Heterogeneous Ensemble Models for In-Hospital Mortality Prediction

Mattyws F. Grawe, Viviane P. Moreira

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{mfgrawe,viviane}@inf.ufrgs.br

Abstract. *Electronic Health Records data are rich and contain different types of variables, including structured data (e.g., demographics), free text (e.g., medical notes), and time series data. In this work, we explore the use of these different types of data for the task of in-hospital mortality prediction, which seeks to predict the outcome of death for patients admitted to the hospital. We build base learning models for the different data types and combine them in a heterogeneous ensemble model. In these models, we apply state-of-the-art classification algorithms based on deep learning. Our experiments on a set of 20K ICU patients from the MIMIC-III dataset showed that the ensemble method brings improvements of 3 percentage points, achieving an AUROC of 0.853.*

1. Introduction

Electronic Health Records (EHR) data have extensively been used in the development of machine learning models for several tasks within the medical field and pose a set of challenges for the machine learning area. These challenges stem from various reasons, including the temporal and sparse nature of the data, and the heterogeneity of data types that are generated by a patient's hospital admission. Data about a patient can be structured text (*e.g.,* demographics), free text (*e.g.,* a description of the patient's evolution), numerical (*e.g.,* vital signs), or even sound, image, and video. The volume and complexity of the data increase even more if the patient needs to be admitted into an Intensive Care Unit (ICU). These different types of data can be used to help understand the patient's current condition and create means to foresee the most likely outcomes for the patient.

In this work, we explore in-hospital mortality prediction, *i.e.,* the task of assessing whether a patient is likely to die during the course of a hospital stay. Such a prediction should be made preferably using the data from the first hours of the patient's admission into the hospital or ICU. Predicting a possible outcome of a patient in the early stages of admission is important to give the health professionals time to take an adequate course of action to treat the patient properly – thus our prediction is based on data available in *the first 48 hours* since the time that patients were admitted. More specifically, our analysis is on *mortality prediction of ICU patients*. We model this problem as a binary classification task, and the solution relies on supervised learning algorithms.

Our methodology was applied to 20K ICU stays from the Medical Information Mart for Intensive Care (MIMIC-III) [Johnson et al., 2016] database. We explored the diversity of patient data in MIMIC-III, including medical notes and vitals, to generate accurate predictions. We grouped the patient's features into structured data, structured time series data, and textual time series data. Our analysis considered each data both

separately and combined using ensemble learning. Our goal was to assess the contribution of different types of data combined in an ensemble and whether such a combination can improve in-hospital mortality prediction.

The results obtained by the ensemble were 0.853 (95% CI [0.846, 0.861]) in terms of the area under the receiver operating curve and 80% of true positive rate. This represents an increase of three percentage points compared to the best experimental run using a single data type.

2. Background

In this section, we introduce the underlying concepts and techniques related to our methodology for predicting in-hospital mortality.

2.1. Text Representations.

Because free text does not have a structure, machine learning methods cannot directly extract information from its raw format. As a result, we need to find a way to represent this type of data so it can be fed into a machine learning algorithm. Representation Learning is the field of Machine Learning that allows systems to learn and transform data into a representation that differs from its original format. The performance of Machine Learning algorithms is directly affected by the data representation, and the process of hand engineering a representation is laborious and requires domain knowledge. A popular method for generating a continuous representation of a sentence is Doc2Vec [Le and Mikolov, 2014]. Doc2Vec can be used to generate a real-valued vector representation of the medical notes associated with the patients. The new representation generated for an input text is able to represent the semantic information from this text and can be used as input to a neural network directly.

2.2. Deep Learning Algorithms.

Deep learning algorithms have shown impressive results in many classification tasks and quickly became state-of-the-art. A key aspect is that these algorithms do not require feature engineering, which is a time-consuming step in the classical machine learning background. Recurrent Neural Networks (RNN) are Neural Networks that have the ability to process sequential data, learning a probabilistic distribution over the sequence.

The Long Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] is an RNN architecture that uses a sophisticated, non-linear network activation function, based on specialized units called *gates* to capture long-term dependencies in the data. An LSTM model is composed of multiple gates that act on the gradient, controlling the gradient value inside each memory unit that builds the network. LSTMs can deal with time-series data and are widely used in a diversity of classification problems for this capability. Here we use it as one of our base-models. Convolutional Neural Networks (CNN) [Lecun et al., 1998] are capable of extracting spatial or temporal features, using two different operations, *convolution* and *pooling*. With these operations, CNNs output a smaller, more meaningful representation that is fed to a highly connected network. Temporal Convolutional Neural Network (TCN) is a modification to CNN, creating constraints to pooling and convolution to transform sequential data without future data leaking to the past.

2.3. Ensemble Learning

An ensemble learning algorithm is a method of combining multiple models or generalizers into a single model with better predictive performance. Usually, the models that compose an ensemble reflect subspaces of the entire hypothesis space reflected by the data used. One method to create and combine multiple models is Stacked Generalization, or *Stacking* [Wolpert, 1992].

Given a set $D = \{(x_n, y_n), n = 1, ..., N\}$, with x_n being the values of the features for the data instance n and y_n being the class of this instance, let us denote D_j and $D^{-j} = D - D_j$ as the test and training set, respectively, of a j-fold cross-validation over the dataset D. Now let us define a set of k learning algorithms, which we call *base-models*, chosen to train over this data to generate a set of base-models M_k^{-j}. For each instance x_n in D_j, we use the predictions z_{kn} generated by each model M_k^{-j} as the new representation for x_n. At the end of the cross-validation process, data is represented by $D_{CV} = \{(y_n, z_{1n}, ..., z_{kn}), n = 1, ..., N\}$ and is used to train a new model, which is called *meta-model* [Ting and Witten, 1999].

3. Related Work

In medical sciences, an early prediction of the patient's outcome is beneficial to improve the quality of treatment and reduce costs. Thus, there is a significant body of research aimed at early detection which relies on medical scores and/or machine learning.

3.1. Mortality Prediction using SOFA

The Sequential Organ Failure Assessment (SOFA) score [Vincent et al., 1996] provides means for the identification of multiple organ failures. SOFA uses a numeric scale from 0 to 4, in which higher values mean a more significant organ dysfunction. Although it was not designed for mortality prediction, some studies established a correlation between the SOFA score (measured at the patient's admission) and mortality [Jentzer et al., 2018, Mbongo et al., 2009, Ho et al., 2007].

A systematic review [Minne et al., 2008] surveyed 18 studies that evaluated the performance of SOFA for mortality prediction of ICU patients. The AUROC scores reported in these studies range from 0.61 to 0.87. Subsequently, Mbongo et al. [2009] analyzed 864 patients, with a mortality rate of 8.2%, and found that SOFA achieved an AUROC of 0.846 when discriminating survivors vs. non-survivors. More recently, Jentzer et al. [2018] analyzed 9,961 ICU patients with 893 (9%) of those having died in-hospital. The SOFA score calculated on the first day predicted mortality with an AUROC of 0.828.

3.2. Mortality prediction using machine learning

Harutyunyan et al. [2019] presented a benchmark of several tasks in the medical field using the MIMIC-III database [Johnson et al., 2016]. The authors trained several LSTM architectures in different tasks in the medical domain. The goal was to use multitask learning to extract certain useful information from the input sequence that single-task models could not leverage. Instead, the authors created a single model that was trained using all of the evaluated tasks. The results achieved a higher AUROC in all the tasks in comparison with the metrics and the individual task models. For in-hospital mortality prediction, the AUROC was 0.87 (95% CI [0.852,0.887]).

Pirracchio et al. [2015] compared medical scores for severity assessment with machine learning algorithms. A *Super Learner* algorithm was trained on data from 24,508 patients from the MIMIC-II database Saeed et al. [2011] for in-hospital mortality using the first 24h of data following the ICU admission. The AUROC results were 0.88 (95% CI [0.87, 0.89]) outperforming APACHE-II, SAPS-II, and SOFA scores.

Sushil et al. [2018] used clinical notes from the MIMIC-III dataset to train classification algorithms for mortality prediction (in-hospital, post-30 days discharge mortality, post-one-year discharge mortality). Using *Stacked denoising autoencoder* (SDAE) and Doc2Vec to generate new representations for the concatenation of all clinical notes associated with the patient. The work achieved the highest AUROC of 0.9457 by feeding only a bag-of-words representation into a feedfoward neural network for the in-hospital mortality task.

Specifically, on the topic of ICU mortality prediction using different types of data, there are two closely related works, which also used MIMIC-III [Weissman et al., 2018, Hashir and Sawhney, 2020]. Weissman et al. [2018] used structured data from lab results and bedside measurements and unstructured data from the medical notes. By employing two approaches to add unstructured data to the machine learning model, the authors found that adding unstructured data yields an improvement of 0.06 to 0.09 points in AUROC, reaching 0.89 (95% CI [0.88, 0.90]).

Hashir and Sawhney [2020] uses hierarchical CNN-RNN to model multiple notes. The experiments used around 35K ICU stays, and achieved the best results when a joint model of notes and structured clinical time series was applied. This configuration scored 0.902 (95%CI [0.898, 0.906]) in terms of AUROC using data from the first 48h of the patient's stay.

The main aspects that differentiate this work from the works listed in this section are as follows. The work by [Weissman et al., 2018] uses medical notes but it ignores the time-changing information aspect of the patient state when there is a decision to concatenate all medical notes associated with each patient. In our work, we maintain the time-varying aspect of data and make use of Deep Learning and representation learning techniques that can handle time series as input. By maintaining the time aspect of the data, we make use of the information present in the patient's evolution during the ICU stay, which could be favorable to our final predictor. In relation to the work by [Hashir and Sawhney, 2020], we point to two main differences: (i) besides considering time-series data, we also consider static structured data, and (ii) while their multimodal approach uses joint training, ours relies on an ensemble of independent base-models.

4. Data preparation

Our data comes from the MIMIC-III database [Johnson et al., 2016], which has data for patients admitted at the Israel Deaconess Medical Center ICU, in Boston. Each patient has one or more hospital admissions in the database, and each hospital admission could have one or more ICU stays.

During a stay, medical staff visits the patient several times to perform observations, take measurements, or administer treatments. Each of these is called an *event*, *i.e.,* a categorical or numerical value measured at a specific point in time. Several events

could be created for a patient in a single visit. MIMIC-III has a total of 61,532 ICU admissions. Our data selection for in-hospital mortality employed the same procedure adopted by Harutyunyan et al. [2019]. The following stays were discarded:

- *Multiple stays*. Hospital admissions that had multiple ICU stays or had any transfers between ICU units or wards during the period of hospitalization. The reason is that the multiple ICU stays would be correlated as they belong to the same individual and a survival outcome may be followed by a death adding noise to the model. These admissions correspond to 11,346 records.
- *Patients under 18 years old* were also removed due to differences between adult and pediatric physiology. This step removed 7,910 ICU stays.
- *Short stays*. ICU stays of less than 48 hours are not relevant to this study since our goal is to use the first 48 hours to predict the outcome. This filter removed 20,974 ICU stays.
- *Stays without medical notes*. All ICU stays that had no recorded medical notes in the first 48h of stay were also removed. The reason for this is that it would create an inconsistency in the comparison between the models generated with structured temporal data and textual temporal data since missing values would affect the final ensemble model.

Our final dataset has a total of 20,083 stays. Of those, 17,359 patients were discharged alive from the ICU (negative instances) and 2,724 patients died (positive instances), making that an unbalanced dataset. From the selected ICU stays, we extracted the events that are relevant to our classification problem. The values of each feature are aggregated by hour, starting at the time of admission in the ICU, until the 48^{th} hour. If more than one measurement was made in this one-hour bucket, we calculate the mean. To handle missing data, we filled each value with the last measurement, and in case no measurement was taken, we use the mean value extracted from the entire dataset. For clinical notes, we aggregated them by the concatenation of the notes.

5. Predicting In-hospital Mortality

We model the task as a binary classification problem in which the positive class is the death outcome. We rely solely on data generated during the *first 48 hours* of the ICU stay. Data about a patient can be divided into three types: (i) structured data (*e.g.,* weight, height, and sex); (ii) structured time-series data (*e.g.,* results of exams and vital signs); and (iii) textual time series (medical notes taken through the ICU stay).

We approached model creation using a stacking algorithm to create an ensemble using multiple models trained on the different types of data. In addition, solutions for tackling the class imbalance problem were employed both in the base- and meta-models.

5.1. Data Types

Each data type requires specific methods to allow its use in training a machine learning algorithm.

Structured Data (SD). Our structured data consists of information that does not change over time or data for which changes are not substantial. The purpose of the structured data is to give contextualized information about the patient. The features in this category

are weight and height at admission, age, and sex. The weight and height variables were extracted from the admission notes for each ICU stay, and the other values were extracted from the admission data recorded for each stay.

Structured Time-Series (STS). data are generated by measuring patients' vital signs and recording the results of their laboratory exams performed during the patient's ICU stay. It consists of numerical and categorical data measured at a specific point in time by a healthcare provider. Our STS features are Glasgow Comma Score, Systolic Blood Pressure, Diastolic Blood Pressure, Mean Blood Pressure, Central Venous Pressure, Heart Rate, Respiratory Rate, Blood Oxygen Saturation, Body Temperature, Hemoglobin, Hematocrit, WBC, Platelets, Arterial pH, PaO2, FiO2, PaCO2, Lactate, Creatinine, BUN, Bilirubin, Potassium, and Glucose. These features were selected by a physician and reflect the health state in which the patient's physiological systems are at the moment. Data from an ICU poses a diversity of challenges since their generation is a product of measurements made by a team composed of different professionals and the nature of the patient's disease and condition. As a consequence, measurements are spread through time at a variable rate, sometimes having hours between measurements and sometimes minutes or days. One of the problems created by the irregular measurement of features is missing values. For each feature, we have a total of 960,219 events; from those, 126,987 events come from patients who died in the ICU, and 833,232 from patients who survived. During our experiments, the missing values were replaced by the previous measurement or the mean value obtained by adding all events for each feature. To deal with the STS data, the method needs to be able to process and extract meaningful patterns through time and to deal with large volumes of high-dimensional data. For the purpose of classifying STS data, we train LSTM and TCN models as these architectures have good performances dealing with temporal data, without the need for dimensionality reduction or feature extraction. The training process for the model follows a process of hyperparameter tuning on a subset separated from the training data, and the use of k-fold cross-validation as training process.

Textual Time-Series (TTS). As measuring vitals and performing exams on patients generates structured temporal data, a visit of a healthcare provider or a health exam can also produce textual data describing the information that is important to understanding the patient's history and condition. Along with the challenges of dealing with textual data, these texts are distributed through time, which adds another layer of complexity. Here, the problem of missing data is yet more prevalent than on STS data. To be precise, the total number of events in the extracted dataset is 143,970, with positive instances having 21,552 events and negative instances having 122,418. To be able to use textual data as input to a machine learning model, we adopted the sentence embedding method using Doc2Vec (discussed in Section 2.1) to transform the raw text into a real-valued vector. This is done for each text associated with each patient, and in that manner, we preserve the temporal aspect of the data, differently from the approach adopted by Weissman et al. [2018] discussed in Section 3. To train the Doc2Vec model, we used all text available in the MIMIC-III database, and not just the texts in our sample. Having large volumes of text is important to yield good-quality representations. Since Doc2Vec is an unsupervised method, it does not take the class we wish to predict into consideration. Thus, it does not introduce bias over our classification model. After generating the representation model and transforming the texts of our dataset to the new representation, we followed the same

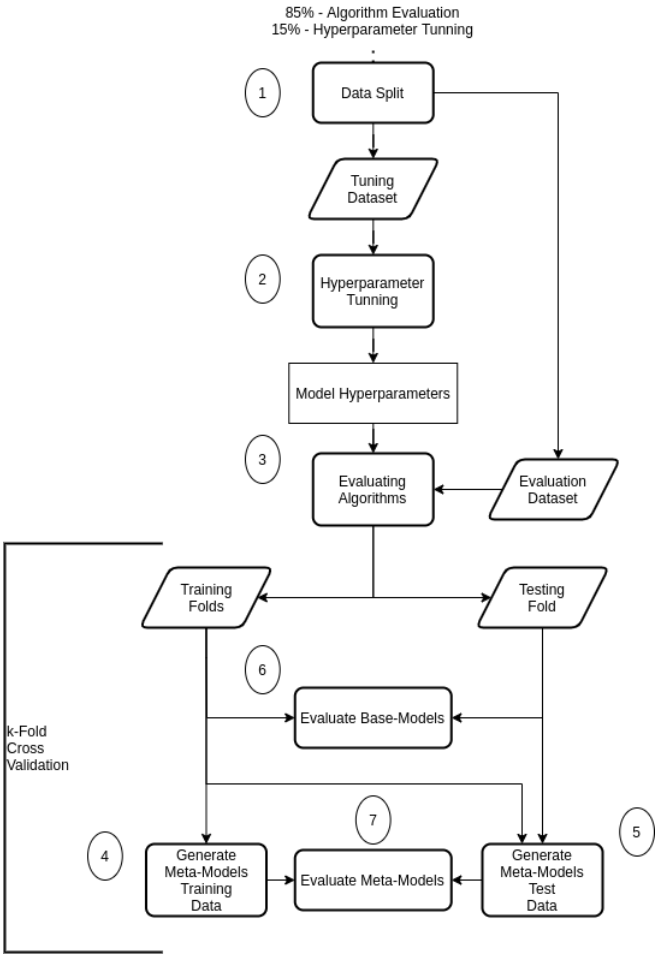

Figura 1. The model training and evaluation pipeline.

steps taken for the STS data. It is important to notice that both models trained on STS and TTS data used the same samples of patients and folds.

5.2. Balancing the Training Data

As discussed in Section 4, our dataset is very unbalanced, with the positive instances corresponding to 13.5% of the total. This distribution of data between classes can be harmful to the final performance of the classification model since it could be biased towards the negative class, which is more prevalent. To deal with this issue, we applied a random undersample of the training data, and consequently the optimization data, by a rate of 1:1, while maintaining the same distribution between classes in the test set

5.3. Ensemble Model Creation

As discussed in Section 2.3, ensemble classifiers usually have better predictive performance since they combine more sources of evidence from the data. Our meta-models combine the predictions of the base-models that were trained using data from a single data type. There are four base-models (i) LSTM trained on the STS data, (ii) TCN trained on the STS data, (iii) LSTM trained on TTS, and (iv) TCN trained on TTS. The classification results yielded by the models trained solely using SD were unsatisfactory

(*i.e.,* around 0.6 in terms of AUROC). Thus, the class predictions made using SD were not directly fed into the ensemble model. Instead, we concatenated the raw SD into the predictions generated with STS and TTS data to compose the meta-model training and test data. The rationale was to test whether, despite not yielding good predictions on its own, SD could still be useful when combined with other patient data that could potentially provide more context.

Figure 1 illustrates the pipeline used to train our classification models. In (1), the dataset is split into hyperparameter tuning and model evaluation with each subset accounting for 85/15% of the instances, respectively. The split was stratified to maintain the same class distribution in each partition. Hyperparameter tuning (2) takes the tuning partition and runs the hyperparameter search algorithm for each of the base-models. Then, with these selected hyperparameters, in (3) we execute the leraning process for each classification algorithms, using k-fold cross-validation.

In step (4), the predictions (*i.e.,* class probabilities) from the trained base-models are used as training data for the meta-model. To avoid data leakage, the training folds is split into p-1 *training folds* to generate the probabilities for the remaining fold. This process is repeated p times, and at the end of this process, we have the predictions for all instances for the first training folds. In (5) we generate the test data for the meta-models by training the base-algorithms on the training folds and generating the predictions on the test fold. In (6) and (7) the base- and meta-models evaluation metrics are computed on the test-folds.

6. Experiments

In this section, we describe the experimental evaluation that we performed aiming at answering the following questions: (i) What type of data provides the best classification performance? and (ii) Does combining different types of data in a heterogeneous ensemble model improve the results of in-hospital mortality prediction?

6.1. Materials and Methods

Our experiments were done in Python. Keras was used for training the models, and Pandas was used to manipulate and process the dataset. Scikit Learn was used to compute the evaluation metrics and to generate the meta-models. Hyperparameter tuning was done using the Hyperband method [Li et al., 2017] in the Keras Tuner library. For each hyperparameter optmized, we define a min and max value to use as range for the Hyperband to search on. The hyperparameters and their values are: number of hidden layers between one and four; the units for each layer between eight and 256. For TCN specifically, we varied the kernel size between three and five, the number of dilations between one and four. The use of a dropout was conditioned randomly for each layer and varied between 0.2 and 0.5. We set the loss function to Binary Crossentropy, the hidden layer activation as LeakyRelu, the training optimizer as Adam, and the activation as the Sigmoid function.

The training process follows the Stacking algorithm. The base-models were trained using 5-fold cross-validation and, for each loop, the network output for each instance in each testing fold was concatenated with its respective structured features and to the meta-model. After generating the training data, we create the data that will be used to evaluate the meta-models by training each classification algorithm used in the base-models in the entire training dataset, generating their predictions and concatenating them

with the structured features for the ensemble evaluation dataset. For the meta-models, we employed Support Vector Machine with linear kernel and Logistic Regression.

Our results were scored according to six widely used classification metrics, namely Precision, Recall, TP Rate, AUROC, and F-Score. Since our data is unbalanced, we looked at F-Score in two ways – assigning equal weights to the instances (W F-Score) and assigning equal weights to the classes (U F-Score). Using different metrics is important to allow interpreting the results from different perspectives.

6.2. Results

Base-models. Table 1a shows the results of the evaluation metrics calculated using 5-fold cross-validation for all base-models. Lines 1 and 2 show the scores for models trained only on SD data. We observe a high *TP Rate* and a low *Recall*, which shows that these models sacrificed the ability to classify negative instances to be able to classify positive instances. Between the algorithms, the highest *TP Rate* was obtained by SVM (line 1). SVM is the best-performing model for classifying positive instances, however, its low *Recall* is a result of a high number of false positives. STS data type had a better performance with TCN (line 4), achieving the best score for *Precision*, *W F-Score*, *U F-Score*, and *AUROC*. This algorithm had a better *TP Rate* than the LSTM (line 3). The TTS data type results have a particular case. Both TCN and LSTM (lines 5 and 6) have similar scores in *AUROC*. LSTM had a better performance classifying negative instances and TCN had a better performance classifying positive instances. This can be seen by LSTM's lower *TP Rate* and higher *Recall*.

By analyzing all results obtained using each data type, STS clearly had a better classification performance compared to the other data types. Thus, we conclude that the answer to our first question is that *STS is the data type with the best predictive power*.

Finally, our results show that no single algorithm or data type was able to achieve good results in all evaluation metrics. In this sense, the choice of the best model depends on the goals of the task at hand.

Meta-Models. The results of the experimental runs using a combination of data types are in Table 1b. The best experimental run is the one in which all types of data were combined, *i.e.,* SD+STS+TTS using LR (line 8). It had the best results in terms of *Recall*, *W F-Score*, *U F-Score*, and *AUROC*, while maintaining a high performance in *TP Rate* and *Precision*.

Comparing the results of the different classification algorithms on the ensembles, we see that SVM is still associated with the highest *TP Rate* (line 13) and the lowest *Recall* (line 11), *U F-Score*, and *W F-Score*. LR had more stable results. By analyzing the performances, we conclude that if the goal is to prioritize the identification of the positive class, the SVM algorithm is the best, but at cost of the performance on the negative class, which produces a high number of false positives. However, if the goal is to maintain a good balance between the performance in positive and negative classes, the LR algorithm is the best option.

Figure 2 shows the results for each data type used on their own and in heterogeneous combinations (the best run for each data type/combination was selected for comparison). We use primary/secondary colors to denote the data types and their combinations,

Tabela 1. Results for the experimental runs. Best scores in bold

(a) Results for the base-models using 5-fold cross-validation.

	Data Type	Algorithm	Precision	Recall	W F-Score	U F-Score	TP Rate	AUROC
1	SD	SVM	0.807	0.314	0.296	0.276	**0.830**	0.582
2	SD	LR	0.801	0.551	0.620	0.473	0.616	0.612
3	STS	LSTM	0.854	**0.730**	**0.769**	0.621	0.710	0.790
4	STS	TCN	**0.869**	0.728	**0.769**	**0.631**	0.794	**0.824**
5	TTS	LSTM	0.831	0.639	0.693	0.542	0.670	0.712
6	TTS	TCN	0.840	0.578	0.639	0.508	0.757	0.714

(b) Results for the heterogeneous ensemble models.

	Data Types	Algorithm	Precision	Recall	W F-Score	U F-Score	TP Rate	AUROC
7	STS+TTS	SVM	0.874	0.739	0.778	0.643	0.813	0.844
8	STS+TTS	LR	0.873	0.755	0.790	0.654	0.788	0.844
9	SD+STS	SVM	0.875	0.666	0.718	0.591	0.869	0.841
10	SD+STS	LR	0.872	**0.761**	**0.795**	0.657	0.775	0.841
11	SD+TTS	SVM	0.864	0.466	0.522	0.436	0.911	0.759
12	SD+TTS	LR	0.850	0.650	0.704	0.564	0.759	0.761
13	SD+STS+TTS	SVM	**0.881**	0.573	0.615	0.518	**0.927**	0.852
14	SD+STS+TTS	LR	0.876	**0.761**	**0.795**	**0.661**	0.800	**0.853**

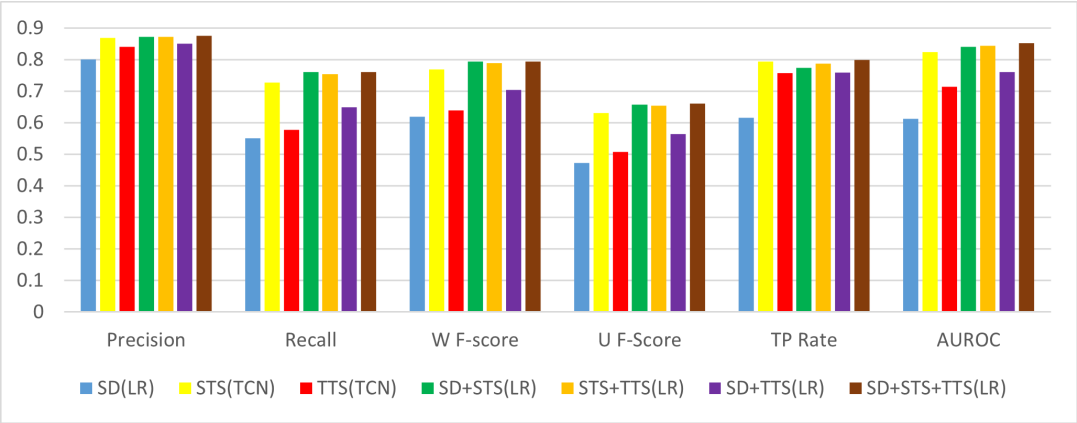

Figura 2. Results for different data types and heterogeneous combinations

i.e., the individual data types are represented by blue (SD), yellow (STS), and red (TTS); the combination of SD and STS is in green, and so on. We found that despite SD having the lowest results, if used in combination with other data types, the results improve. This can be seen comparing, for example, lines 3 and 10 in which SD+STS is better than any of these data types used in isolation. In Figure 2, the green bar is consistently higher than the blue and yellow bars. We can conclude that the answer to our second question – *Does combining different types of data in a heterogeneous ensemble model improve the results of in-hospital mortality prediction?* is yes. We see a clear improvement when all data types are combined using LR algorithm.

Comparison with a Baseline. The intuition for using SOFA as a baseline is to obtain a classification model based on a popular score that is widely applied in predicting the clinical outcomes of critically ill patients. To be comparable with our experimental runs, in which only the first 48 hours of data are considered, our baseline used the hourly SOFA

scores for this time window. With the hourly SOFA scores for each patient, we extracted time-series features for SOFA values for each ICU stay using the Python package *tsfresh* [Christ et al., 2018]. Then, the same package was used to select the relevant features. Next, we ran a Logistic Regression classifier on these features.

The heterogeneous ensemble used in the comparisons in this section combines all three data types using a LR classifier. Its details are in line 8 in Table 1b. We also calculated *AUROC* for the SOFA baseline. The result was 0.755 (95% CI [0.744 0.767]), being outperformed by our ensemble model which had an *AUROC* of 0.853 (95% CI [0.846, 0.861]). As reported in Section 3.1, previous works using SOFA for mortality prediction have identified AUROC results ranging from 0.61 to 0.87. Our scores are within that range. In relation to the work by [Jentzer et al., 2018], the lower scores found here can be explained by our larger number of instances (we use twice as many patients) and the fact that the patients in the aforementioned work were from a single cardiac ICU and here the patients come from five different ICUs and from a wider spectrum.

Finally, we inspected the confusion matrices generated for the SOFA model and for the same heterogeneous ensemble (LR-SD+STS+TTS). The heterogeneous ensemble was able to correctly classify an additional 368 patients into the positive class (an increase of 25%) and 169 into the negative class.

7. Conclusion

We investigated the use of ensemble models built using heterogeneous types of data (structured, text, and time series) for in-hospital mortality prediction. We carried out a series of experiments using data for 20K hospital stays from MIMIC-III. We designed a methodology to process each type of data. First, base-models were created for each type on its own and then the different data types were combined in meta-models using ensemble learning with the stacking strategy. Our results were evaluated using six classification quality metrics.

Looking at the results of the base-models, we concluded that, among the individual data types, structured time series provided the best classification models. When the heterogeneous meta-models were considered, we verified that the use of different types of data brings increases the classification performance.

The results obtained in this work can be used as a base for future studies using heterogeneous data types. Here we applied an ensemble methodology, in which models could be trained with specific types of data without influencing each other, and achieved an increase in performance with the meta-model. This leaves an open question as to whether it is possible to obtain performance improvements by combining the heterogeneous data types in some other manner, *e.g.,* in a joint training method [Hashir and Sawhney, 2020].

Referências

Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018. ISSN 0925-2312.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1), Jun 2019. ISSN 2052-4463.

Mohammad Hashir and Rapinder Sawhney. Towards unstructured mortality prediction with free-text clinical notes. *Journal of Biomedical Informatics*, 108:103489, 2020.

Kwok Ho, KY Lee, Teresa Williams, Judith Finn, M Knuiman, and S Webb. Comparison of acute physiology and chronic health evaluation (APACHE) II score with organ failure scores to predict hospital mortality. *Anaesthesia*, 62:466–73, 06 2007.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667.

Jacob Jentzer, Courtney Bennett, Brandon Wiley, Dennis Murphree, Mark Keegan, Ognjen Gajic, R. Wright, and Gregory Barsness. Predictive value of the sequential organ failure assessment score for mortality in a contemporary cardiac intensive care unit population. *Journal of the American Heart Association*, 7:e008169, 03 2018.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Cissé-Luc Mbongo, Pablo Monedero, Francisco Guillen-Grima, Maria Yepes, Marc Vives, and Gemma Echarri. Performance of saps3, compared with APACHE II and SOFA, to predict hospital mortality in a general ICU in southern europe. *European journal of anaesthesiology*, 26:940–5, 08 2009.

Lilian Minne, Ameen Abu-Hanna, and Evert de Jonge. Evaluation of SOFA-based models for predicting mortality in the ICU: A systematic review. *Critical care*, 12(6):1–13, 2008.

Romain Pirracchio, Maya L Petersen, Marco Carone, Matthieu Resche Rigon, Sylvie Chevret, Prof Mark, and J Van Der Laan. Mortality prediction in the ICU: can we do better? Results from the Super ICU Learner Algorithm (SICULA) project, a population-based study. *The Lancet. Respiratory medicine*, 3 (1):42–52, 2015.

Mohammed Saeed, Mauricio Villarroel, Andrew Reisner, Gari Clifford, Li-wei Lehman, George Moody, Thomas Heldt, Tin Kyaw, Benjamin Moody, and Roger Mark. Multiparameter intelligent monitoring in intensive care II MIMIC-II: A public-access intensive care unit database. *Critical care medicine*, 39: 952–60, 05 2011.

Madhumita Sushil, Simon Šuster, Kim Luyckx, and Walter Daelemans. Patient representation learning and interpretable evaluation using clinical notes. *Journal of Biomedical Informatics*, 84:103–113, 2018. ISSN 1532-0464.

K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, jan 1999. ISSN 10769757.

Jean-Louis Vincent, Rui Moreno, Jukka Takala, S Willatts, A Mendonça, H Bruining, C Reinhart, Peter Suter, and L Thijs. The SOFA (sepsis-related organ failure assessment) score to describe organ dysfunction/failure. *Intensive care medicine*, 22:707–10, 08 1996.

Gary E. Weissman, Rebecca A. Hubbard, Lyle H. Ungar, Michael O. Harhay, Casey S. Greene, Blanca E. Himes, and Scott D. Halpern. Inclusion of unstructured clinical text improves early prediction of death or prolonged ICU stay. *Critical Care Medicine*, 46(7):1125–1132, 2018. ISSN 15300293.

David Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 12 1992.