Machine Learning Project 2022/23

Ignacio Ávila Reyes[a], Julia Pérez Barreales[b], David Ramírez Palacios.[c]

[a]avilareyes.2081239@studenti.uniroma1.it

[b]perezbarreales.2057066@studenti.uniroma1.it

[c]ramirezpalacios.2056715@studenti.uniroma1.it

July 14, 2023

Abstract

A short (up to 250 words) summary of the main contributions of your project and the context of the research.

The following list is a guideline:

- Sort the authors is alphabetical order according to your surname
- Fill the sections in the **sections/** folder.
- Use the **bib.bib** file to insert bibtex resources for citing purposes if you try to implement a model that has been proposed in the literature. You can cite as Velardi and Madeddu (2021).
- **DO NOT** exceed the three page limit per section.
- Include only .svg or .pdf files for the images in the **images/** folder. **DO NOT** include .png/.jpg images: they lose quality when zoomed in.
- When describing the prediction method, **DO NOT** include code snippets. I can see the code from the source files you'll be delivering. Instead, provide an architectural description of your model if it's not a simple one (e.g., SVM, Decision Tree, Random Forest, Naive Bayes).
- The language of the project must be **ENGLISH**.

1 Introduction, context, and motivations

For quite some time the field of Machine Learning has been expanding exponentially and is constantly being researched and more developed. Because of the growth of the volume of medical data and the digitalization of these, Machine Learning presents a unique opportunity to improve and make quicker diagnosis based on a large volume of information.
The motivation behind these project is take advantage of these digitalization of medical data and use it to train a model that is capable of predicting if a patient (that we have the medical information of) is going to have a major cardiovascular event in the next 6 months.

2 Dataset description

The dataset that we are provided with consists of seven files. Each one of the following files is a table containing relevant information for maing a diagnosis of heart problems for the patient. For identifying the patients, we have a unique key in each table formed by the pair of attributes *idcentro* and *idana*.

2.1 anagraficapazientiattivi.csv

This table contains personal information of the patients.
Particularly, aside from the key for identifying patients, it provides us with the *gender* of the patient, the *year of the diagnosis of diabetes*, the *type of diabetes*, the *education level* of the patient, their *civil status*, their *profession*, their *origin*, the *year of birth* of the patient, the *year of first access* of the patient and the *year of death* of the patient (in this last column we have assumed that the value NaN indicates that the patient is still alive).

2.2 diagnosi.csv

This table contains diagnosis tests. It contains the key for identifying patinets, and another three columns: The column *data* that indicates the date of the diagnosis; the column *codiceamd* that indicates what was being tested and the column *valore* that indicates the result of the test.
In the column *codiceamd* the data is of the form "AMD" followed by a number, we have been provided a table explaining which test corresponds to each of the numbers.

2.3 esamilaboratorioparametri.csv

This table contains information about tests performed in laboratories. It has the same attributes as the previous table.

2.4 esamistrumentali.csv

This table contains information about medical tests. It contains the same attributes as the tables *diagnosi.csv* and *esamilaboratorioparametri.csv* with a significant difference.
In the other two tables, the column *valore* contains numerical data, while in this table, this column takes the values 'N' and 'P'.

2.5 prescrizionidiabetefarmaci.csv

This table contains information about the prescription of medication for diabetes.
Particularly, it contains the key for identifying the patients, the date of the day when the medicine was prescribed (in the column *data*), the drug that was prescribed (in the column *codiceatc*) and the dosis of the medicine (in the column *quantita*), the column *idpasto* and the medical description of the drug (the column *descrizionefarmaco*).

2.6 prescrizionidiabetenonfarmaci.csv

This table contains information about blood glucose controls and diets assigned to patients. It has the same attributes as the tables *diagnosi.csv*, *esamilaboratorioparametri.csv* and *esamistrumentali.csv.*
However, in this table, the data in the column *valore* take values 'NaN' or 'S'.

2.7 prescrizioninondiabete.csv

This table contains information about the prescription of non diabetes medicines. It has the same attributes as the previous table but the column *valore* takes different values.

3 Task 1 description

Since the dataset contains a large volume of data, the purpose of *Task 1* is to preprocess the data containing active patients (which are the only ones we are interested in).
In order to do this, we perform the following operations on the data:

3.1 Exercise 1: Select events of interest

For this exercise, we have made a list of the amd codes of the cardiovascular events, and, since we have realised that all of them only apper in the diagnosi table, we create a new table with only the cardiovascular events of the diagnosi table.

Then, we use this new table and the patients table to obtain only the patients with at least one cardiovascular event.
Lastly, we eiminate the entries of the patients we are not interested in of the rest of the tables.

3.2 Exercise 2: Invalid feature cleaning

First, we clean the patient's table.
To do this, we check that the patient's birth year, the year of the diabetes diagnosis and the year of their first access is not NaN.
Then, we check that these three years are all smaller than their year of death, that the year of birth is smaller than the year of the diabetes diagnosis and that the year of birth is smaller than the year of their first access.
After this, we eliminate the entries of patients with invalid dates in the rest of the tables. Furthermore, we check that the dates in the tables are valid and that they are between the year of first access and the year odf death of the patient. Particularly, for the prescrizionidiabetefarmaci table, we check that the prescriptions are made after the year of the diabetes diagnosis of the patient.

3.3 Exercise 3: Remove patients with all dates in the same month

First we convert the column *data* of each table (that has a column *data*) into datetime type and we calculate the maximum and minimum dates for each table and every patient.
Then, for each patient we take the maximum date of all tables and the minimum date of all tables and we compute the difference.
We take from the patients table only the patients with a difference greater than 30 days and then eliminate the entries in the other tables of the patients that have just been eliminated.

3.4 Exercise 4: Modify the actual ranges of *esamilaboratorioparametri*

We use a MinMaxScaler from *sklearn* library to make the values fit into the ranges provided, except for the NaN ranges, where we do nothing.
Then we check that the scaling has been successful.

3.5 Exercise 5: Cohort selection and label definition

We take the dataframe tat we already calculated in the first exercise that contains the cardiovascular events of the patients. From this table, we take the minimum date and the last two dates of the patients that have more than one event.
If the difference between the last event and the first event of the patient is smaller than 180 days, we ignore the patient.
If not, we calculate the difference between the latest two events of the patient, if the difference is greater than 180, we label the patient as 1; if not, we label the patient as 0.
Lastly, we eliminate from the tables the entries of patients that are not labeled.

3.6 Exercise 6: Concentration

For improving the quality of our dataset, we are going to eliminate the features that do not provide relevant information.
We have considered that if 70% or more of the data of the column is equal to 'NaN', then it does not provide relevant information and we eliminate the feature.

4 Task 2 description

After processing the data and labeling the patients, we realise that the dataset is imbalanced. The aim of *Task 2* is balancing the class distribution.
To do this, we first upsample the minority class, after this, we perform another balancing strategy and then evaluate different prediction models with the balanced dataset:

4.1 Exercise 1

First, we divide all the dataframes into two (one for the events of patients of class *0* and the other for the events of patients of class *1*).
Then we eliminate the events of the last six months for each patient (both class *0* and class *1*).
To upsample the minority class (which is class *1*), we are going to create m copies of each patient in class *1* where each copy has random events from the patient. To difference these copies, we define a new attribute *idcopy*.
This means that two patients that have the same values in the patient's table (except for *idcopy*), will have different events in the rest of the tables.

4.2 Exercise 2

4.2.1 Balancing Strategy

For our balancing strategy, we are going to count the number of cardiovascular related events of each patient and we are going to take only the patients that have a number of cardiovascular related events greater than a threshold.
With cardiovascular related events we are refering to events that affect directly to cardiovascular problems such as diabetes and cholesterol but also other events that are not decisive but propitiate cardiovascular problems such as if the patient smokes or the patient's weight.

4.2.2 Vanilla-LSTM

First we convert the datasets so that they can be processed by the model we are going to create next.

Then we proceed to the creation of the model using the Keras sequential library. We use two fully connected layers and the sigmoid function as an activation function. In addition, we set the Binary Cross Entropy as the objective function and the Binary Accuracy as the accuracy function.

Then we train the model, adding each time the events related to a new patient, similarly to batching.

Finally, we evaluate the model over a randomly defined batch of data.

4.2.3 T-LSTM

First, we define our LSTM model (the weights and biases of the input state, the forget state, the output state and the cell state) and the operations to be performed to the input data that we have seen in the theory.
Now we have to transform our data to use it for the model:
We concatenate the dataframes of the patients in class *0* and the patients in class *1*.
Then we merge the patient's dtaframe with the dataframe of the macroevents and we use a function to convert our key *(idcentro, idana, idcopy)* into a unique id.
After this, we reorder the columns and drop the columns that we had previously used as key. We categorize the columns that contain text and convert every column into float type.
Then we divide the data into three groups: the dates, the labels and the rest of the data; we create a list of batches and then we train the LSTM model with this batches.

4.2.4 PubMedBERT

We also need to transform our data for the PubMedBERT model.
First, we add to the dataframes with the attribute *codiceamd* or *codiceatc* a new column $comment_text$ that expresses in the natural language the correspondant AMD or ATC code of the event.
Then we eliminate the NaN values of the tables and, for each patient, we create a paragraph written in the natural language that describes the information that we have about the patient in all of the dataframes.
After this, we install the packages needed for using the PubMedBERT and divide our dataset into a train set, a validation set and a test set.
Lastly, we train our model with batches from the train set and then validate it with the data from the validation set.

5 Task 3 description

The objective of the *Task 3* is to choose a prediction strategy according to the changes that we have done in the previous two tasks and taking into account that the order of the macro events of a patient are important but the model does not need to learn the order of the microevents.

5.1 Exercise 1

We are going to use a Gated Recurrent Unit as model in order to take into account the order of the macroevents.
As we did with the *T-LSTM* model in *Task 2*, first we need to transform our data to use it for the model: We merge the patient's dataframe with each of the other dataframes and then we concatenate all of these dataframes.
After this, we use a function to convert our key *(idcentro, idana, idcopy)* into a unique id, we reorder the columns and drop the columns that we had previously used as key.
We categorize the columns that contain text and convert every column into float type. Then we divide the data into *Ytrain* which only includes the *label* column and *Xtrain* which is the dataframe without the columns *id*, *data* and *label* and we transform these dataframes to tensorflows.
Finally, we construct our GRU model with three GRU layers, each one with 50 units and the function *tanh* as the activation function, and an output layer.

References

Velardi, P., & Madeddu, L. (2021). Aim in genomics: Ontological and connectivity structure of disease-gene modules in the human interactome. In *Artificial intelligence in medicine* (pp. 1–15). Springer.