveness.

The study concluded that need for intubation, predominant systemic cardiovascular involvement, and hospital death. 8 other factors (e.g. Glasgow Coma Score, mean arterial pressure, temperature, pH), were also significantly associated with death outcomes. The RF algorithm had an accuracy of 80%, positive predictive value of 73.26%, negative predictive value of 84.85% and F1 score of 0.74

The study concluded that the RF algorithm was suited for handling clinical and laboratory data from patients under close monitoring. The study further concluded that machine learning did have great potential in aiding medical staff in predicting hospital mortality and resource prioritization with reasonable predictive accuracy and consistency.

Fifth Review: Prediction model of in-hospital mortality in intensive care unit patients with heart failure: machine learning based, retrospective analysis of the MIMIC-III database [5]

The predictors of in-hospital mortality for intensive care units (ICUs)-admitted heart failure (HF) patients remain poorly characterized. The authors aimed to develop and validate a prediction model for all-cause in-hospital mortality among ICU-admitted HF patients. First, they extracted the data using PostgreSQL. Then, for normally distributed continuous variables, the missing values were replaced with the mean for the patient group. For skewed distributions related to continuous variables, missing values were replaced with their median. Finally, they used XGBoost and Lasso with dimensionality reduction from 52 till 20 features. Using XGBoost, the 52 selected variables were used to identify patients who had died during their hospital stay.

Using bootstrapping validation, the area under the ROC curve values for the XGBoost model and the LASSO regression model were found to be 0.8515 (95% CI 0.7749 to 0.9115) and 0.8646 (95% CI 0.7971 to 0.9201) in the derivation group, respectively, and 0.8029 (95% CI 0.6849 to 0.9030) and 0.8194 (95% CI 0.7201 to 0.9205) in the validation group, respectively. With a high AUC of 0.8416 (95% CI 0.7864 to 0.8967) and a wide net benefit threshold range (>0.1), this nomogram can be widely used to enhance more accurate clinical decision-making.

Group Model, Methodology, and Algorithm

Overview:

We have data collected from each patient with features based on demographics such as age and sex and including medical conditions: Body vitals (blood pressure, temperature, SO2 levels...) and disease history (diabetes, infections, heart failure...). This data is collected at hospitals and monitored throughout patients' stay at the ICU. By applying different machine learning models, we are able to better predict the outcomes and severity of the situation of individual patients.

This project will be divided into three phases for the model implementation. The first being the EDA phase, where the aim will be to find remarkable features of the data. The second phase of this project will be data cleaning, where the aim will be to remove unnecessary or highly correlated features. This phase may also include feature engineering, SVD and PCA steps. The final step of the model implementation will be to test various ML models, such as linear regression, logistic regression, SVM, KNN, decision trees, random forest, et cetera. The model that performs the best in evaluation will be selected.

Procedure:

Stage 1 - Data Pre-processing:

1. First features (referring to columns in the dataset) were examined for null values. Features containing numerical entries, but possessing null values were imputed with the median of that category.
2. Columns containing features that we deemed would not contribute to the predictive ability of

patients' survivability (such as id of patients and hospitals) were removed from the dataset.

3. Similarly, features containing categorical entries, but possessing null values were imputed with the mode for that category.
4. Categorial features were encoded using pd.get_dummies. These were employed for the columns of 'ethnicity', 'gender', 'icu_admit_source', 'icu_stay_type', 'icu_type' and 'apache_2_bodysystem'. Once these columns were encoded, the original source columns were removed since these columns could not be operated on by models.
5. A 'life_or_death' column containing the actual survivability ['0' = survive, '1' = death] was encoded to indicate actual patient survivability.

Stage 2 - Feature Engineering:

We replaced the categorical variables with their count of occurrences in the dataset, which means counting the number of times each category appears in the dataset and encoding it as a numerical value, using CountEncoder. Then, we used RobustScaler: it calculates the median and the interquartile range (IQR) for each feature and then scales the feature values by subtracting the median and dividing by the IQR. This makes the RobustScaler less sensitive to outliers than the StandardScaler, which uses mean and standard deviation to scale the data.

Stage 3 - All Models and Results

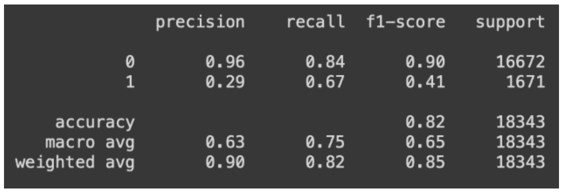
The data was trained on 80% of the data and validated for the remaining 20% of the data for all models. We generated a classification report to get all the metrics, in particular:

- Precision of 1s, which determines the ratio of the number of patients who were correctly predicted to die in the hospital by the model over the total number of patients predicted to die in the hospital by the model.
- Precision of 0s, which determines the ratio of the number of patients who were correctly predicted to survive in the hospital by the model over the total number of patients predicted to survive in the hospital by the model.
- Recall of 1s, which determines the ratio of the number of patients who were correctly predicted to die in the hospital by the model over the actual number of patients who died in hospital
- Recall of 0s, which determines the ratio of the number of patients who were correctly predicted to survive in the hospital by the model over the actual number of patients who survived in hospital

Without Dimensionality Reduction

Naive Bayes Model

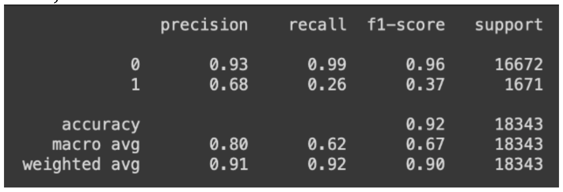
A base Naïve-Bayes Model with no changes to parameters was used, with the above results.



```
              precision    recall  f1-score   support

           0       0.96      0.84      0.90     16672
           1       0.29      0.67      0.41      1671

    accuracy                           0.82     18343
   macro avg       0.63      0.75      0.65     18343
weighted avg       0.90      0.82      0.85     18343
```

Logistic Regression

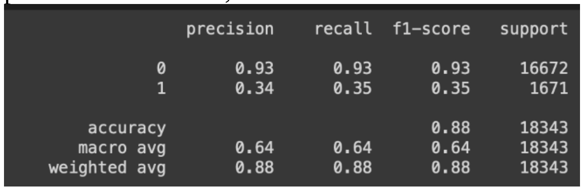
A base Logistic Regression with iterations set to 300 was used, with the above results.



```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     16672
           1       0.68      0.26      0.37      1671

    accuracy                           0.92     18343
   macro avg       0.80      0.62      0.67     18343
weighted avg       0.91      0.92      0.90     18343
```

DecisionTree

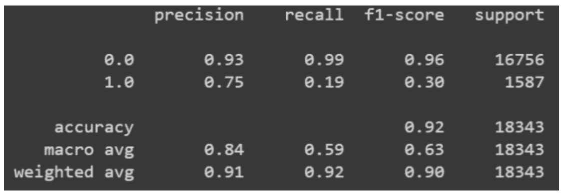
A base Decision Tree Model with no changes to parameters was used, with the above results



```
              precision    recall  f1-score   support

           0       0.93      0.93      0.93     16672
           1       0.34      0.35      0.35      1671

    accuracy                           0.88     18343
   macro avg       0.64      0.64      0.64     18343
weighted avg       0.88      0.88      0.88     18343
```

K-Nearest Neighbors

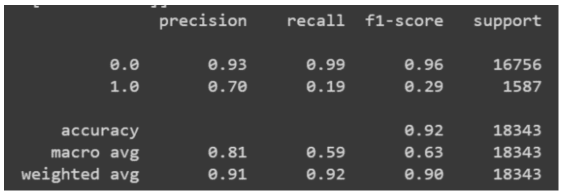
For hyperparameter tuning, a randomized search CV was used and the best parameter of n_neighbors = 27 was chosen for the final result. Figure 1 shows the metrics for the 27-Nearest Neighbors.



```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.75      0.19      0.30      1587

    accuracy                           0.92     18343
   macro avg       0.84      0.59      0.63     18343
weighted avg       0.91      0.92      0.90     18343
```

Linear SVM

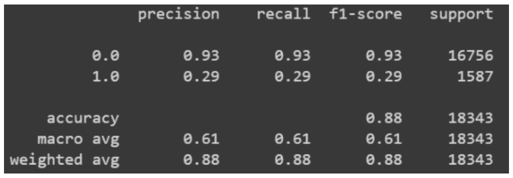
For hyperparameter tuning, randomized search CV was used and the penalty L1, dual False and C = 0.05 were chosen for the final result. Figure 2 shows the metrics for the Linear SVM.



```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.70      0.19      0.29      1587

    accuracy                           0.92     18343
   macro avg       0.81      0.59      0.63     18343
weighted avg       0.91      0.92      0.90     18343
```

SVM

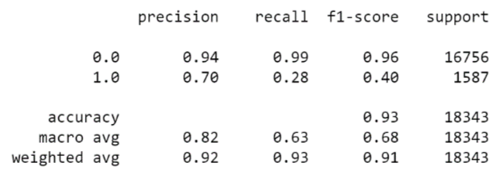
For hyperparameter tuning, a randomized search CV was used, we tested these kernels: poly, rbf and sigmoid and the best result was sigmoid with C = 0.7. Figure 3 shows the metrics for the SVM.



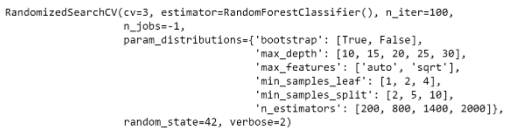
```
              precision    recall  f1-score   support

         0.0       0.93      0.93      0.93     16756
         1.0       0.29      0.29      0.29      1587

    accuracy                           0.88     18343
   macro avg       0.61      0.61      0.61     18343
weighted avg       0.88      0.88      0.88     18343
```

Random forest classifier

For hyperparameter tuning, randomized search CV was used with and the best parameters were chosen for the final result. Figure 4 shows the metrics for the random forest classifier and the hyperparameters and best parameters are as shown in Figure 4.1 and Figure 4.2.



```
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96     16756
         1.0       0.70      0.28      0.40      1587

    accuracy                           0.93     18343
   macro avg       0.82      0.63      0.68     18343
weighted avg       0.92      0.93      0.91     18343
```

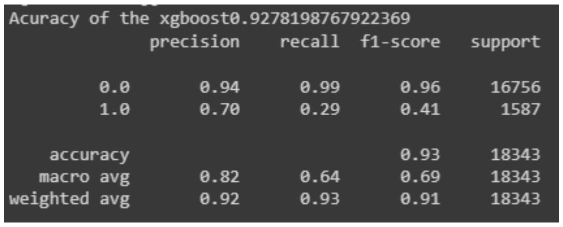


```
RandomizedSearchCV(cv=3, estimator=RandomForestClassifier(), n_iter=100,
                   n_jobs=-1,
                   param_distributions={'bootstrap': [True, False],
                                        'max_depth': [10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [200, 800, 1400, 2000]},
                   random_state=42, verbose=2)
```

XGboost

For hyperparameter tuning, a GridsearchCV was used and the final and best result had the following parameters:

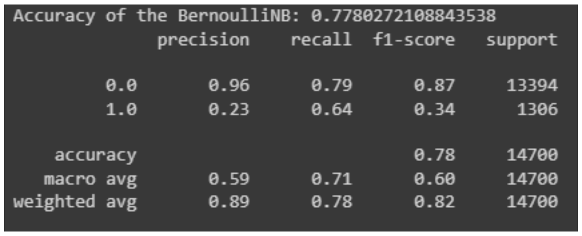
seed = 20, objective = 'multi:softmax', max_depth = 6, learning_rate = 0.1, subsample = 0.7, colsample_bytree = 0.2, colsample_bylevel = 0.5, n_estimators = 100, num_class = 2



```
Acuracy of the xgboost0.9278198767922369
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96     16756
         1.0       0.70      0.29      0.41      1587

    accuracy                           0.93     18343
   macro avg       0.82      0.64      0.69     18343
weighted avg       0.92      0.93      0.91     18343
```

BernoulliNB

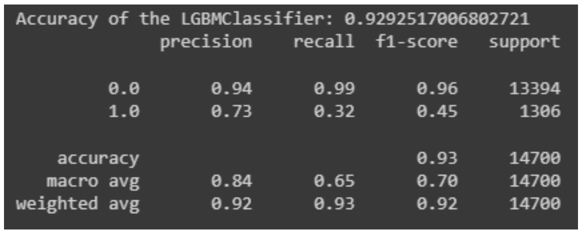
We did not use hyperparamter tuning as the default model got the highest accuracy in prediction, as seen in the results below.



```
Accuracy of the BernoulliNB: 0.7780272108843538
              precision    recall  f1-score   support

         0.0       0.96      0.79      0.87     13394
         1.0       0.23      0.64      0.34      1306

    accuracy                           0.78     14700
   macro avg       0.59      0.71      0.60     14700
weighted avg       0.89      0.78      0.82     14700
```

LightGBM

We applied hyperparameter tuning using gridsearchCV, and found the following parameters to yield the best result in accuracy: learning_rate=0.01, n_estimators=1000, max_depth=9.



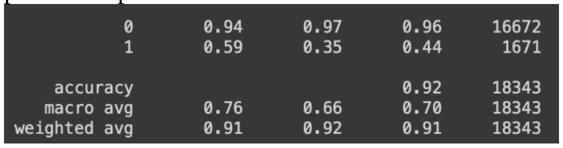
```
Accuracy of the LGBMClassifier: 0.9292517006802721
              precision    recall  f1-score   support

         0.0       0.94      0.99      0.96     13394
         1.0       0.73      0.32      0.45      1306

    accuracy                           0.93     14700
   macro avg       0.84      0.65      0.70     14700
weighted avg       0.92      0.93      0.92     14700
```

With LDA

LDA is a supervised dimensionality reduction technique that aims to find the directions in the feature space that maximizes the separation between different classes. n_components=1 is passed as an argument to specify that the feature space should be reduced to one dimension. The fit_transform method is called on the training data, X_train, and the target variable, y_train, to fit the LDA model and transform the training data to the new feature space. Finally, the transform method is called on the test data, X_test, to apply the LDA transformation to the test data.

<u>Naive-Bayes with LDA:</u>

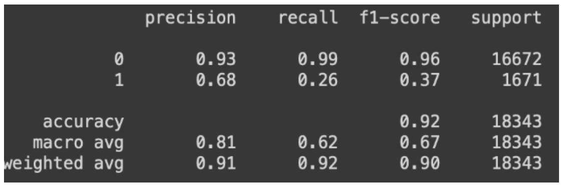
Post-LDA implementation, Naïve-Bayes model demonstrated marginal improvement in accuracy and predictive power of the model.



```
           0       0.94      0.97      0.96     16672
           1       0.59      0.35      0.44      1671

    accuracy                           0.92     18343
   macro avg       0.76      0.66      0.70     18343
weighted avg       0.91      0.92      0.91     18343
```

<u>Logistic Regression with LDA:</u>

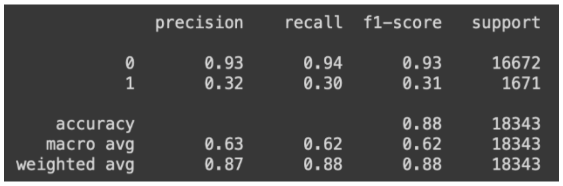
Post-LDA implementation, Logistic Regression model demonstrated no improvement in accuracy and explainability of the model.



```
              precision    recall  f1-score   support

           0       0.93      0.99      0.96     16672
           1       0.68      0.26      0.37      1671

    accuracy                           0.92     18343
   macro avg       0.81      0.62      0.67     18343
eighted avg       0.91      0.92      0.90     18343
```

<u>Decision-Tree with LDA:</u>

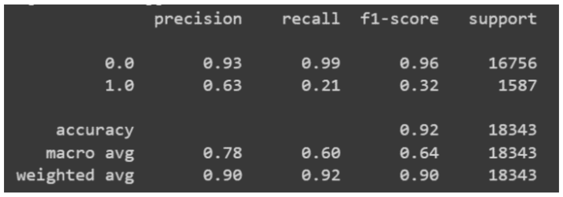
Post-LDA implementation, Decision Tree model demonstrated no improvement in accuracy and explainability of the model.



```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93     16672
           1       0.32      0.30      0.31      1671

    accuracy                           0.88     18343
   macro avg       0.63      0.62      0.62     18343
weighted avg       0.87      0.88      0.88     18343
```

<u>K-Nearest Neighbors LDA</u>

For hyperparameter tuning, a randomized search CV was used and the best parameter of n_neighbors = 24 was chosen for the final result. Figure 1 shows the metrics for the 24-Nearest Neighbors.

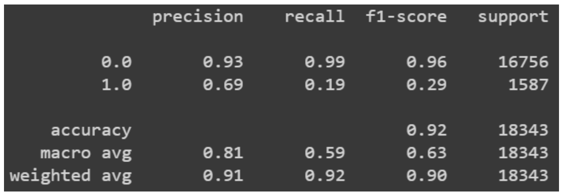


```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.63      0.21      0.32      1587

    accuracy                           0.92     18343
   macro avg       0.78      0.60      0.64     18343
weighted avg       0.90      0.92      0.90     18343
```

Classification report of KNN24

<u>Linear SVM</u>

For hyperparameter tuning, randomized search CV was used and the penalty L1, dual False and C = 1 were chosen for the final result. Figure 2 shows the metrics for the Linear SVM.

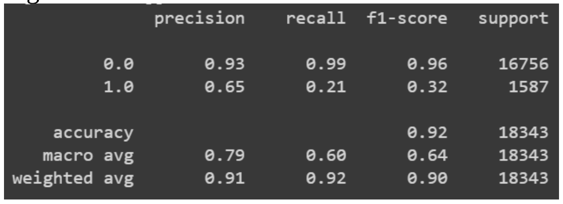


```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.69      0.19      0.29      1587

    accuracy                           0.92     18343
   macro avg       0.81      0.59      0.63     18343
weighted avg       0.91      0.92      0.90     18343
```

Classification report of Linear SVM

<u>SVM</u>

For hyperparameter tuning, a randomized search CV was used, we tested these kernels: poly, rbf and sigmoid and the best result was rbf with C = 0.5. Figure 3 shows the metrics for the SVM.

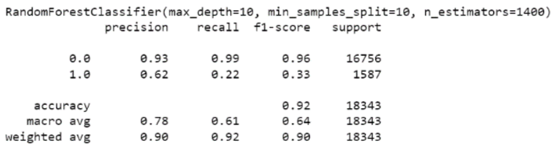


```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.65      0.21      0.32      1587

    accuracy                           0.92     18343
   macro avg       0.79      0.60      0.64     18343
weighted avg       0.91      0.92      0.90     18343
```

Classification report of SVM with sigmoid kernel

<u>Random forest classifier</u>

For hyperparameter tuning, randomized search CV was used with, and the best parameters were chosen for the final result. Figure 5 shows the metrics for the random forest classifier and the hyperparameters we got.

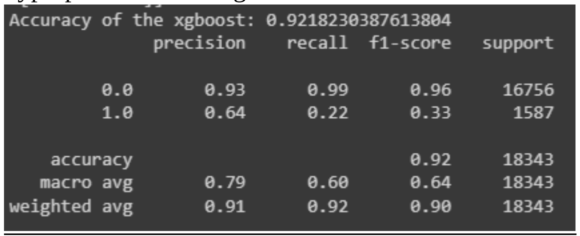


```
RandomForestClassifier(max_depth=10, min_samples_split=10, n_estimators=1400)
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.62      0.22      0.33      1587

    accuracy                           0.92     18343
   macro avg       0.78      0.61      0.64     18343
weighted avg       0.90      0.92      0.90     18343
```

Best parameters with randomized search and Classification report of random forest with LDA data

XGB with LDA

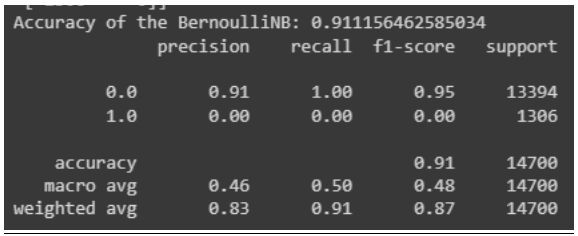
With LDA, our model’s recall slightly decreased, but the general accuracy was maintained high with hyperparameter tuning.



```
Accuracy of the xgboost: 0.9218230387613804
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.64      0.22      0.33      1587

    accuracy                           0.92     18343
   macro avg       0.79      0.60      0.64     18343
weighted avg       0.91      0.92      0.90     18343
```

BernoulliNB with LDA

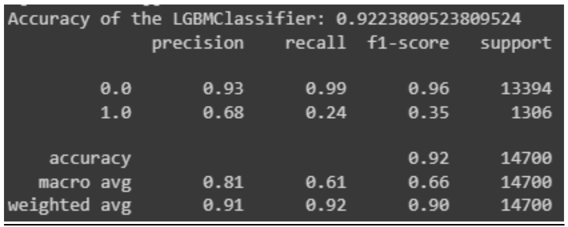
The recall for this model did not yield any result and maintained a value of 0, se we ruled out that Bernoulli on its own had the best results. However, the accuracy was still good considerably above 0.9.



```
Accuracy of the BernoulliNB: 0.911156462585034
              precision    recall  f1-score   support

         0.0       0.91      1.00      0.95     13394
         1.0       0.00      0.00      0.00      1306

    accuracy                           0.91     14700
   macro avg       0.46      0.50      0.48     14700
weighted avg       0.83      0.91      0.87     14700
```

LightGBM with LDA

With LDA, the model’s recall also slightly decreased, but the model accuracy for predicting 1’s and 0’s maintained their good results.



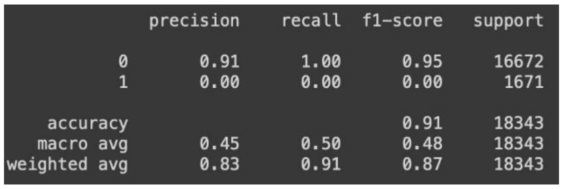
```
Accuracy of the LGBMClassifier: 0.9223809523809524
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     13394
         1.0       0.68      0.24      0.35      1306

    accuracy                           0.92     14700
   macro avg       0.81      0.61      0.66     14700
weighted avg       0.91      0.92      0.90     14700
```

With PCA

PCA is an unsupervised technique that aims to identify the directions of maximum variance in the feature space and project the data into a lower-dimensional space while preserving as much variance as possible. n_components=0.7 is passed as an argument to specify that the feature space should be reduced such that at least 70% of the variance is retained. Then, the fit_transform method is called on the training data, X_train, to fit the PCA model and transform the training data to the new feature space. It is important to note that PCA is an unsupervised dimensionality reduction technique, so it doesn't require the target variable. Finally, the transform method is called on the test data, X_test, to apply the PCA transformation to the test data. It is worth noting that PCA may not always yield optimal results and the value of n_components should be experimentally determined with cross-validation to minimize any loss of information while reducing the dimensionality. We applied grid Search CV and determined that n_components = 0.7 was the resulting hyperparameter.

Naive-Bayes with PCA

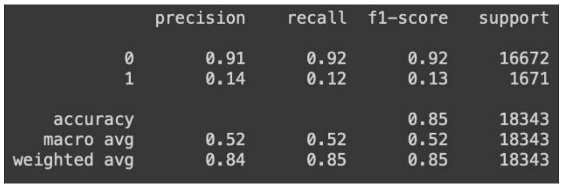
Post-PCA implementation, prediction results under Naïve-Bayes were inconclusive for “1”s, and also a deterioration of F1-score for “1”’s.



```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     16672
           1       0.00      0.00      0.00      1671

    accuracy                           0.91     18343
   macro avg       0.45      0.50      0.48     18343
weighted avg       0.83      0.91      0.87     18343
```

Logistic Regression with PCA

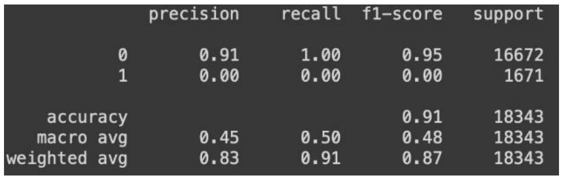
Post-PCA implementation, prediction results under Logistic Regression were inconclusive for “1”s, and also a deterioration of F1-score for “1”’s.



```
              precision    recall  f1-score   support

           0       0.91      0.92      0.92     16672
           1       0.14      0.12      0.13      1671

    accuracy                           0.85     18343
   macro avg       0.52      0.52      0.52     18343
weighted avg       0.84      0.85      0.85     18343
```

Decision Tree with PCA

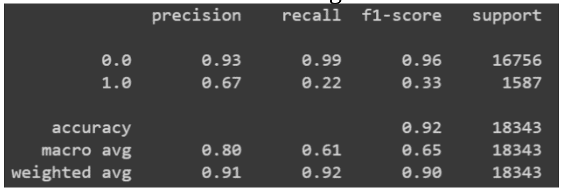
Post-PCA implementation, prediction results under Decision Tree Model were inconclusive for “1”s, and also a deterioration of F1-score for “1”’s.



```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     16672
           1       0.00      0.00      0.00      1671

    accuracy                           0.91     18343
   macro avg       0.45      0.50      0.48     18343
weighted avg       0.83      0.91      0.87     18343
```

K-Nearest Neighbors

For hyperparameter tuning, a randomized search CV was used and the best parameter of n_neighbors = 29 was chosen for the final result. Figure 1 shows the metrics for the 29-Nearest Neighbors.

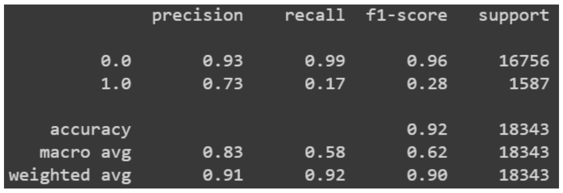


```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.67      0.22      0.33      1587

    accuracy                           0.92     18343
   macro avg       0.80      0.61      0.65     18343
weighted avg       0.91      0.92      0.90     18343
```

Classification report of KNN29

<u>Linear SVM</u>

For hyperparameter tuning, randomized search CV was used and the penalty L1, dual False and C = 1 were chosen for the final result. Figure 2 shows the metrics for the Linear SVM.

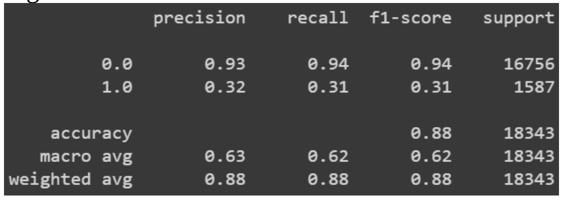


```
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.73      0.17      0.28      1587

    accuracy                           0.92     18343
   macro avg       0.83      0.58      0.62     18343
weighted avg       0.91      0.92      0.90     18343
```

Classification report of Linear SVM

<u>SVM</u>

For hyperparameter tuning, a randomized search CV was used, we tested these kernels: poly, rbf and sigmoid and the best result was sigmoid with C = 0.7. Figure 3 shows the metrics for the SVM.

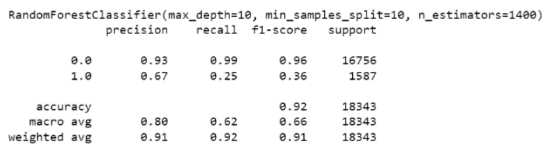


```
              precision    recall  f1-score   support

         0.0       0.93      0.94      0.94     16756
         1.0       0.32      0.31      0.31      1587

    accuracy                           0.88     18343
   macro avg       0.63      0.62      0.62     18343
weighted avg       0.88      0.88      0.88     18343
```

Classification report of SVM with sigmoid kernel

<u>Random forest classifier</u>

For hyperparameter tuning, randomized search CV was used with and the best parameters were chosen for the final result. Figure 4 shows the metrics for the random forest classifier and the hyperparameters we got.

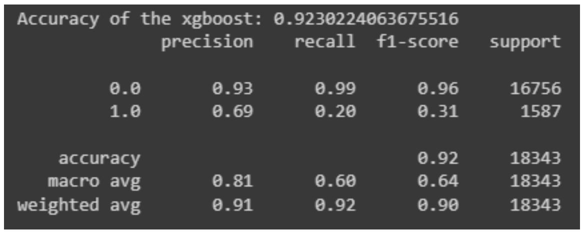


```
RandomForestClassifier(max_depth=10, min_samples_split=10, n_estimators=1400)
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.67      0.25      0.36      1587

    accuracy                           0.92     18343
   macro avg       0.80      0.62      0.66     18343
weighted avg       0.91      0.92      0.91     18343
```

Best parameters with randomized search and Classification report of random forest with PCA data

<u>XGB with PCA</u>

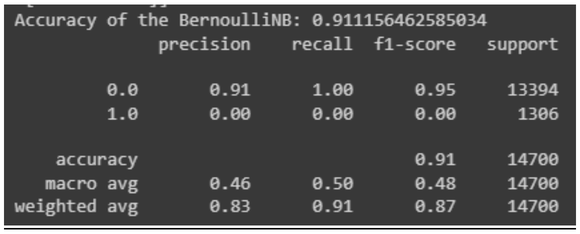
With hyperparameter tuning, we can say that the case of XGB with PCA is similar to LDA in results, only varying slightly in recall between 0.22 for LDA and 0.2 for PCA.



```
Accuracy of the xgboost: 0.9230224063675516
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     16756
         1.0       0.69      0.20      0.31      1587

    accuracy                           0.92     18343
   macro avg       0.81      0.60      0.64     18343
weighted avg       0.91      0.92      0.90     18343
```

<u>BernoulliNB with PCA</u>

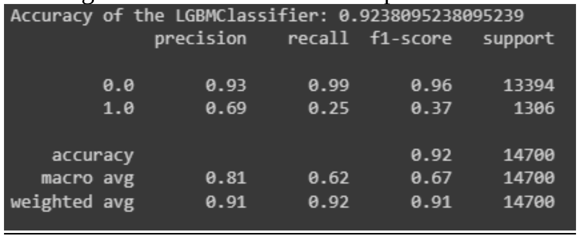
The case here is the same as for LDA, where the recall was 0 but model accuracy maintained a standing above 0.9.



```
Accuracy of the BernoulliNB: 0.911156462585034
              precision    recall  f1-score   support

         0.0       0.91      1.00      0.95     13394
         1.0       0.00      0.00      0.00      1306

    accuracy                           0.91     14700
   macro avg       0.46      0.50      0.48     14700
weighted avg       0.83      0.91      0.87     14700
```

<u>LightGBM with PCA</u>

After hyperparameter tuning, the recall was almost the same, as for the precision at accuracy, with a good stabding overall for this model compared to others.



```
Accuracy of the LGBMClassifier: 0.9238095238095239
              precision    recall  f1-score   support

         0.0       0.93      0.99      0.96     13394
         1.0       0.69      0.25      0.37      1306

    accuracy                           0.92     14700
   macro avg       0.81      0.62      0.67     14700
weighted avg       0.91      0.92      0.91     14700
```

Evaluation/Results

A high recall score means that the model has a low false negative rate. False negative is the number of patients who were incorrectly predicted as a fatality in the hospital by the model. This case should be considered carefully as the number of true fatalities might drastically change by tuning this parameter. So, our final model is the one with the lowest false negative rate, which means highest recall of 1s. First, we will determine, for each model, which case (without dimensionality reduction, with PCA or with LDA) is the best in terms of recall of 1's. Then, we will pick the best result among all the models and choose it as our final model.

Model	***Recall of survival for complete Data***	***Recall of survival for PCA data***	***Recall of survival for LDA data***	***Highest Accuracy***
NB	*0.67*	*0*		*0.82*
Bernoulli NB	*0.64*	*0*	*0*	*0.78*
Decision Tree	*0.35*	*0*		*0.88*
Light gbm	*0.32*	*0.25*	*0.24*	*0.91*
SVM	*0.29*	*0.31*	*0.21*	*0.88*
XGB	*0.29*	*0.2*	*0.22*	*0.93*
Random Forest	*0.28*	*0.25*	*0.22*	*0.93*
Logistic Regression	*0.26*	*0.12*		*0.92*
KNN	*0.19*	*0.22*	*0.21*	*0.92*
Linear SVM		*0.17*	*0.19*	*0.92*

Figure 1: Table of recall metric on survival for all models and accuracy

While Naïve bayes has a very high score for survival recall, we find that the accuracy metric is low for this model. Thus, setting a threshold for the accuracy metric is important. Setting the threshold at 90% and above for the accuracy metric, we find that Light GBM is the best suited model for our problem statement.

Conclusion

Survival detection in ICU can be an extremely useful tool. While implementing any such model, there should be various components that should be taken into consideration. After discussions, this project team has found the metrics of survival recall and accuracy to be of extreme importance. This is because, reducing the false negative cases will result in the greatest number of survivors. At the same time, low accuracy models should not be implemented, as any such models will result in poor predictions. After comparing ten models on a high dimensional data and applying various dimensionality reduction techniques, Light GBM is the model that holds the best potential for this use case.

References

[0] M. Agarwal, "Patient survival prediction," Kaggle, 26-Dec- 2021. [Online]. Available: https://www.kaggle.com/datasets/mitishaagarwal/patient.

[1] M. H. Choi, D. Kim, E. J. Choi, Y. J. Jung, Y. J. Choi, J. H. Cho, and S. H. Jeong, "Mortality prediction of patients in intensive care units using machine learning algorithms based on Electronic Health Records," Nature News, 03-May-2022. [Online]. Available: https://www.nature.com/articles/s41598-022- 11226-4.

[2] R. Sadeghi, T. Banerjee, and W. Romine, "Early hospital mortality prediction using vital signals," Smart Health, 06-Jul- 2018. [Online]. Available:https://www.sciencedirect.com/science/article/abs/pii/S235264831 8300357.

[3] "Prediction of mortality in an intensive care unit using logistic regression and a hidden Markov model," IEEE Xplore. [Online]. Available: https://ieeexplore.ieee.org/document/6420413.

[4] E. Caires Silveira, S. Mattos Pretti, B. A. Santos, C. F. Santos Corrêa, L. Madureira Silva, and F. Freire de Melo, "Prediction of hospital mortality in intensive care unit patients from clinical and laboratory data: A machine learning approach," World journal of critical care medicine, 09-Sep-2022. [Online]. Available:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9483004/.

[5] F. Li, H. Xin, J. Zhang, M. Fu, J. Zhou, and Z. Lian, "Prediction model of in-hospital mortality in intensive care unit patients with heart failure: Machine Learning-based, retrospective analysis of the mimic-III database," BMJ Open, 01-Jul-2021. [Online]. Available:https://bmjopen.bmj.com/content/11/7/e044779.info