**Lab2 Task duration determination**

**Mathys LEBON**

1. **Problem**

Task management is an important part of software development and a huge part of the work of the project manager. The idea is to find a machine learning solution that can make this task easier for managers. Firstly, the objective was to be able to generate automatically an entire Gantt chart. But the first step in order to realize this is to be able to determine the duration of a task automatically so this lab will focus on this goal.

1. **Data collection**

In order to train our machine learning model, we need to get some data, with inputs and outputs.

But this kind of data are private, relative to companies information, so it’s hard to find dataset with this information. More than hard I couldn’t find a dataset interesting enough to be use so I need to find another solution.

This solution was to simulate a dataset with the inputs and the output that I want. This solution is not the best one, but this is the first step in order to train the model and be able to know if this approach could be interesting or not. If yes, we could in a second time try to get some real data from some companies of software engineering old project.

Here is the list of all inputs and their type:

* Description: A string which describes the goal of the task
* Technologies: A string which is the name of a programming language
* Type: A string that represents the type of the task (bug, feature, refactor, test or documentation)
* Start: A date of the beginning of the task
* Deadline: A date of the last date possible for the task
* Team: An integer with the number of people available for the task
* Priority: An integer which represents the priority of the task   
  (1: low, 2: medium,3: high)

The output is “duration” and is an integer representing the number of days needed for the task to be completed.  
First all the input were generated randomly in some predefine and realistic ranges.  
And in order to generate the duration I work with two values: *min\_duration* and *max\_duration* and the duration was a random value between these two ones. *Min\_duration* is modified with some rules that I create about the inputs generate.  
Some rules use:

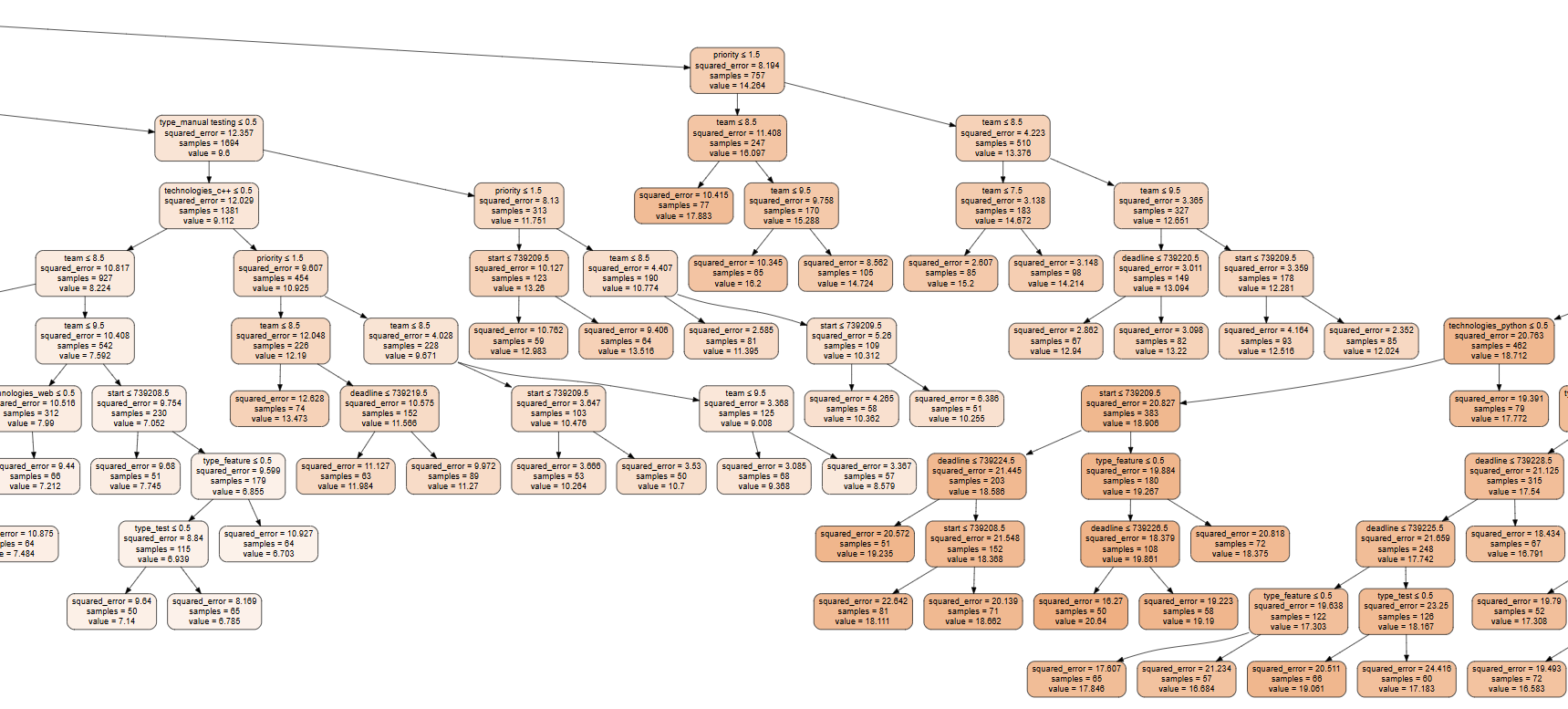
* Min\_duration is longer if the team is shorter
* Min\_duration is pondered with the type of task
* Min\_duration is pondered with the technologies use
* Min\_duration is pondered with the length of the description
* Max\_duration is pondered with the priority of the task

Same for the *max\_duration* that varied in function of the priority of the task.

This solution was interesting, but the type and technology weren’t linked together, and the description was just a list of random words. I decided to use ChatGPT to generate a csv file with description, type and technologies all related together and use this one to generate even more data. This file is *“data/IT\_tasks\_list\_with\_logic.csv”*. This allow to have a link between the different input, desc,type and technologies. They were after use to generate new random task with the other inputs necessary.

1. **Machine learning approaches**
   1. **Which methods**
      1. **Decision tree**

A decision tree will make a prediction with a succession of yes/no questions in order to split the data in two categories at each time. Here it will be used to split into n categories at the end. Each category is a number representing the duration of the task.

*Figure 1: A part of the final decision tree*

* + 1. **Linear regression**

The idea of using the linear regression is to determine if a simple model can give us interesting results too. Linear regression is a method which try to find a relation between the output and the inputs using some coefficient in front of each input assuming it following a linear relationship.

* 1. **Implementation details**

The implementation was done using Python and more especially Jupyter Notebook, allowing me to create a work environment with title and description of the different steps.  
The library SkLearn is used in order to implement the two machine learning models.  
For the decision tree three hyperparameters are important to set up, *max\_depth*  the maximum number of yes/no question in a row, *min\_samples\_split* the minimum number of sample by split for each yes/no question, and then *min\_samples\_leaf* the minimum number of sample in the last answer. In order to determine the best value for these hyperparameters the use of a gridsearch is interesting, it will perform a decision tree with each set of value from list for each hyperparameter and keep the best result. This allows us to find the best hyperparameter.  
Here are the best hyperparameter: max\_depth : 15, min\_samples\_leaf : 50, min\_samples\_split : 2.

1. **Results**

In order to determine the quality of the prediction the Root Mean Squared Error (RMSE) is used. This represents the square root of the averages squared differences between the predicted values and the actual values.

Here is the formula:

*Figure 2: RMSE formula*

With n is the number of samples and is the real value and is the predicted value.

|  |  |  |
| --- | --- | --- |
| RMSE | Tree model | Linear model |
| Best result | 3.4933608294240104 | 3.432229596049775 |

*Figure 3: RMSE results*

This shows that linear model is the best model using only this metric, for a little margin.

But this value alone doesn’t represent everything the use of the graph of predicted/actual values.

The model will fit the best with reality, if points follow a 45° line (the red one).

A graph of a graph

Description automatically generated with medium confidence

*Figure 4: Predictions vs True values*

Again, these graphics show that both model give pretty close results and tend to follow the 45° line which is a good point. Decision tree model has more isolated values this can show that he is less reliable than the linear regression.

Finally, a third way to compare the model is used, representing the distribution of the difference between the real value and the predicted one

A blue and white graph

Description automatically generated*Figure 5: Distribution of the difference between actual and predicted*

This give us the same conclusion than the other metrics, models are close but the tree model tend to have a largest range of different values which show again a reliable problem. We can also conclude that even if models don’t find the exact value a majority of the time the error is less than 3 days.   
We can conclude that the simple model of linear regression give us better results. But with an RMSE which is not perfect, and the model can maybe be improve or use another one can be interesting.

1. **Future improvements or research**

Using different machine learning techniques can be interesting and give the best results.

Research about the interesting inputs and maybe new ones can possibly help the solution.

Another point of improvement is now that we are able to determine with an interesting precision the duration of tasks, to be able to select and assign automatically team members to different tasks, knowing their specialties and qualities, for example.

The determination of not only duration but also effort and links between tasks, generating automatically all the Trello/timetable.