Advanced Softwareengineering

Football World Cup Predictor

Github link:

Denis Baskan – 878 571

01 September 2018

Inhalt

[Introduktion 2](#_Toc523591046)

[Crawl websites 2](#_Toc523591047)

[Predictor 2](#_Toc523591048)

[Football tournament 3](#_Toc523591049)

[UML 4](#_Toc523591050)

[Use Case Diagram 4](#_Toc523591051)

[Package Diagram 5](#_Toc523591052)

[Activity Diagram – Web Scraper 6](#_Toc523591053)

[Activity Diagram – Predictor 7](#_Toc523591054)

[Metrics 8](#_Toc523591055)

[Clean Code Development 9](#_Toc523591056)

[Continuous Delivery 10](#_Toc523591057)

[AOP 11](#_Toc523591058)

[DSL 12](#_Toc523591059)

[Functional Programming 13](#_Toc523591060)

[Logical Solver 14](#_Toc523591061)

[Scala / Clojure 15](#_Toc523591062)

# Introduktion

In football world cup 2018 we could watch many exciting matches. Although Germany has performed very bad, I was motivated to develop code in order to predict scores. Therefore, we need data.



Source: <https://www.kuredu.com/wp-content/uploads/2018/06/fifa.jpg>

## Crawl websites

What data could be relevant for our predictor? Football players usually play not more than three world cups. So my suggestion was to take many matches such as past world and european cups, U20 and U21 matches, qualifying and other matches.

The domain <http://www.weltfussball.de/alle_spiele/> and their sub-domains provide many matches. As I wrote the code snippets, the data of qualifying and other matches were available on <http://www.sportdaten.t-online.de/fussball/> . Unfortunately, this website is no longer accessible. But I’d still had the data stored on my computer.

## Predictor

I decided to use an artificial neural network (ANN) to get some predictions. Inputs are two nations, year and kind e.g. word cup, european cup and so one. Outputs are two score vectors that contain probabilities. The two highest probabilities are our predictions for a given match.

## Football tournament

You might have played a little tournament with your friends and family where points are given in respect to your predictions. If you want to compare your predictor against some of your revivals there is a function to evaluate your results.

# UML

## Use Case Diagram

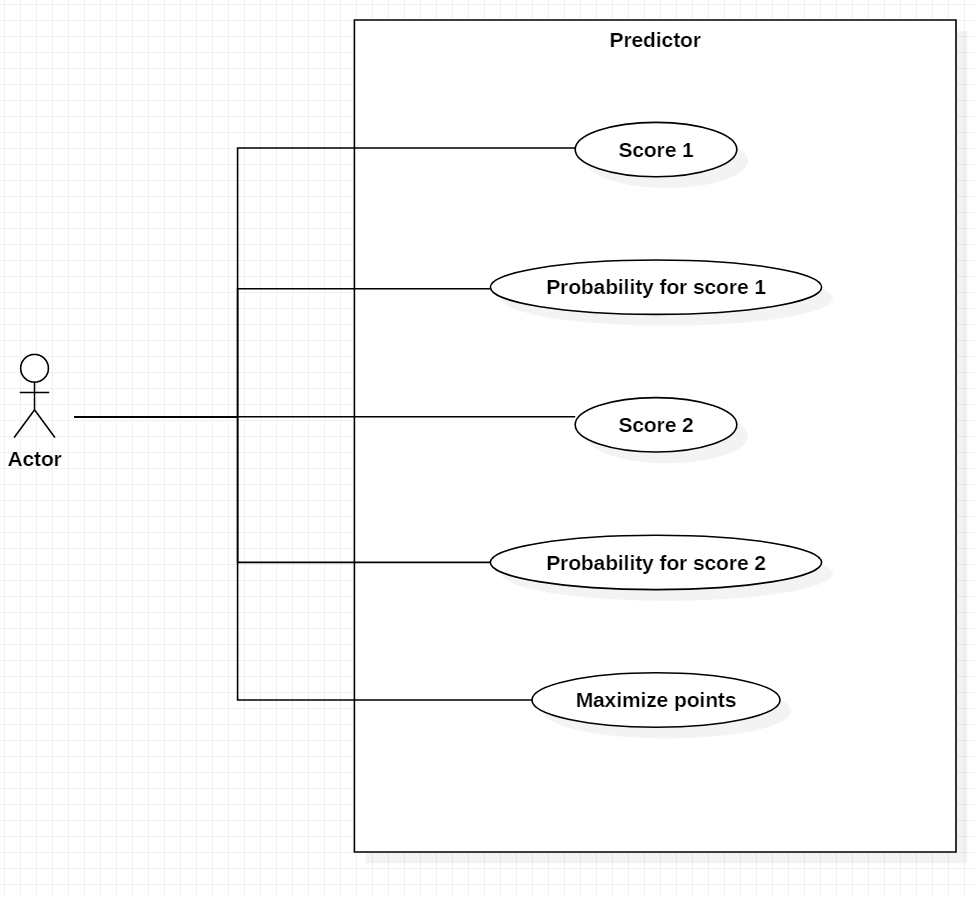


Figure 1: Use Case Diagram

The actor wants to get predictions for the scores for the next match. He also wants to know how confident the predictions are.

## Package Diagram

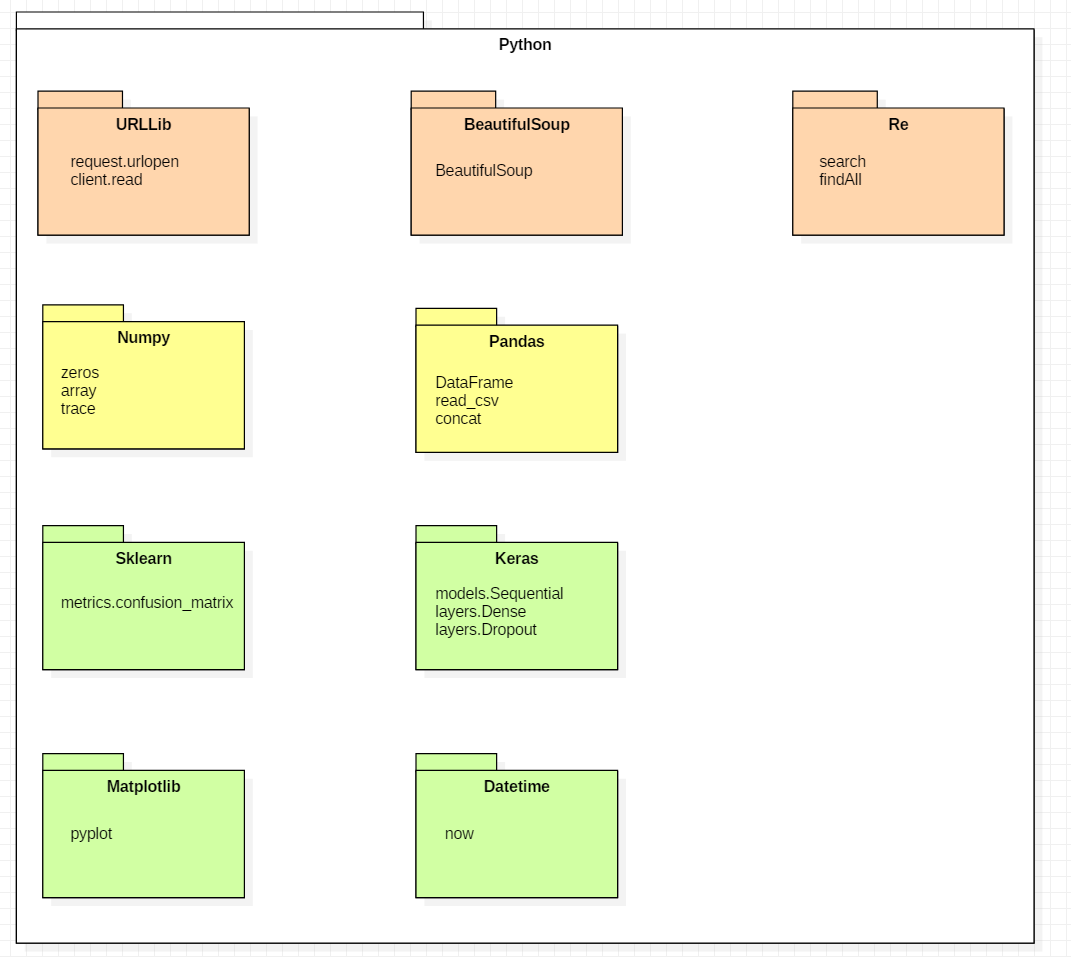


Figure 2: Package Diagram

These are the necessary packages in python that you need to run the code.

## Activity Diagram – Web Scraper

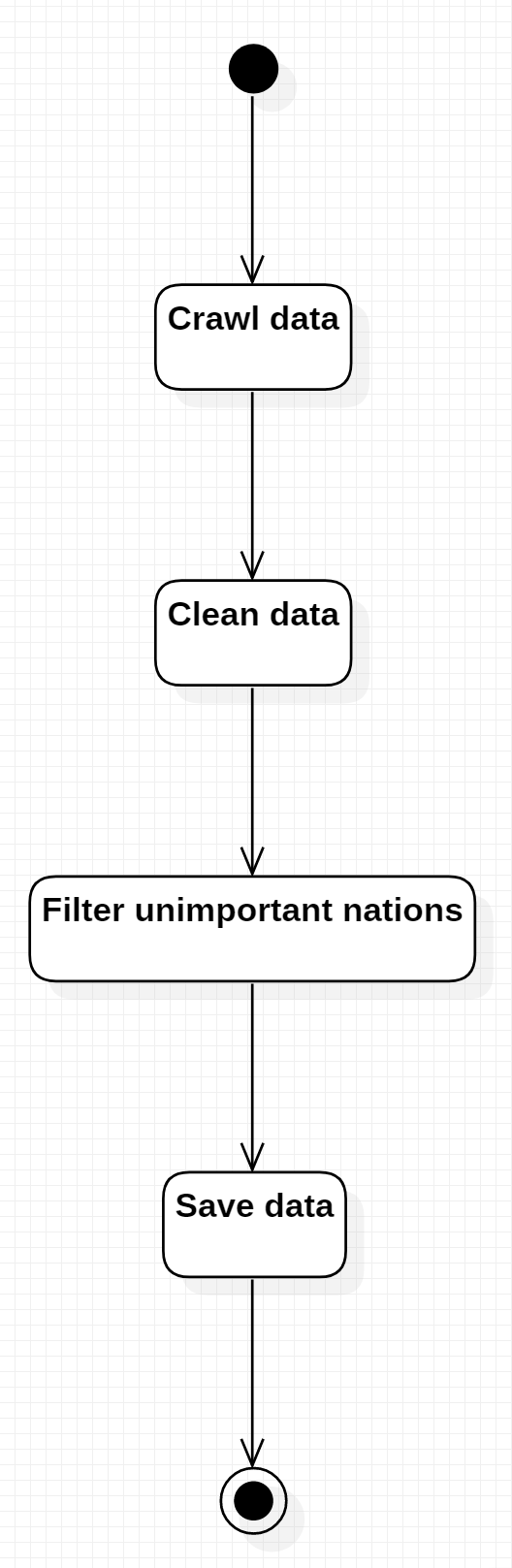


Figure 3: Activity Diagram – Web Scraper

This diagrams shows the process of getting clean data from the web.

## Activity Diagram – Predictor

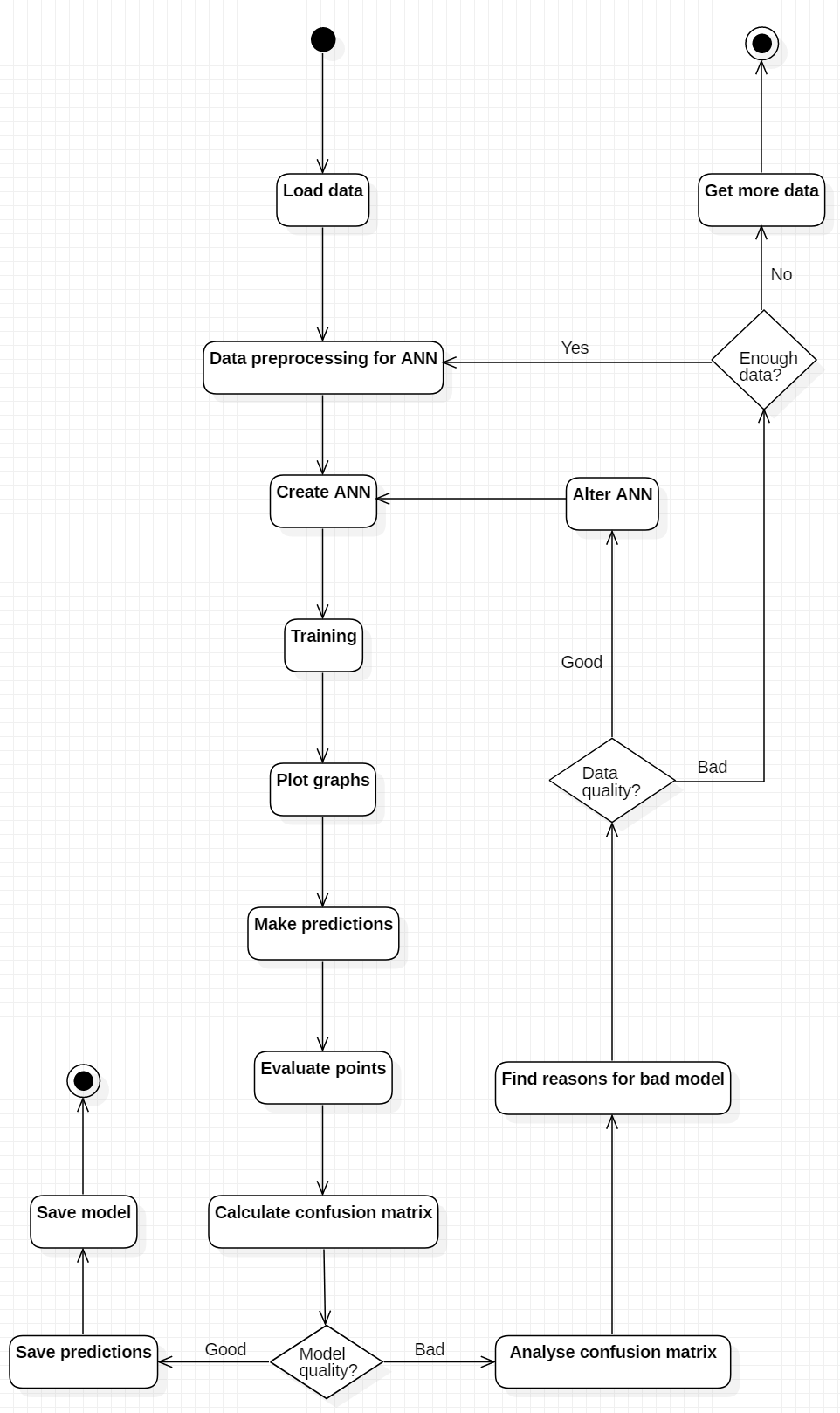


Figure 4: Activity Diagram – Predictor

This activity diagram shows the complex steps of the predictor.

## Sequence Diagram

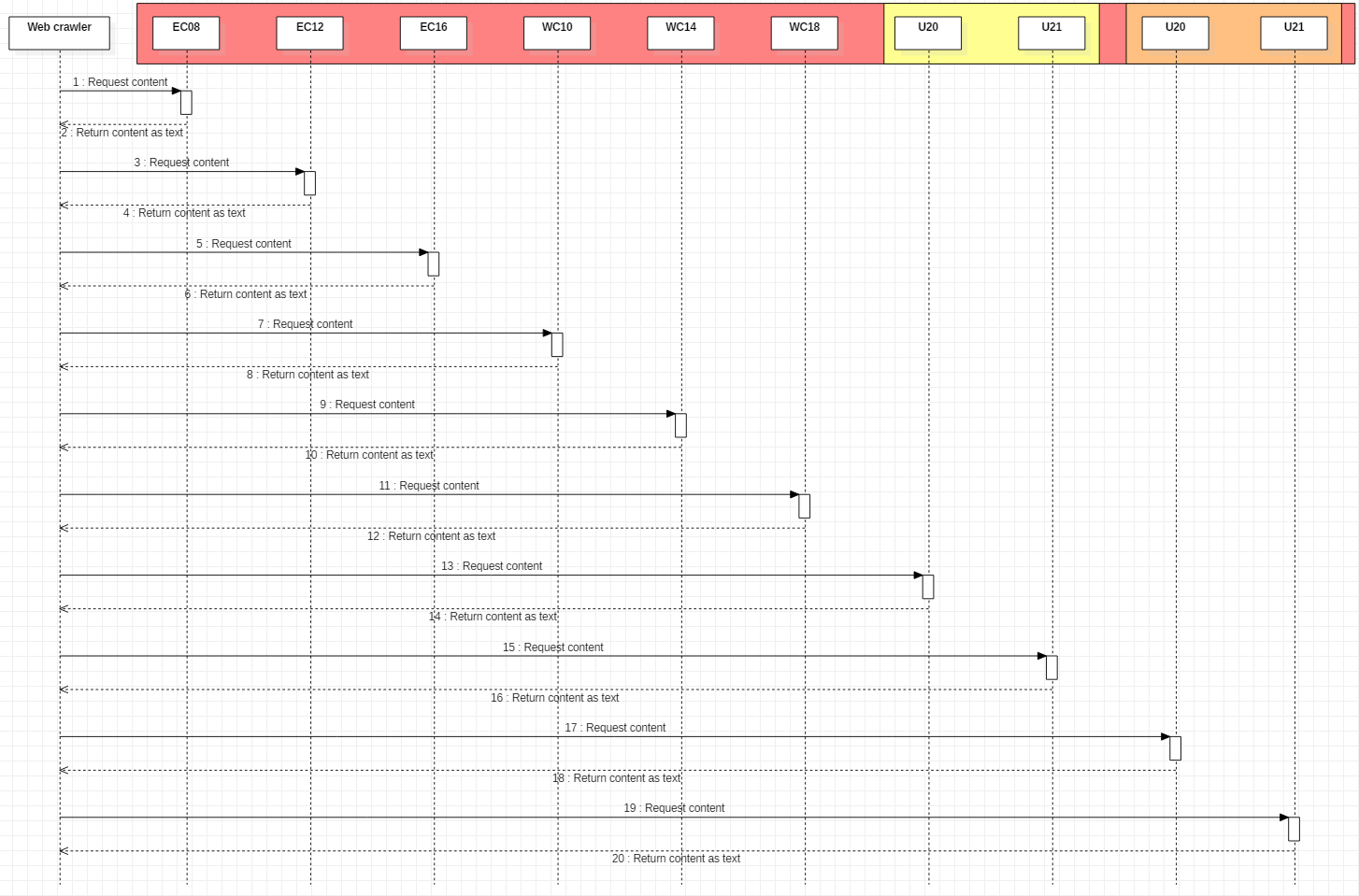


Figure 5: Sequence Diagram

This Sequence diagram shows the requests to the websites.

# Metrics

## Sonar Cloud

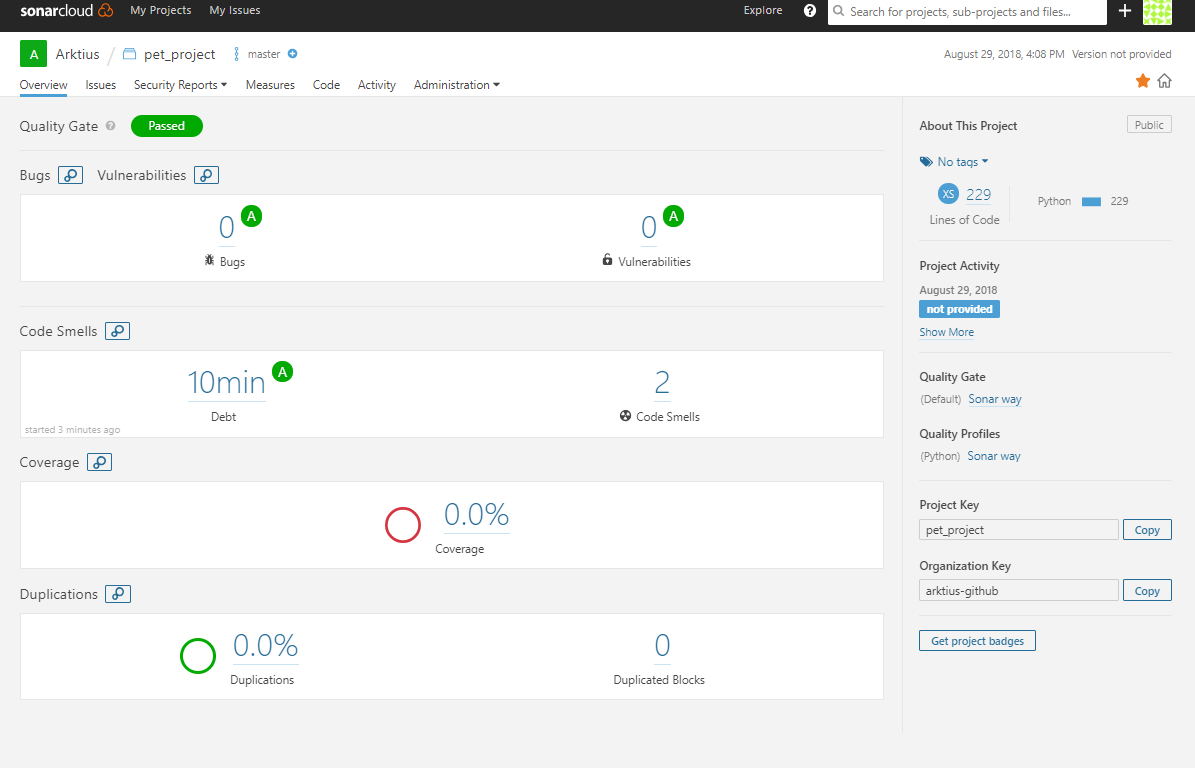


Figure 6: Sonar Cloud result

The result is pretty fine. Code smells were detected, because I commented two packages out which might be needed in the future.

## Pylint

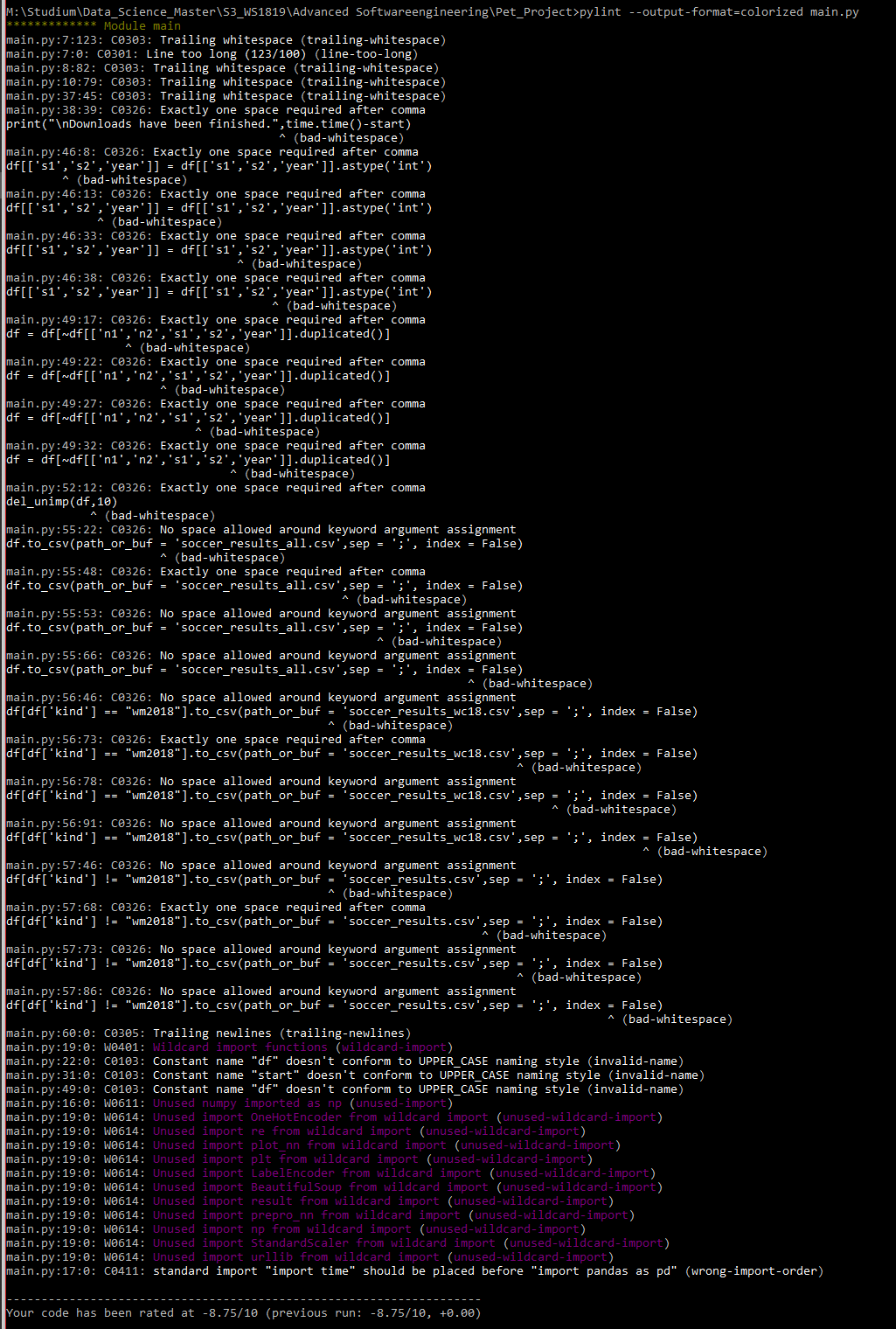


Figure 7: Pylint

Pylint was more informative than Sonar. Pylint has showed me way more things that need to be corrected such as all the whitespaces after a comma, name styles and unused imports.

# Clean Code Development

## Version Control

Since the beginning of this project, git was used as version control system.

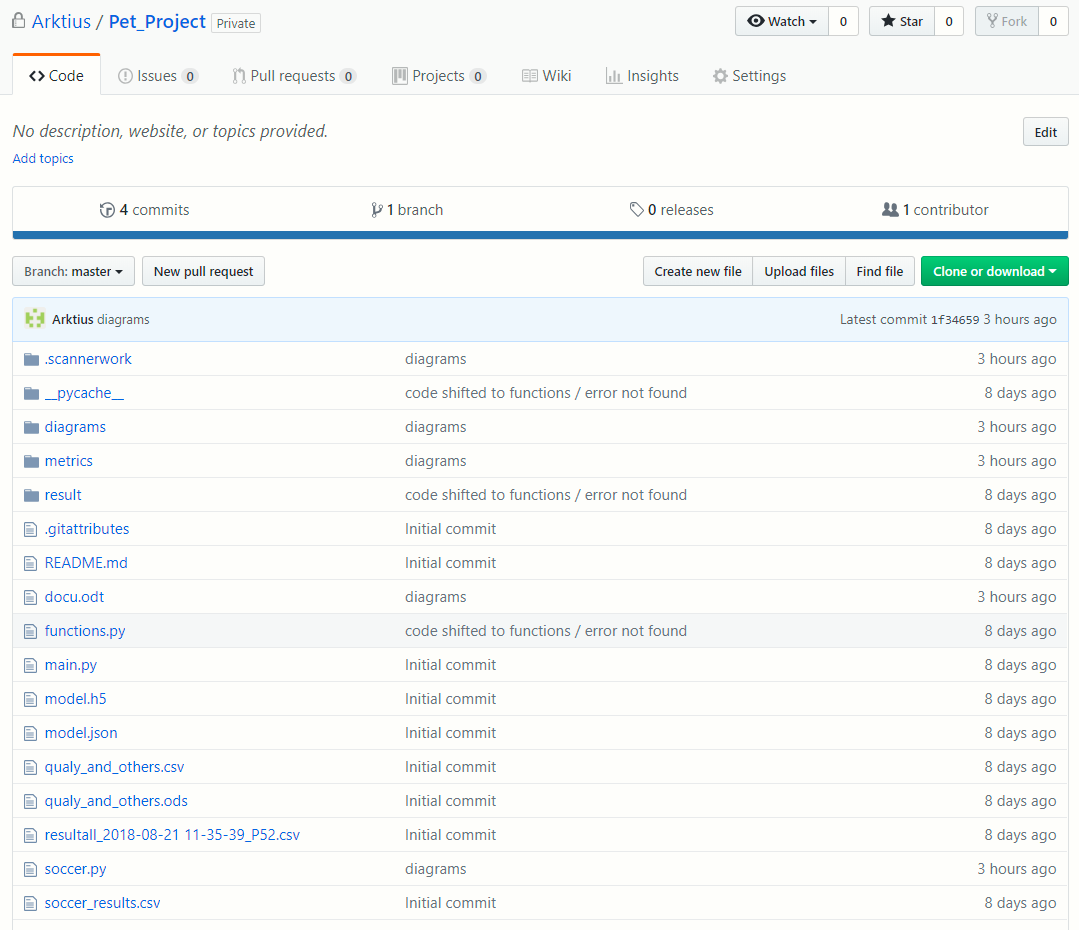


Figure 8: GitHub Repository

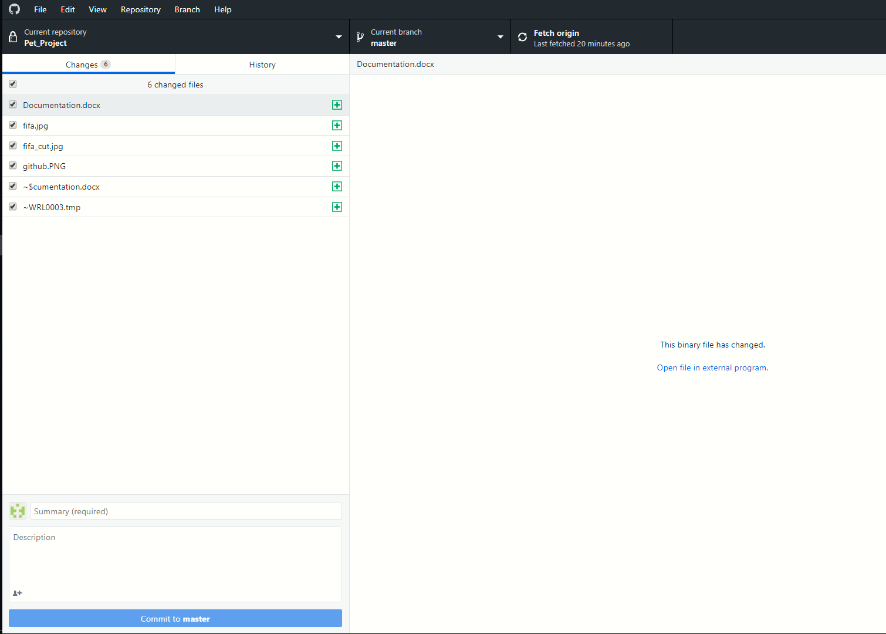


Figure 9: GitHub for Desktop

To handle the operations easily, I have been using GitHub for Desktop all the time.

## Source Code Conventions

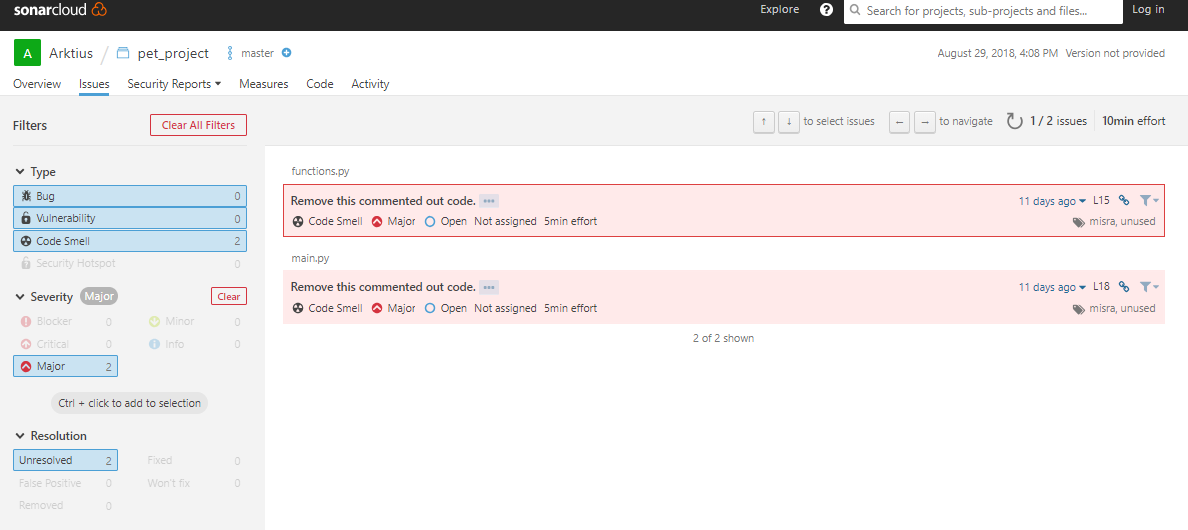


Figure : Sonar Cloud detects code smells

# Continuous Delivery

# AOP

# DSL

# Functional Programming

# Logical Solver

# Scala / Clojure