Machine Learning Project 2022/23

Braulio Villalobos-Quiros[a]
[a]villalobosquiros.1999250@studenti.uniroma1.it

July 14, 2023

Abstract

This project uses Electronic Health Records of patients and aims to predict the presence of cardiovascular events within a six months horizon. The variable to predict is dichotomous as the patient can or can't experience a cardiovascular event. The data presented some challenges such as big dimensionality, sparsity, poor quality of data in some cases, among others. Understanding how to join data from different sources was also an issue to overcome. After an intensive pre-processing, carried out in Task 1, several Neural Networks architectures are implemented, trained and tested in Task 2, such as LSTM, T-LSTM and PubMedBERT. Three different level of complexity LSTM were implemented a simple, a medium and a complex version. On the other hand, a logistic regression and a simple neural network were trained and tested based on the embedddings created with PubMedBERT. An strategy to ignore the temporality of the microevents (i.e, non cardiovascular events) coupled with a Bayesian Optimization carried out for hyperparameter tuning over the T-LSTM model is done in Task 3.

1 Introduction, context, and motivations

Data has revolutionized the world in which we live in, by impacting almost every scientific field in recent years. The medical field has been no exception to this Nambiar, Bhardwaj, Sethi, and Vargheese (2013). Electronic Health Records (EHC) are an important source of information than can help to bring a better quality of life to patients, by predicting health risks or suggesting treatments for conditions that might not yet be present Miriovsky, Shulman, and Abernethy (2012). This is a huge change in the medicine paradigm, as the practice of medicine passes from being reactive to being preventive. In this way, EHC might be the founding pilar of what is known as Precision Medicine.

However, the analysis of this data is not straightforward. EHC data is usually characterized by: first, a significant heterogeneity, as patients not necessarily suffer from the same symtoms or events and they also hardly follow a pattern. Second, poor data quality is frequently a challenge to overcome when working with EHC. This is a consequence of slowly changing to digitalized processes, which is generally accompanied by missing, wrongly taken data, etc. Third, unbalanced data usually posses a problem to EHC analysis. Intuitively, we would expect that the majority of people don't suffer a hearth attack, even though this might be the most usually cause of death. For this reason, highly unbalanced data is frequently present in EHC. Fourth, anonymity and confidentiality issues are frequently associated with EHC Keshta and Odeh (2021). In recent years, federated learning or networks, have emerged as a solution for this. Several strategies were used to overcome the descripted challanges associated with EHC data analysis.

As previously mentioned, given a set of features, we want to predict the presence of a cardiovascular event within 6 months time horizon. To accomplish this, we combined data about the patients' diets, prescripted drugs, medical tests, laboratory tests, diagnosis tests, among others. The data is described in the following section.

As requested in the project guidelines, an intensive data pre processing was carried out in the first task. In the second task, after balancing the dataset, some Deep Learning architectures were implemented, trained and used to predict the presence of a cardiovascular event. These models were: Long Short Term Memory (LSTM), Time-Aware Long Short Term Memory (T-LSTM) and PubMedBERT. In the third task, we developed a strategy to consider only the temporal sequence of the macro events and drop the temporality of the microevents, while preserving some information about the frequency of microevents. Lastly, a Bayesian Optimization is carried out with the purpose of finding the optimal hyperparameters

for the T-LSTM model. We make a comparison between the optimal bayesian T-LSTM and the default parameters of LSTM and T-LSTM.

This project aims to expose and solve challenges that are usually encountered when working with EHC data. The author of this project strongly believes that benefits such as a better quality of life alongside with a bigger efficiency of the health systems, are hidden in data like the one analyzed in this project. Therefore, solving these challenges and coming up with new methodologies is of great importance to achieve these benefits.

2 Dataset description

The dataset is composed by 8 different tables. All of them have at least 2 common columns called "id center" and "id ana" which together serve as the unique identifier of a patient. The dataset contains data of quite diverse nature as it includes demographic information of the patients along with their medical history, given by diagnosis, prescripted drugs, medical and laboratory tests, among others.

Another important aspect to clarify before describing the data is what we understand by AMD codes, ATC codes and STITCH codes:

- AMD codes: codes that identity the different type of patologies related with diabetes and other diseases or conditions, created by an association of diabetical medicine in Italy.
- ATC codes: Anatomical Therapeutic Chemical Classification System which serves to codify prescripted drugs.
- STICH codes: these are created by the Sapienza Information-Based Technology Innovation Center for Health.

We have demographic information for 250.000 patients. Following, we proceed to concisely describe each of the tables that are used in this project:

1. **anagraficapazientiattivi** (active_patients_info): contains demographic data of patients such as sex, year of diagnosis of diabetes, type of diabetes, education, marital status, profession, origin, birth year, year of first access to the system and death year. Particularly the type of diabetes, education, marital status, profession and origin aren't used as they contain a huge number of missing values or they don't add any value to the prediction power of the models, given their low variablity or the missing values.
2. **diagnosi** (diagnosis_tests): contains the diagnosis for each patient along with the date of the diagnosis, its code and its value.
3. **esamilaboratorioparametricalcolati** (laboratory_tests_calculated_parameters): contains the laboratory tests carried out to patients along with its amd code, stich code and the date of the test.
4. **esamilaboratorioparametri** (laboratory_tests): contains the laboratory tests carried on on each patient along with the amd code, the value and the stich code.
5. **esamistrumentali** (medical_tests): contains the medical tests done to the patient along with its corresponding amd code, date of the test and the value.
6. **prescrizionidiabetefarmaci** (prescriptions_of_diabetes_drugs): contains information on the prescriptied drugs to each patient, related to diabetes. It includes the atc code, the drug prescription, among others.
7. **prescrizionidiabetenonfarmaci** (prescriptions_of_non_diabetes_drugs): contains information about the non diabetes drugs that were prescribed to the patient along with the date and the code of the prescription according to the AMD.
8. **prescrizioninondiabete** (patients_diets_and_blood_glucose_controls): contains the code, date and value of the controls of glucose of the patients.

   It is important to mention that as a previous pre processing step, we dropped the duplicates. This doesn't change the number of observations much, however the following table allows us to appreciate the dimensionality of each table and the slight reduction as we dropped duplicated instances.

Table 1: Dimension of Tables before and after dropping duplicates

Table Name	Before	After
active_patients_info	(250000, 13)	(250000, 12)
diagnosis_tests	(4427337, 6)	(4427337, 5)
laboratory_tests	(28628530, 6)	(28628502, 5)
laboratory_tests_calculated_parameters	(10621827, 7)	(8698045, 6)
medical_tests	(1015740, 6)	(1015740, 5)
prescriptions_of_diabetes_drugs	(7012648, 8)	(7012648, 7)
prescriptions_of_non_diabetes_drugs	(548467, 6)	(548467, 5)
patients_diets_and_blood_glucose_controls	(5083861, 6)	(5083861, 5)

3 Task 1

We define the following as the macro events and also the events of interest:

3.1 Select events of interest

1. AMD047: Myocardial infarction
2. AMD048: Coronary angioplasty
3. AMD049: Coronary bypass
4. AMD071: Ictus
5. AMD081: Lower limb angioplasty
6. AMD082: Peripheral By-pass Lower Limbs
7. AMD208: Revascularization of intracranial and neck vessels
8. AMD303: Ischemic stroke

We selected those patients that have a cardiovascular event and filtered all the tables by them. The previously mentioned cardiovascular events are only present in the Diagnosis table. So we identified all the unique combinations of id center and id ana, whose amd code was within the amd codes of interested. This gave us 50.000 patients. We then filtered all the remaining tables to only contain these patients of interest. The change in dimensionality after this filtering is shown in table 2.

Table 2: Dimension of Tables before and after selecting events of interest

Table Name	Before	After
active_patients_info	(250000, 12)	(50000, 12)
diagnosis_tests	(4427337, 5)	(1938342, 5)
laboratory_tests	(28628502, 5)	(7371151, 5)
laboratory_tests_calculated_parameters	(8698045, 6)	(2279156, 6)
medical_tests	(1015740, 5)	(290793, 5)
prescriptions_of_diabetes_drugs	(7012648, 7)	(1989613, 7)
prescriptions_of_non_diabetes_drugs	(548467, 5)	(150340, 5)
patients_diets_and_blood_glucose_controls	(5083861, 5)	(1995073, 5)

3.2 Invalid feature cleaning

To remove the invalid features (i.e observations that don't make sense because there's a inconsistency on the data) several tests were carried out.

We removed those patients who:

1. Patients with a birth year that happened **after** the death year of the patient.

2. Patients whose first year of access to the system was before their birth date or after their death year.

3. Patients whose year diabetes diagnosis was before their birth date or after their death year.

By doing this, we removed 10 patients. The change in dimensionality by removing these patients and the consequent filtering of the remaining tables, is shown in table 3.

Table 3: Dimension of Tables before and after invalid feature cleaning

Table Name	Before	After
active_patients_info	(49990, 12)	(49990, 12)
diagnosis_tests	(1938342, 5)	(1938123, 5)
laboratory_tests	(7371151, 5)	(7370649, 5)
laboratory_tests_calculated_parameters	(2279156, 6)	(2278985, 6)
medical_tests	(290793, 5)	(290769, 5)
prescriptions_of_diabetes_drugs	(1989613, 7)	(1989453, 7)
prescriptions_of_non_diabetes_drugs	(150340, 5)	(150323, 5)
patients_diets_and_blood_glucose_controls	(1995073, 5)	(1994962, 5)

3.3 Remove patients with short trajectory

It was requested to remove those patients who have a trajectory of 1 month. By trajectory, we understand the collection of all the events, in any of the tables, that are related to one patient. In other words, all the registers of a patient through all the tables. Then, if all the events of a patient took place in the **same** month, we remove those patients. For the sake of clarity, this means that if all the events of a patient took place within 1 month, but there exists one single event that took place outside that month, that patient is preserved. This month is doesn't refer to 30 days but to calendar month. This means that if a patient has an event on July 31th and another one on August 1st, this patient **isn't** removed.

This was done by grouping by id ana and id center and counting the number of unique months and years. Then, we removed those patients whose events are only concentrated in 1 month and 1 year. If one of those unique values is different than 1, this would mean that the event took place out of the 1 month interval.

By doing this, we removed 897 patients, who had all of their events concentrated in the same month. After this, we proceed to filter the rest of the tables by removing these patients. Changes in dimensionality are shown in table 4.

Table 4: Dimension of Tables before and after removing patients with short trajectory

Table Name	Before	After
active_patients_info	(49990, 12)	(49093, 12)
diagnosis_tests	(1924665, 10)	(1919045, 10)
laboratory_tests	(7369604, 10)	(7361304, 10)
laboratory_tests_calculated_parameters	(2278643, 11)	(2276064, 11)
medical_tests	(290732, 10)	(290314, 10)
prescriptions_of_diabetes_drugs	(1989158, 9)	(1986333, 9)
prescriptions_of_non_diabetes_drugs	(150287, 7)	(150031, 7)
patients_diets_and_blood_glucose_controls	(1994691, 7)	(1991552, 7)

3.4 Modifying ranges of esamilaboratorioparametri

We modify the ranges of some AMD and STITCH codes in the table esamilaboratorioparametri by clipping the values that are below the requested lower limit of the range, to that lower limit. Conversely, we clip the values that are above the requested upper limit of the range, to that upper limit. In this case we modify the ranges to the true ones of the requested codes.

This is an operation that doesn't involve other tables and, therefore, the dimensionality of all the tables remains the same as the one obtained in the previous subtask.

3.5 Cohort selection and label definition

We are asked to do three things in this subsection: to remove all the patients that have only 1 single event, to eliminate the patients that have a trajectory shorter than or equal to 6 months (i.e remove the patients whose all events happened within a period of 6 months), and label the patients.

For this, we created the trajectories of patients by joining all the events across all the tables. We then counted how many events we had for each patient and removed those who had only one event. None of the 49092 patients were removed, which means all of them had at least two events.

To remove the patients with a trajectory shorter than 6 months, we found the earliest (maximum) and latest (minimum) date, within the trajectories, for each patient. Then we removed the patients that didn't fulfill the requirement (i.e all their events were encompassed within a 6 months period). This caused our trajectories table to be reduced from 15.974.643 rows to 15.918.821 rows, for a decrease of 0,34%.

To label the patients with a 1, the patient must simultaneously fulfill two conditions: it must have had a cardiovascular event, and the event should have happened within the last six months of its history. We tested this last point by finding the earliest (maximum) date of the event of each patient, substracting 6 months from it, and checking if that patient has an event within that interval. Then if the patient fulfilled the two conditions simultaneously, it was labelled with a 1, and it was labelled with a 0 otherwise.

After this, we ended up with 32370 patients with label 0 and 15427 with label 1. This is a clearly unbalanced dataset, which is as we stated since the beginning, once of the most frequent challenges to overcome in this context of EHC analysis.

After the removal of the patients with a trajectory shorter than 6 months, we obtain the change in dimensionalities shown in Table 5

Table 5: Dimension of Tables before and after removing patients with shorter than 6 months trajectories

Table Name	**Before**	**After**
active_patients_info	(49093, 12)	(47797, 12)
diagnosis_tests	(1919045, 10)	(1909664, 10)
laboratory_tests	(7361304, 10)	(7337343, 10)
laboratory_tests_calculated_parameters	(2276064, 11)	(2268428, 11)
medical_tests	(290314, 10)	(289176, 10)
prescriptions_of_diabetes_drugs	(1986333, 9)	(1980368, 9)
prescriptions_of_non_diabetes_drugs	(150031, 7)	(149422, 7)
patients_diets_and_blood_glucose_controls	(1991552, 7)	(1984358, 7)

4 Task 2

4.1 Class imbalance 1

In this task, there are some steps that must only be applied to patients with label 1. However, the first thing we did, was to remove the last six months of history **for each patient, regardless of its label**. For this purpose, we identified the last event recorded for each patient (this means the maximum date) and set that as the "latest event" for each patient. Then, for each patient and by means of the *remove_last_six_months* function, we drop the events that took place in the last 6 months of history of the patient. Table 6 depicts how many observations each table has after we have removed the events that took place within the last 6 months of history of each patient.

Table 6: Change in dimensionality after removing last 6 months of data

Table Name	Shape before	Shape after
prescriptions_of_non_diabetes_drugs	(149422, 7)	(131316, 7)
prescriptions_of_diabetes_drugs	(1980368, 9)	(1783856, 9)
patients_diets_and_blood_glucose_controls	(1984358, 7)	(1753899, 7)
medical_tests	(289176, 7)	(261702, 7)
laboratory_tests	(7337343, 7)	(6488795, 7)
laboratory_tests_calculated_parameters	(2268428, 8)	(1990367, 8)
diagnosis_tests	(1909664, 7)	(1716216, 7)

After this, the following steps only applied to patients with label = 1

Patients with label = 1

We take the patients with label 1 and create one single copy of each of them. The reason why we created just 1 copy is that if we created more copies, we would instead of solving the unbalance problem, we would create a new unbalance where the class 1 is present in greater number than the class 0.

It is important to mention that, since we create exact copies of the patients with label = 1, we need to differentiate them from the original patients that gave birth to the copies, so that they are new patients, even though we know they are copies. For this, we decided to modify just the id center, in order to make the copied patients unique. We change the id center to consecutive numbers of 500, as this is the highest id center in the dataset and we want to still be able to differentiate between the copies and the original patients.

We need then to create the data, which is contained in all the other tables, for the recently created copies. For this we use the function *filter_by_replicas* which not only gathers the data for the copies but also shuffles the rows and drops a 0.05% of them, in order to introduce a little bit of randomness.

After all these steps, we end up with 32370 patients for the class 0 and 30854 patients for the class 1. **What was the number we had before the balancing for the label = 1**

4.2 Balancing strategy 2

As previously stated, after up-sampling the minority class through the creation of copies, we ended up with 32370 patients for class 0 and 30854 for class 1. We can consider this dataset as one which is **not** unbalanced.

However, since it is requested to implement another balancing strategy, **an undersampling of class 0 is carried out**. This was done by dropping, **at random**, 1516 patients of the class 0, in order end up with the same number of patients for both classes. Thus, we will work with 30854 patients for each class.

It is worth mentioning that this undersampling mechanism isn't the most effective one. Nonetheless, since this balancing isn't strictly needed, we decided to go for one of the simplest balancing strategies.

For the following Deep Learning Algorithms, were use data about the trajectories of the patients. As stated before, this is a compilation of all the events, across all the tables, that happened to a patient. It includes the patient identifier, the date of the event, the code of the event, the label and the age at the end (explanation forthcoming).

4.3 LSTM

For the LSTM Sak, Senior, and Beaufays (2014), given that it wasn't specified what it is understood by "Vanilla-LSTM", we decided to implement 3 versions of LSTM, which differ in its level of complexity.

To reflect this, we named them simple LSTM, medium LSTM and complex LSTM. Following the main characteristics of each of them:

1. **Simple LSTM**
   We encoded the code variable by using a label encoder. As features we took the identifiers of the patient and the code of the event (as it was previously explained). The size of the training set was 80% of the total dataset and the other 20% was taken as testing.

   The simple LSTM was defined as a sequential model with 64 units and a final fully connected layer of 1 unit with sigmoid activation function. These two hyperparameters were taken given the fact that the variable to predict is dichotomous. We used binary cross entropy as the loss measure and the adam optimizer.

2. **Medium LSTM**
   The medium LSTM has a higher complexity than the simple LSTM. This increased level of complexity was achieved by adding: first, a scaler over the features to help for the gradient convergence. Second, making the network denser by adding another layer with a higher number of units (128 units). Third, adding dropout layers with a dropping probability of 0.2. This with the aim of increasing the robustness of the model,increasing its generalization power and reducing the overfitting. Finally, we included early stopping also to prevent overfitting, although this will not be used in practice as we couldn't run the model for several epochs.

3. **Complex LSTM**
   We increased the level of complexity with respect to the medium LSTM by:

   - First, padding the input sequences. This is done with the aim not only of having more efficient computation and memory allocation during training, but also to boost the preservation of the sequence information by padding the shorter sequences with zeros and therefore retaining the temporal structure.
   - Second, adding bidirectional LSTM. This type of network processes input sequences, as suggested by its name, in two directions: forward and backward. It is formed by two layers working in parallel where one of them processes the beginning from the start to the end and the other one processes the sequence in the opposite direction. This is beneficial as we might more efficiently capture dependencies that span across the entire sequence. This might have a particular important effect in time series analysis, for instance.
   - We also tried to implement attention mechanisms and the use of embeddings as input sequences but it wasn't possible to achieve the desired implementation.

Now, two challenges were faced and motivated some decisions. The great amount of time needed to run the computations, coupled with continuous crashing due to lack of sufficient RAM memory, motivated the decision of just running the complex LSTM for 1 epoch. Additionally, we tested the simple and medium LSTM and attained results that were slightly worse than the ones attained by the complex LSTM. For these reasons, only the computation of the complex LSTM is shown.

Then, after 1 epoch, with a batch size of 128 and a learning rate of 0.01, **the complex LSTM achieved an accuracy of 82% over the test dataset**. This is considered a very good result, even more considering that the model was only trained by 1 epoch.

4.4 T-LSTM

For the implementation of the Time-Aware LSTM Baytas et al. (2017) we created the variable "age at event" which contains the age of the patient in years when the event happened. This feature aims to help the T-LSTM to be aware of the temporality in the input sequence, in a hopefully more intelligent way of capturing the hidden relationships in the sequence. With the same objective, we created the feature *days_since_min_date* which computes the days that have passed since the first record we have for each patient.

Then, we created a simple T-LSTM made up of 64 units followed by a fully-connected layer of 1 unit with sigmoid activation function. This is the replica of the simple LSTM. Given the difficulty of understanding what the T-LSTM was, it wasn't possible to implement a more complex version of this model.

We ran it with **batch size equal to 128** and default learning rate and attained **81,28% of accuracy**. This is, as in the case of the LSTM, a fairly good result.

It is worth noting that the T-LSTM attained a slightly lower accuracy than its counterpart the LSTM.

4.5 PubMedBERT

First thing that should be mentioned is that creating the embeddings with PubMedBERT is an incredibly computationally expensive process. Estimations based on how much time it took to create embeddings for 1 million registries, suggested creating the embeddings for the entire dataset would have taken more than 24 hours. This doesn't even account for the limited RAM resources provided by Google Collaboratory. For this reason, and being conscious that this isn't the optimal approach, **only the embeddings for the first 1 million entries are created.**

Then, we used the pretrained PubMedBERT Gu et al. (2020), specifically the version *microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract-fulltext* obtained from Hugging Face. As stated in the webpage, PubMedBERT is pretrained using abstracts of PubMed and entire articles from PubMedCentral.

Another important aspect to clarify is how we passed the inputs to the model. The patient records are passed as a string representation of each event, which is obtained by concatenating the id center, the id ana, the date and the code of the event. Even thought the original data had many more features than the ones that we ended up using, including some of these features would have increased the level of complexity of the models. We decided to first experiment with the simplest features and see if the model attains a fairly good accuracy by only considering them.

Moreover, with this input we created the embeddings by tokenizing, then using the mentioned model and applying an average pooling in the end. The output of this operation was saved in the csv *patient_data_with_embeddings.csv* which can be shared upon request.

Once the embeddings were produced, it was decided to use them to train two models, again one simple and another one slightly more complex: a logistic regression and a neural network, accordingly.

For the logistic regression we had to impute the missing values by the "most frequent" criteria. Not imputing the missing values caused repetitive errors in the code, until it was found that this was the root of the problems. The trained logistic regression was then used to make predictions and attained a 0.94 of accuracy. This is considerably high and higher than the other two models. **However**, given that we only took the first million registries, this result isn't trustworthy because we could have taken examples with just the same label and therefore the high accuracy of the model might not hold over new upcoming data.

For the simple neural network, we also imputed the missing values. A simple sequential model was implemented, which was made up of fully connected layers with ReLU activation function with 64 and 32 units, coupled with dropout layers with dropping probability of 0.2 and finished with a fully connected layer of 1 unit and sigmoid activation functions. We ran it with a batch size of 128, a learning rate of 0.001 and for 20 epochs and attained a **0.96 of accuracy**. However the must-have-caution note of the previous paragraph holds here too.

Table 7: Accuracy - Logistic Regression and Neural Network - PubMedBERT Embeddings

Model	**Accuracy**
Logistic Regression	0.942
Neural Network	0.9650

5 Task 3

5.1 Ignoring the order of microevents

We are requested to implement a way to ignore the order in which the micro events happened, while obviously conserving the order in which the macro events happened. In other words, we need to manipulate the data that is passed to the model such that it captures the temporal component of the macroevents (the order in which they happened) but ignores the temporal component of the microevents (therefore, not considering the order in which these happened). Hence, it appears that we have at least two options:

1. Completely ignore the microevents and drop them, so we just consider the macroevents.
2. Or we can aggregate/summarize those microevents, in order not to completely lose the information given by them, and attach these summaries to the macroevents. In this way, the aggregation of the microevents will "accompany" the macroevents.

We followed the tortuous second approach with the aim of not losing information that might give us predicting power.

Lastly, note that the data, which we have used through the project, doesn't contain numerical values. For this reason, the don't have a pletora of options from which to choose our strategy to accomplish this task, as we can't aggregate using minimum, maximum, mean, standard deviation, but just plain frequencies.

The Strategy

We are gonna group by patient, macroevent and date of the macroevent. In that order. And we are going to count frequency of micro events that match with the date of the macroevent.

By following this approach, we hope to capture the overall characteristics and temporal component of the macro events, while discarding the specific sequence in which the microevents happened between or within each macroevent.

However note that it is possible that some microevents happened when no macroevent happened. In this case, since we don't have a macro event date to match the microevent, instead of dropping the microevent, we search among the dates when macroevents happened for this patient, for the closest to the date of the microevent. Once we have found this closest macroevent date, we attach the microevent to that date.

To do this, we treated macro and micro events separately. Then for each microevent we put into a list format all the possible dates of macroevents that we have for that patient and we find the closest date between the date of the microevent and the dates within the just mentioned list. This is carried out in what we have called the "key step", which is a lambda function that takes at least 40 minutes to run and creates the "closest macro date". Then we group by patient and the closest date of macroevent and count the number of microevents.

This way we end up with a dataframe of 365.192 rows and 7 columns, where we have the macroevents that each patient suffered, together with its corresponding date. Additionally, the age of the patient when the event happened and the number of microevents that happened close to that date of the macroevent. We also put a zero in the number of microevents for those macroevents that didn't have a microevent that happened close to them.

We used the resulting dataframe to run the Bayesian optimization and the comparison between it and the LSTM and T-LSTM with default parameters.

5.2 Bayesian Optimization - Concentration

The Bayesian Optimization was carried out as the concentration item of this project. For this, the BayesianOptimization Nogueira (2014–) library was used.

It is important to mention that we applied Bayesian Optimization over the T-LSTM model only, as it takes massive amounts of time to run and we didn't consider strictly necessary to optimize the LSTM, for instance.

Now, we explored and optimized two hyperparameters through Bayesian Optimization: the number of units in the layer and the learning rate. For the first one we explored the interval between 16 and 128 units and for the second one we explored an interval between 0.001 and 0.01 for the learning rate.

The result of the Bayesian Optimization suggested that the **optimal number of units should be** 64 and the **optimal learning rate should be** 0.008.

We used these new values for the hyperparameters to run the T-LSTM. And we also ran the complex LSTM and T-LSTM with default parameters to then compare it with the optimized T-LSTM.

Now, the table 8 resumes the accuracy and the loss attained by the optimized through bayesian optimization T-LSTM, the T-LSTM with default parameters and the complex LSTM **over the test dataset**. This is an important consideration as the hyperparameter tuning must be done over the validation set and not over the test set.

Table 8: Accuracy and Loss Comparison between Models

Model	**Accuracy**	**Loss**
Standard LSTM	0.35	0.79
Standard T-LSTM	0.36	0.79
Bayes-Optimized T-LSTM	0.38	0.79

As shown by the table, the accuracy of the bayes-optimized model is higher than the other 2 models. However, it is clear that the difference is not massive. We could have expanded the intervals of number of units and/or learning rate, in order to have a bigger exploration landscape. However, due to the time limitation, we didn't proceed with this and decided to stay with these results.

The **confusion matrixes**, of these models are shown in Figure 1. We can see the bayesian optimization model is the best one to predict patients with a cardiovascular event (label 1), but it is the one that predicts more false positives. Naturally, the increase in the false positive is undesired, however it is less serious than producing false negatives, because of the context of EHC and cardiovascular events. Note that the bayesian optimization fulfills this, as it is the one with the smallest number of false negatives but the highest false positives.

References

Baytas, I. M., Xiao, C., Zhang, X., Wang, F., Jain, A. K., & Zhou, J. (2017). Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (p. 65–74). New York, NY, USA: Association for Computing Machinery. DOI: 10.1145/3097983.3097997

Gu, Y., Tinn, R., Cheng, H., Lucas, M., Usuyama, N., Liu, X., ... Poon, H. (2020). *Domain-specific language model pretraining for biomedical natural language processing.*

Keshta, I., & Odeh, A. (2021). Security and privacy of electronic health records: Concerns and challenges. *Egyptian Informatics Journal*, *22*(2), 177-183. DOI: https://doi.org/10.1016/j.eij.2020.07.003

Miriovsky, B. J., Shulman, L. N., & Abernethy, A. P. (2012). Importance of health information technology, electronic health records, and continuously aggregating data to comparative effectiveness research and learning health care. *Journal of Clinical Oncology*, *30*(34), 4243-4248. (PMID: 23071233) DOI: 10.1200/JCO.2012.42.8011

Nambiar, R., Bhardwaj, R., Sethi, A., & Vargheese, R. (2013). A look at challenges and opportunities of big data analytics in healthcare. In *2013 ieee international conference on big data* (p. 17-22). DOI: 10.1109/BigData.2013.6691753

Nogueira, F. (2014–). *Bayesian Optimization: Open source constrained global optimization tool for Python.* Retrieved from https://github.com/fmfn/BayesianOptimization

Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, *abs/1402.1128*. Retrieved from http://arxiv.org/abs/1402.1128

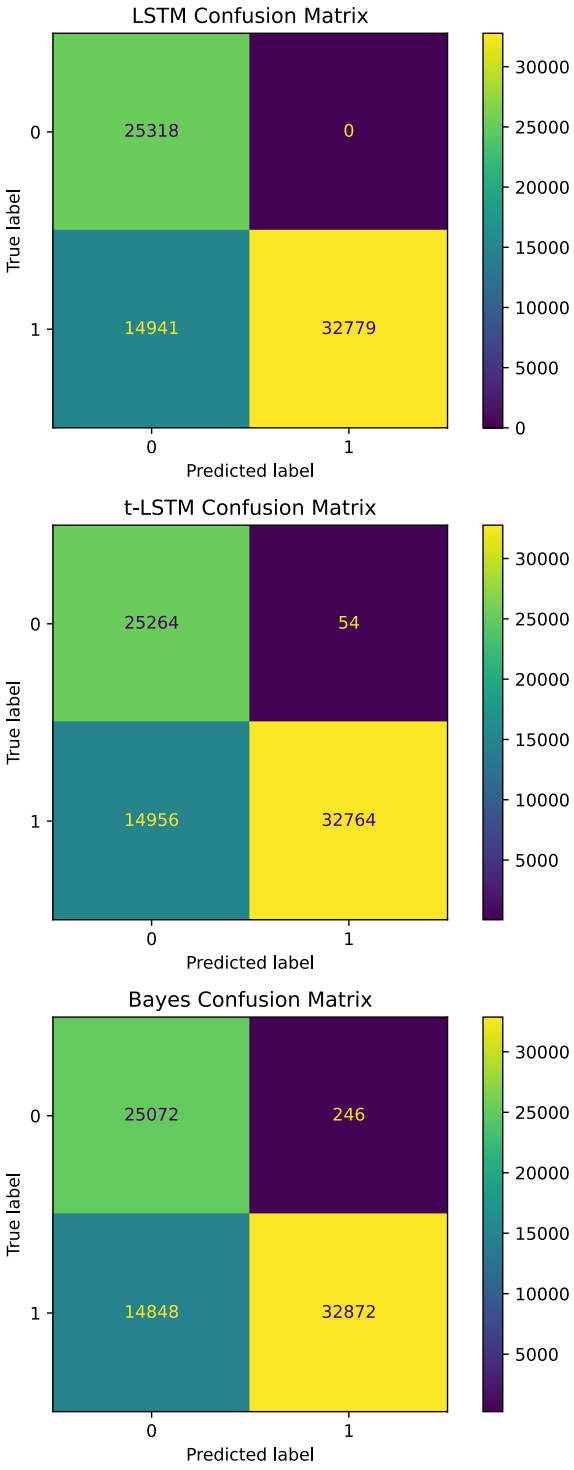

Figure 1: Confusion Matrixes