POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Time prediction of software development via machine learning

Artificial Intelligence applied to Software Engineering

Supervisors
prof. Maurizio Morisio

Candidates
Jacopo Nasi [255320]

Internship Tutor
dott. Davide Piagneri

Anno accademico 2019-2020

Summary

The software development is fundamental in the new world, how about using artificial intelligence to improve it.

Acknowledgements

Un ringraziamento speciale ai cavalieri di Smirnuff, luce della mia battaglia.

Contents

Chapter 1

Introduction

1.1 General Problem

Forecasting is one of the most critic part of a company, it could drive to easily success as well as drive to failure. A software project is not different from a manufacturing product, its development, infact, require analysis of different kind, from resources needed to costs and time required.
The software development experience shows that the process of analysis is really difficult, due to the nature of the problem, coding is a mind product and the time required to produce it can varying in accord to a lot of different factors.

1.2 Tools used

This work is mainly conducted using software tools, here a list of the tools used:

Python The main programming language of the thesis project. Used for data management, feature extraction, machine learning models and for interfaction with other softwares. The specific version used is the v3.7.0

Pandas Open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

NumPy Scientific computing with Python.

Matplotlib 2D Plotting library for Python.

Seaborn Another plotting library for Python.

Tensorflow Platform for machine learning.

Keras High level API for neural networks.

SciKit-Learn Tools and libraries for machine learning.

GitLab Sourcing platform based on Git. Used for the code of the project, available here:
https://gitlab.com/EiS-Projects/analytics/temp/thesisProjectJN.

GitHub Sourcing platform based on Git. Used for the thesis and calendar sourcing:

- Thesis: https://github.com/Jacopx/Thesis
- Calendar: https://github.com/Jacopx/ThesisCalendar

JetBrains IDEs Student-free IDE for different language development, product used:

- PyCharm: https://www.jetbrains.com/pycharm/
- DataGrip: https://www.jetbrains.com/datagrip/

Chapter 2

State of art

2.1 Related works

Speaking about other works.

Chapter 3

Datasets

The following section illustrate the structure of the all the principal datasets used during this thesis project.

3.1 SEOSS33

The SEOSS33[1] is a dataset collecting bug, issue, reports, commit and lot of other information of 33 open source project, following their progress via sourcing platform. The dataset is enriched also with time stamps, release versions, component information and developer comments.
At today there are no other public research conducted over this datasets, this works seems to be first.
The data is retrived by the issue tracking system (ITS) and the version control system (VCS), the process is shown in figure 3.1.

To unify the project specific difference, the typed issue, e.g. *New Feature* or *Bug Report*, are mapped to five issue categories:

- Bug: A problem which impairs or prevents the functions of the product
- Feature: A new feature of the product
- Improvement: An enhancement to an existing feature
- Task: A task that needs to be done
- Other: Various

The study collects 33 projects that are using Atalassian Jira and git, for popularity reasons. Is also required that the projects in the dataset should be majorly written in one programming language, Java is choosen. Due to machine learning nature, the choosen project must have great number of issue, all project were in

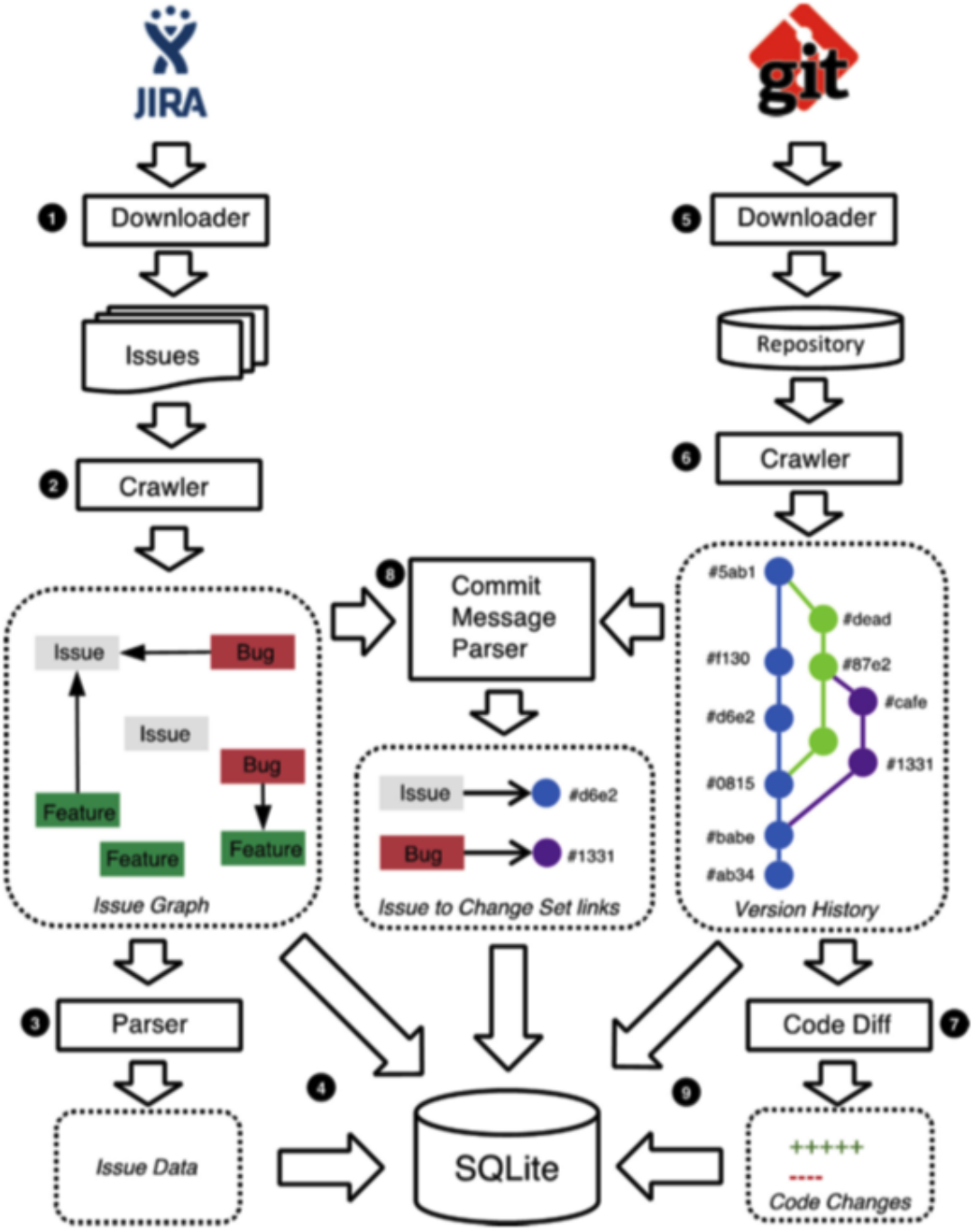

Figure 3.1. SEOSS33 data retrival

development for at least three years. Among these products we have selected five

of them, because of size, as shown in table 3.1:

Table 3.1. Project data distribution

Project	**Month**	**Issue**
Hadoop	150	39086
Hbase	131	19247
Maven	183	18025
Cassandra	106	13965
Hive	113	18025

More selection charactheristics can be found in chapter 2.1 [1].
Is fundamental to understand the structure of this dataset, the majority of the forecasting operation tests are conducted using the data stored by this research.
Each project is stored in a SQLITE file, a SQL offline database, the structure is based on the entity of the *issue*, identified by an *issue_id*, the other tables are used to link additional information, like the number of commit, the version referred, comments and others features. The figure 3.2 show the database schema.

Each table contains different types of data that will be used to generate the data used to forecast.

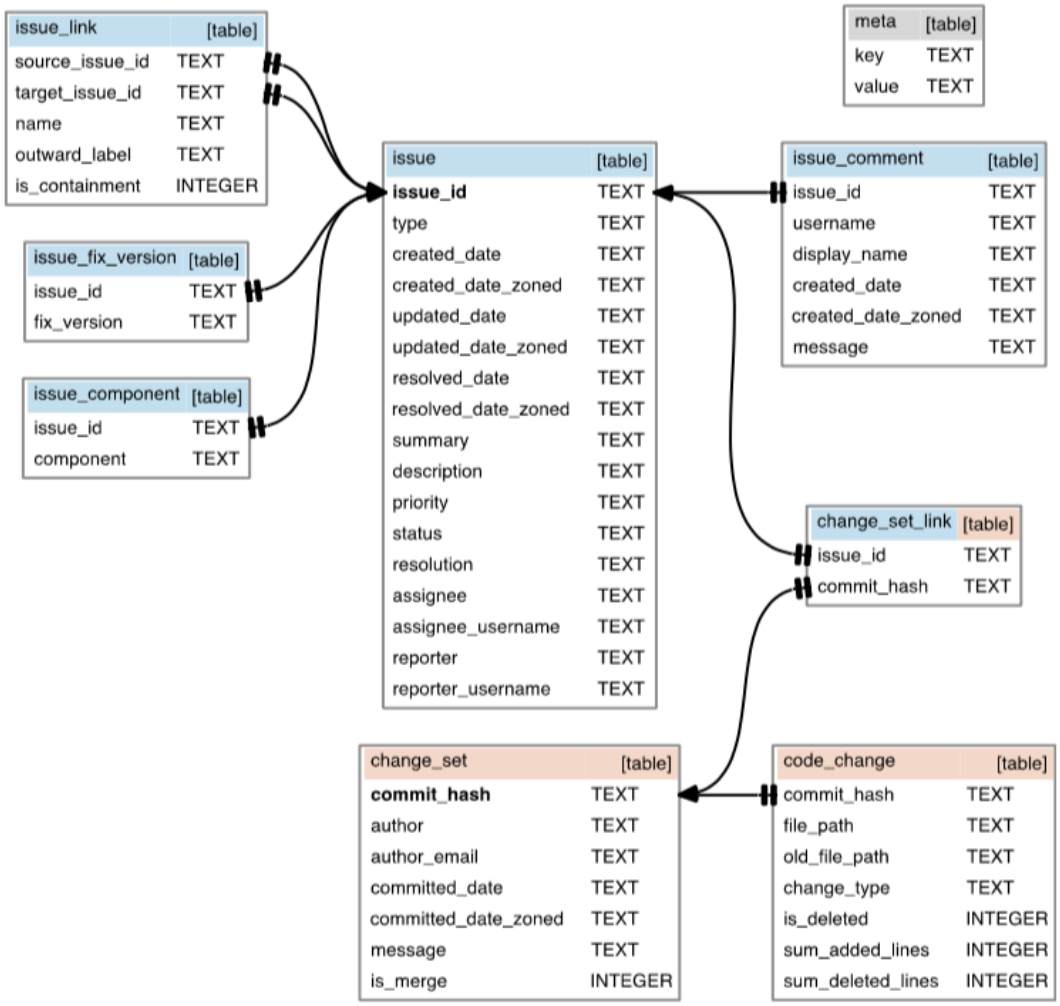

Figure 3.2. SEOSS33 data model

Chapter 4

Machine Learning

4.1 Introduction

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. The word ML is almost in the public domain now, in the last decades the usage of these kind of algorithms has drammatically raisen although most of it had already been developed for years. The main reason is the increase in the computational capacity of the systems.
There are two types of systems:

- Knowledge-based: Acquisition and modeling of common-sense knowledge and expert knowledge (from rules to facts)
- Learning: Extraction of knowledge and rules from example/experience (from facts to rules)

All the machine learning techniques try to develop systems similar to the second definitions. ML algorithms can be divided in three categories: supervised, unsupervised and reinforcement learning. The first will be used to predict class based on previous knowledge, the second tries to labeling data without a priori experience and the third bases its learning on action reward.
There a lot of different models available, the following chapters will focus on the models used in this project.

4.2 Supervised Learning

Is a powerful technique to process labeled, which is a dataset with that has been classified, to infer a learning algorithm. These dataset is used as basis to predict, and learn how, unlabeled data. There are two types of supervised learning,

classification and linear regression. The goal of classification models is to predict categorical class labels of new instances, based on a training set of past observations. The classification can be binary or multi-class. Instead, regression, aim to predict continuos outcome, it tries to find mathematical relationships between variables to predict the target with a reasonable level of approximation. Our project is only focused on regression, classification will not be further discussed.
All the supervised learning method used are based on ensable, the basic idea is to merge multiple, different, hypotesis in order to improved the quality of the prediction, for example the random forest or gradient boosting are ensable of multiple decision trees. The following paragraphs will deal with the models used specifically.

Random Forest Random Forest (RF) is a supervised learning algorithm which uses ensable learning method for classification and regression. They are built by combining the predictions of several trees, each of which is trained independently and the prediction of the trees are combined through averaging [2]. A visualization of a RF is shown in figure 4.1.

Three parameters are important defining the RF, (1) the methods for splitting

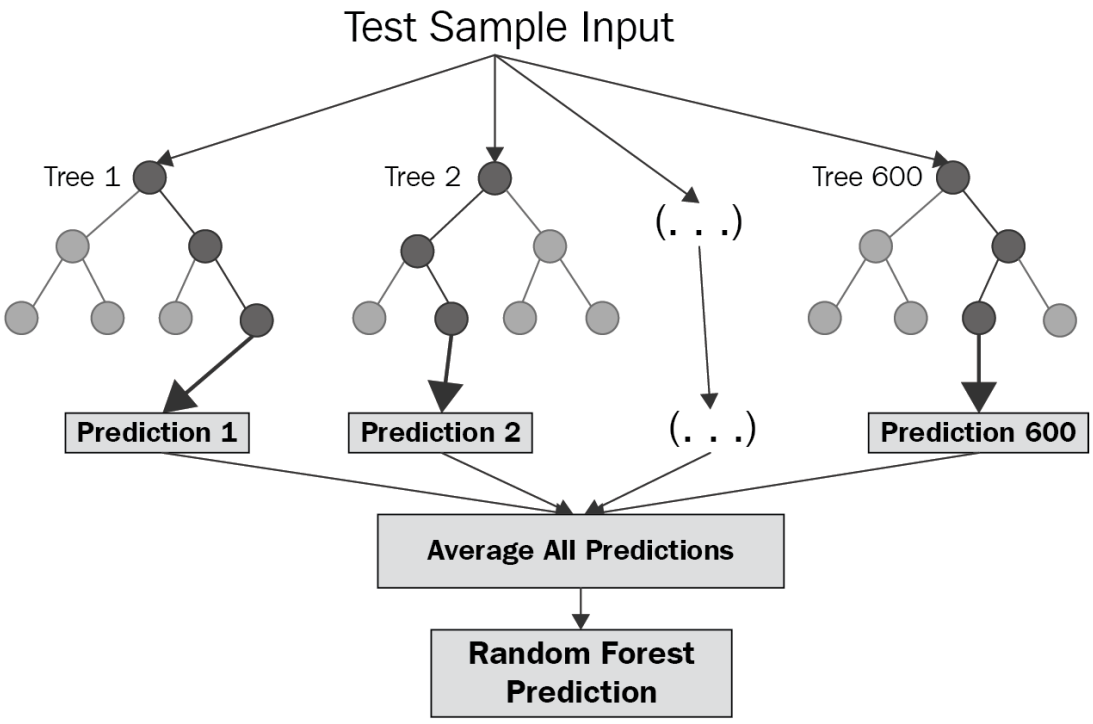

Figure 4.1. Random Forest simplified scheme [3]

the leafs, (2) the type of predictor to use in each leaf, and (3) the method for injecting randomness. Splitting of require the selection of shape and a method for evaluating the quality of each candidate. Typical choices are axis aligned splits, where data is routed to sub-tree depending on a threshold. The threshold can be chosen randomly of by a leafs optimization function. After the generation of different candidate splits a choosing criterion must be defined in order to split the leaf. One of the simplest strategy is to choose among the candidates uniformly at random, otherwise is possible to select the candidate split which optimize a purity function (information gain for example) over the leaf that would be created. The most common predictors, for each leaf, is the average response over training points which fall in that leaf. The injection of randomness can happen in different ways, the size of candidate splits, coefficients for random combinations, etc... In any case the thresholds can be also defined randomly or by optimization over data. Another solution is build tree using bootstrapped or sub-sampled dataset.
The training phase is performed independently by each tree by exploiting an assignement in structure and estimation points to respectively change the shape of the tree and to improve the estimator fit.
Once the forest has been trained it can be used to make predictions for unlabeled data points. In the prediction phase, each single tree, independently make its own prediction than an average of all the trees is computed to make a single outcome value. The contribution of each tree to the final value is the same.
Out implementation use Random Forest developed by SciKit-Learn v0.21:

```
from sklearn.ensemble import RandomForestRegressor
```

The specific parameters will be explained during the chapter 5 about forecasting.

Gradient Boosting Machines Gradient Boosting Machines (GB) are a family of powerful machine learning techniques that have shown considerable success in a wide range of pratical application. They are higly customizable to the particular needs of the application, line being learned with respect to different loss functions [4]. Techniques like RF rely on simple averaging of model in the ensable. The family of boosting methods is based on a different constructive strategy of ensemble formation. Boosting add, sequentially, new models to the ensable. In GBM the learning phase consecutively fits new models to improve the accuracy of the estimations. Ideally they construct new basic layers, decision trees of fixed size, to be maximally correlated with the negative gradient of the loss function of the ensable. The loss function should be chosen by the developer, even ad-hoc loss function could be implemented. The attempt is to solve this minimization problem numerically via steepest descent [5].
This high flexibility of GBM makes them really customizable to any kind of task. Given its simplicity a lot of experimentation can be performed over the model.
Our project implement Gradient Boosting Decision Tree (GBDT) developed by

SciKit-Learn v0.21:

```
from sklearn.ensemble import GradientBoostingRegressor
```

4.3 Neural Networks

Neural Networks (NN) are algorithm, for pattern recognition, inspired by the structure of the human brain, with elaboration units (neurons) and connection network (synapses), exposed to enough of the right data, this kind of algorithms is able to establish correlations between present and future events. Figure 4.2 shows a simplyfied versiona of a neural network.
NN can be used to solve different kind of problems, classification, clustering and regression. Our problem will be solved using the regressive type.
Regression analysis can be used to forecast one or more label based to other features. The structure of these networks can be really complicated and a whole thesis can be made upon this topic that, for this reason, will not more discussed.

Recurrent Neural Networks Recurrent Neural Networks (RNN) is a class of NN that keep connections between nodes and a temporal sequence. The main difference, respect NN, is that has feedback connections, this memory allow to keep track of temporal dynamic behaviour, they can process single data point or entire sequence of data, like video or speech.

Long-Short Term Memory One of the most used type of RNN is the Long-Short Term Memory (LSTM), were developed to solve the problems of exploding and vanishing of gradient typical of normal RNN.

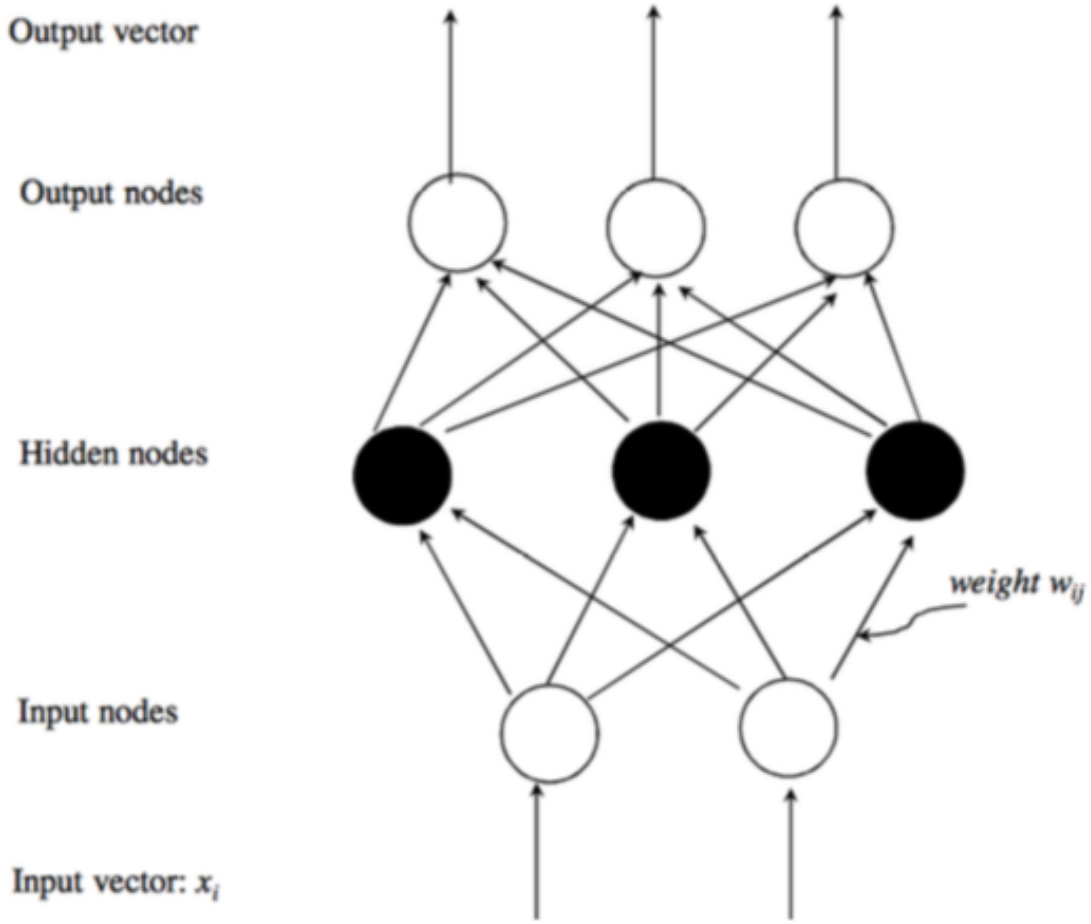

Figure 4.2. Structure of a neural network

Chapter 5

Forecasting

5.1 Introduction

Forecasting is the process of making predictions of future based on past and present data by trends analysis. Forecasting is one of the most desired machine learning functionality, it could be used to improve each kind of process, from finacials to production ones. Of course this task is not easy to achieve, a lot of resources and studies are needed to accomplish it. The software development is identical to a product development process, starts from the ideation and ends with the production itself. The goal is to predict the defectiveness in order to efficently allocate the development effort.

5.2 Features

The main advantage, in data analysis, of machines is that they can compute a lot of different data and finding a lot of patterns and correlation that human can't find. Combine the human attitude of logical correlations and machines capacity of number analysis can drive to a powerful combination that can drastically improve the forecasting ability. Each artificial intelligence algorithms require a correct and properly studied data in order to perform a valuable prediction, one of the basic step is the data preparation, providing correct and organized data is fundamental to correctly fit the network over the problem.

5.3 Models detail

5.4 One-Shot Prediction

5.5 Recurrent forecasting

5.6 Results

Chapter 6

Conclusion

Bibliography

[1] M. Rath, P. Mäder, "The SEOSS 33 Dataset — Requirements, Bug Reports, Code History, and Trace Links for Entire Projects" in *Data in Brief*, v. 25, p. 104005, 05 2019. [Online]: https://doi.org/10.7910/DVN/PDDZ4Q

[2] M. Denil, de Freitas, in "Narrowing the Gap: Random Forests In theory and In Pratice"

[3] "Random Forest and its implementation." [Online]: https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

[4] A. Natekin, A. Knoll, "Gradient boosting machines, a tutorial" in *Frontiers in Neurorobotics*, v. 7, p. 21, 2013. [Online]: https://www.frontiersin.org/article/10.3389/fnbot.2013.00021

[5] SciKit-Learn, "Ensable methods in Sci-Kit." [Online]: https://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting