Advance Software Engineering(Pet Project)

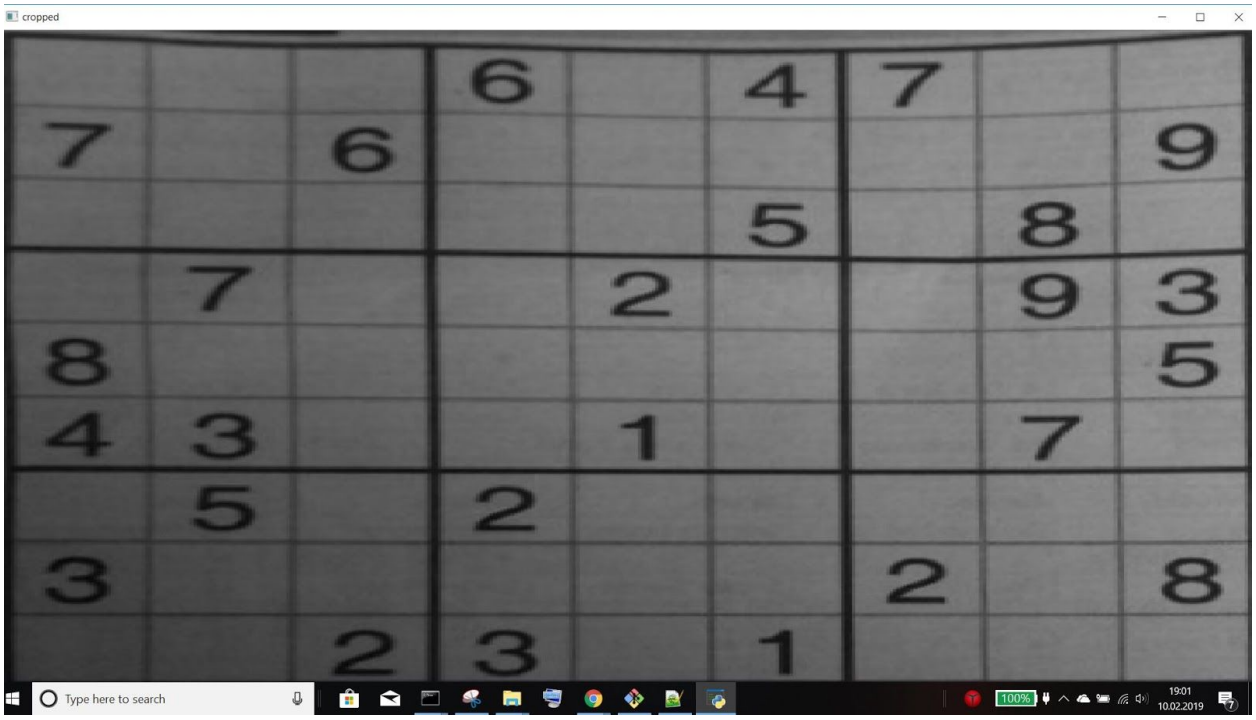
Index

Project Name: SudokuKiller

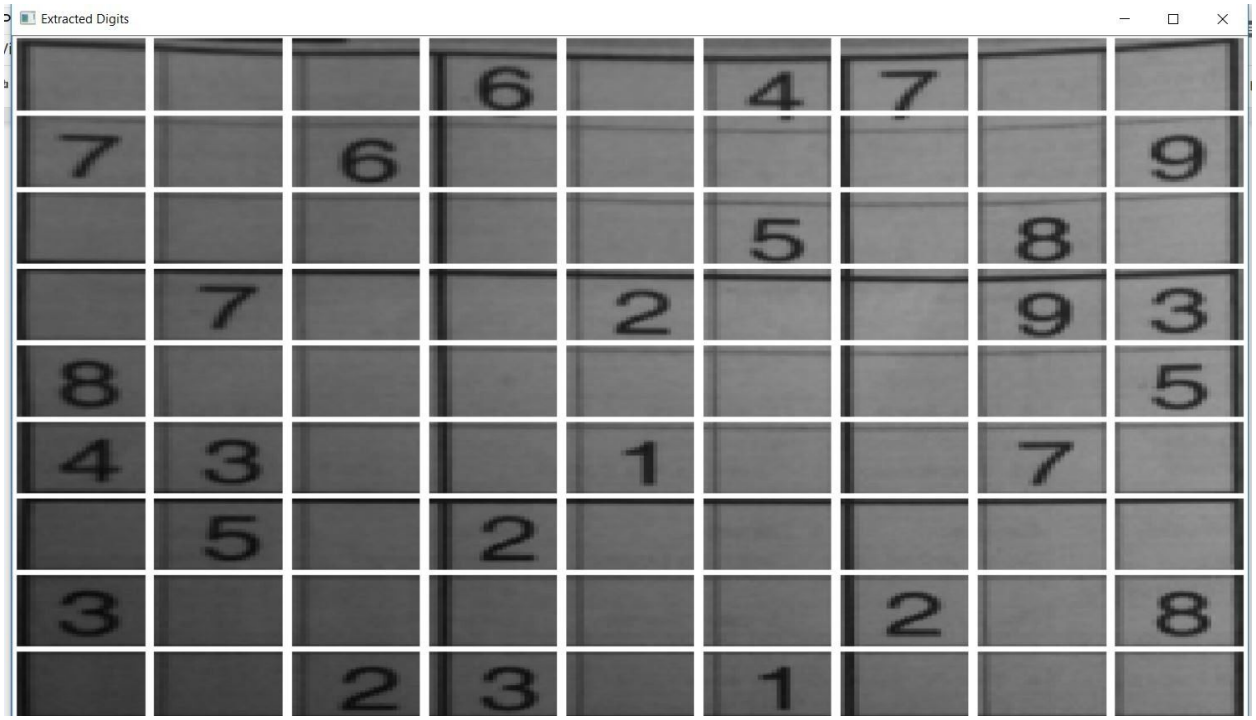
1. Introduction:

This project is machine learning (Artificial Intelligence) based , in which user will be able to get the solution of a sudoku puzzle if image is provided.

After processing the image, the app searches for the four corners of the grid and crop the grid from the image. This was done by image processing techniques rather than machine learning so it may not be entirely accurate every time.

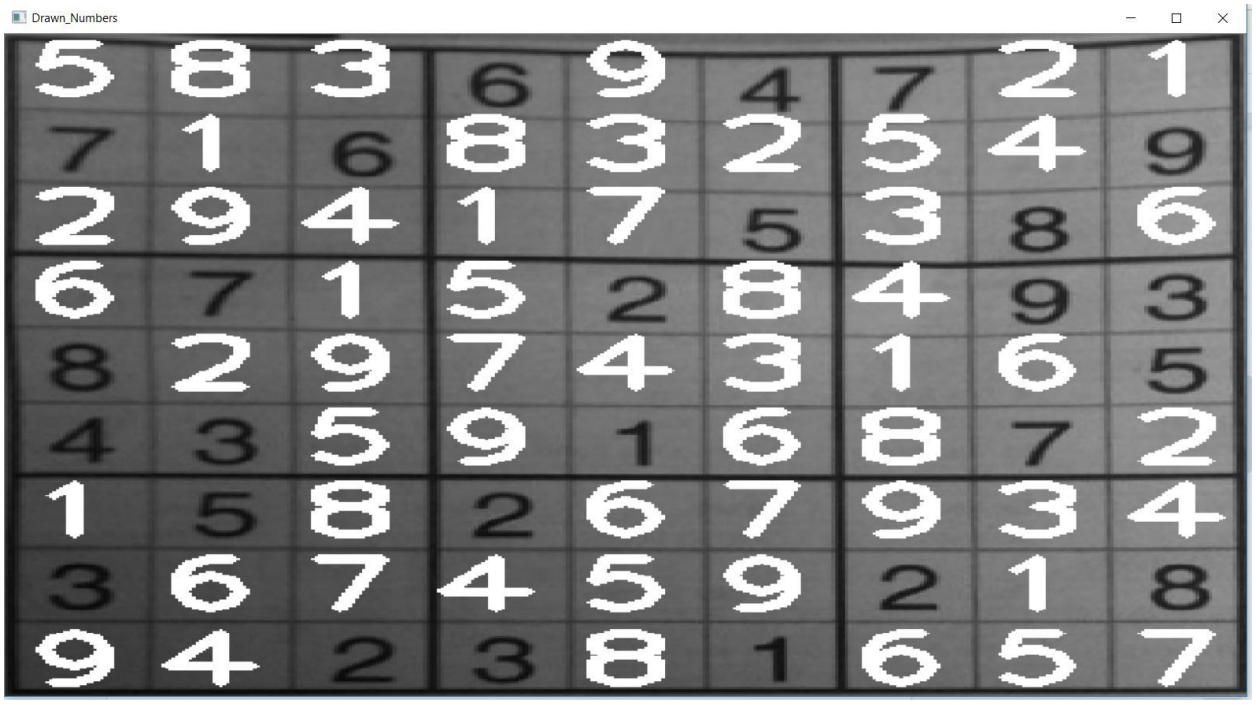


Next it estimates the grid and square locations based solely on the shape of Sudoku grids. After getting the board it will divide the whole board according the grid.



Next it will send the each box in a grid to the trained neural network model to identify digits. The trained model was collected from the internet.

Now we have the initial sudoku board and it's just a straightforward puzzle now.The final solution of a sudoku board looks like this.



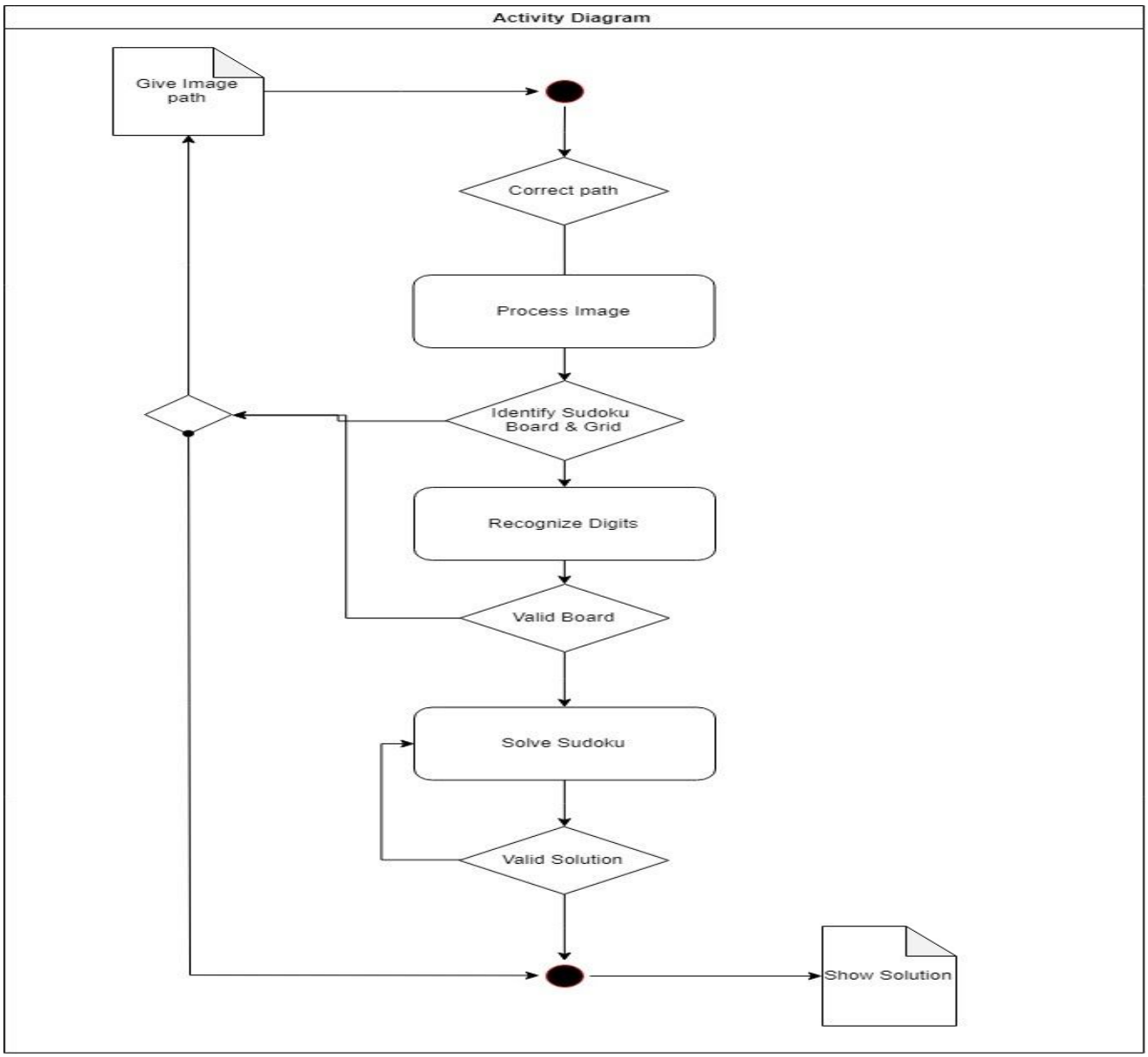
2.Technical Requirement :

- Python 3
- Numpy
- Matplotlib
- Opencv
- Tensorflow

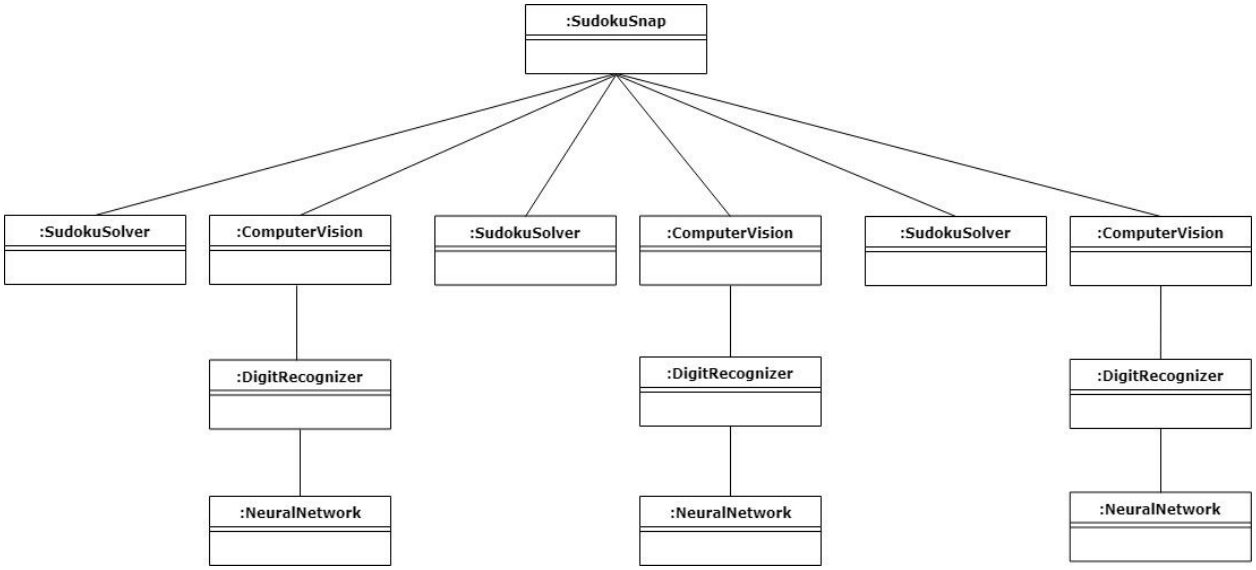
3. UML (Unified Modeling Language):

Visual representation of the system:

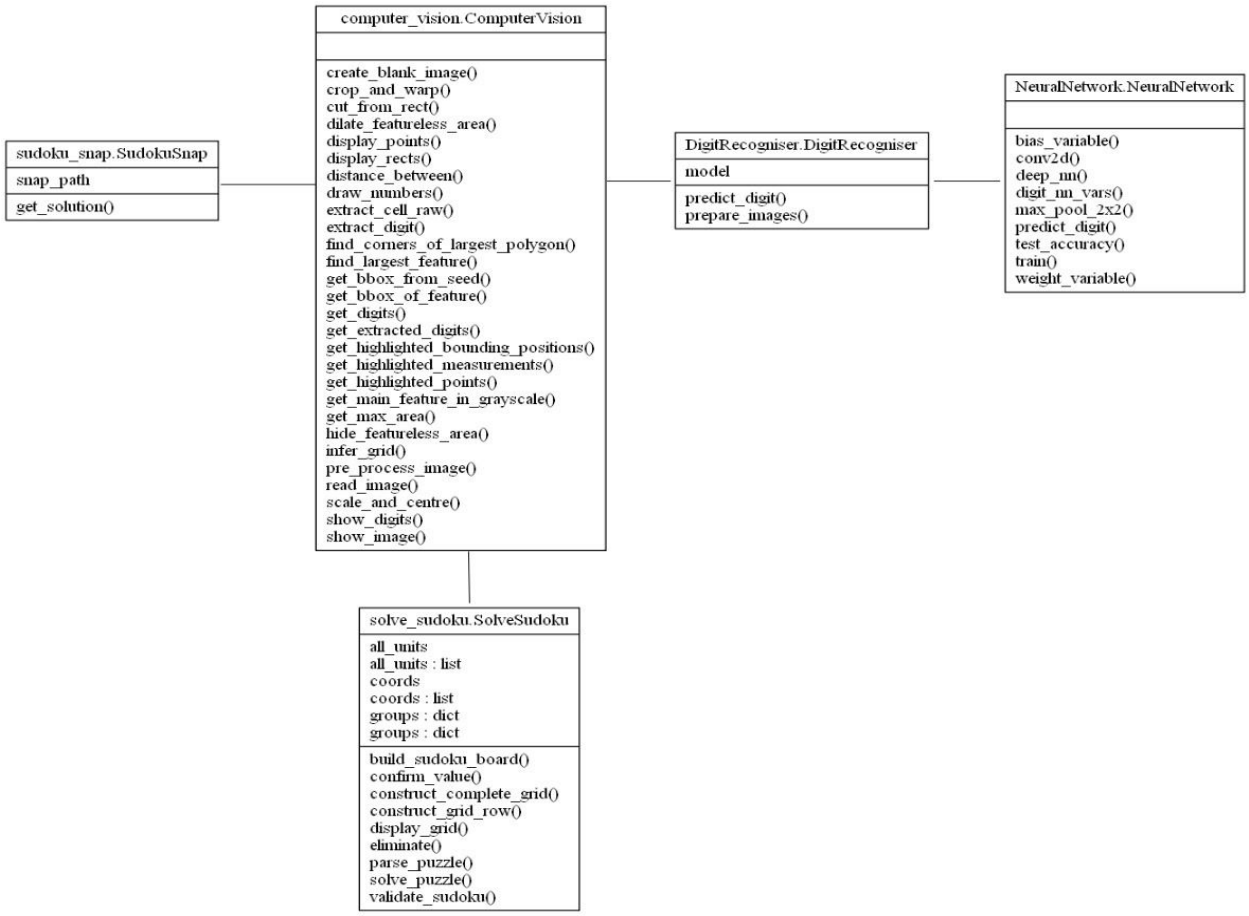
- Activity Diagram



- Object Diagram



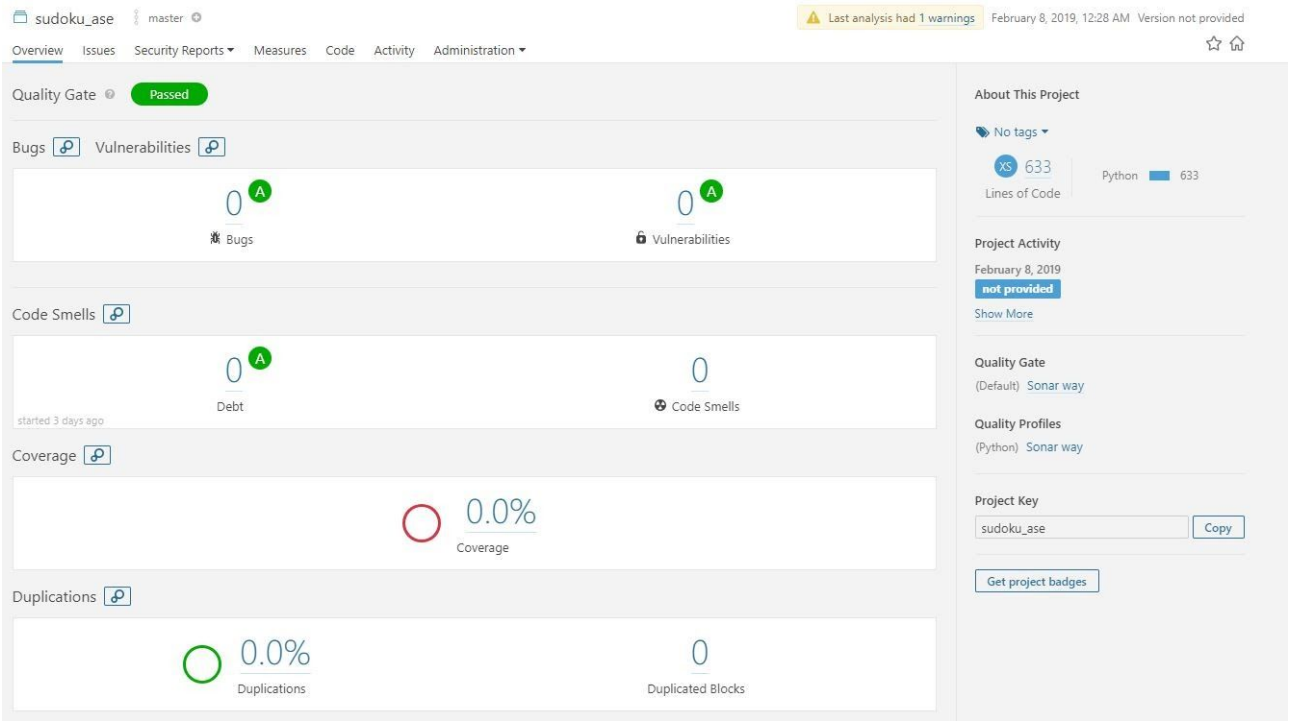
- Class Diagram:

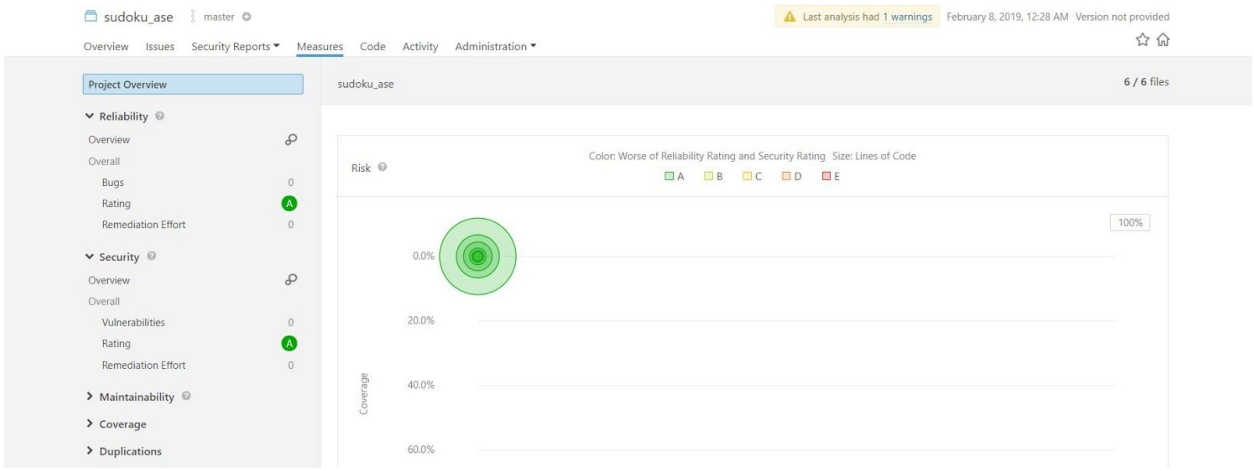


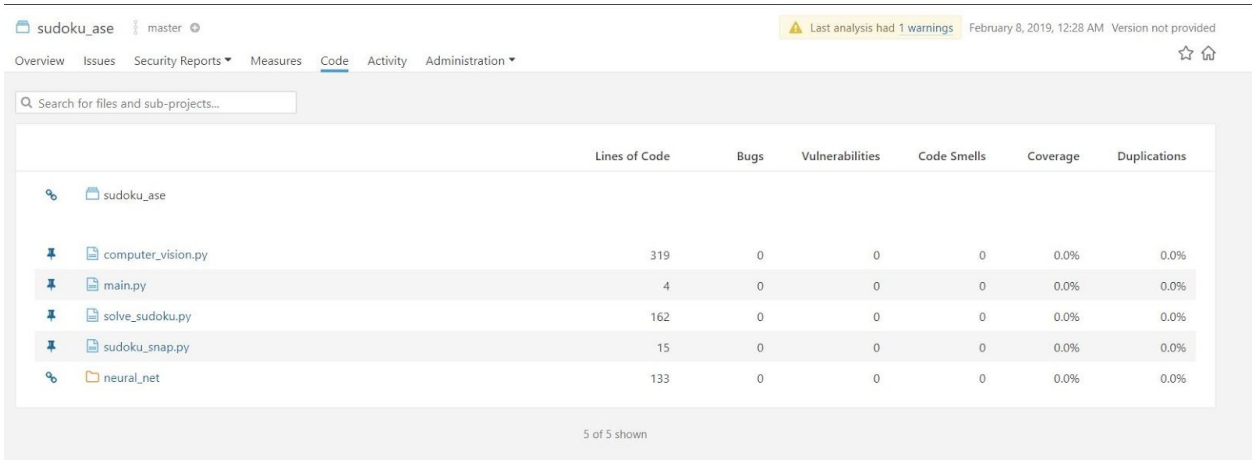
4. Metrics:

Metrics are an important component of quality assurance, management, debugging, performance, and estimating costs, and they're valuable for both developers and development team leaders. The goal of tracking and analyzing software metrics is to determine the quality of the current product or process, improve that quality and predict the quality once the software development project is complete.

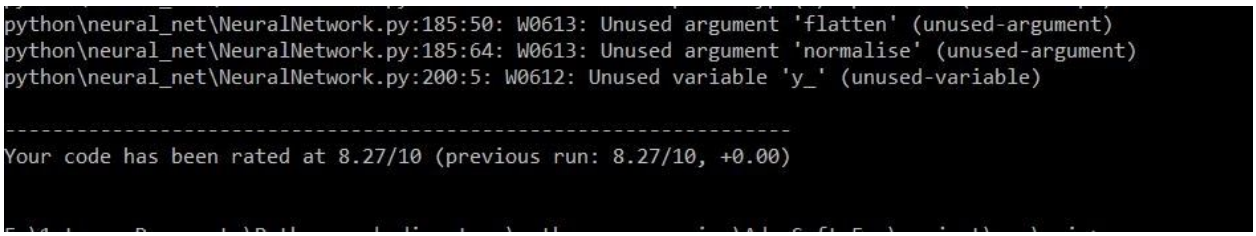
SonarQube provides the capability to not only show health of an application but also to highlight issues newly introduced. With a Quality Gate in place, you can fix the leak and therefore improve code quality systematically.







As like SonarCube i used Pylint - It checks the app performance and health at every stage , issue or error tracker.

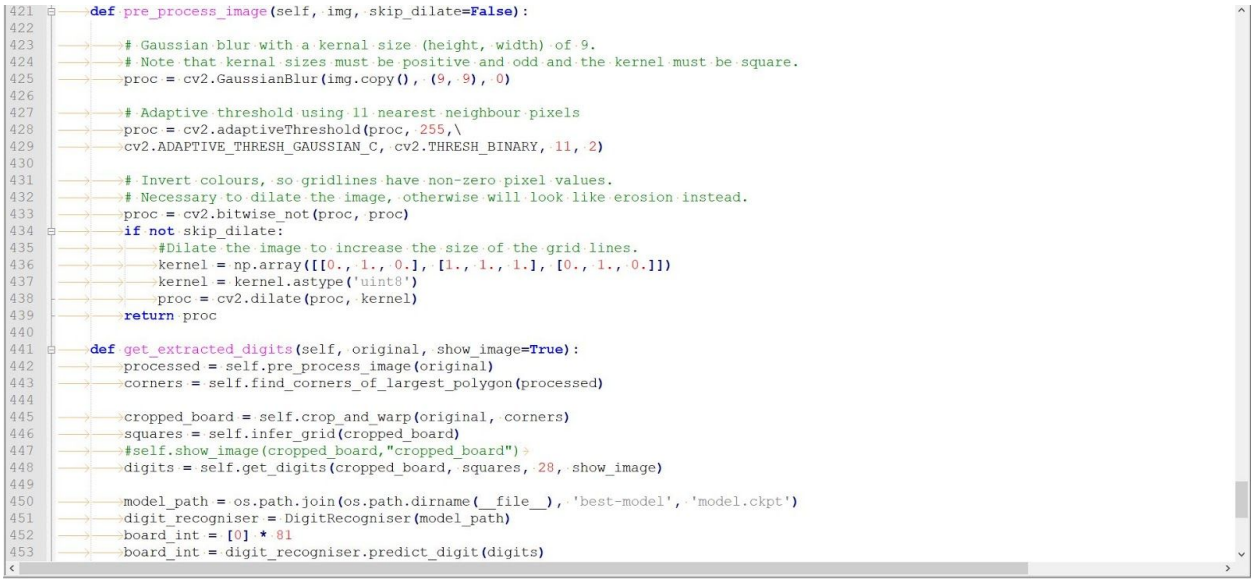


5. Clean Code Development :

There are two things- Programming and Good Programming. Programming is what we have all been doing. Now is the time to do good programming. We all know that even the bad code works. But it takes time and resources to make a program good. Moreover, other developers mock you when they are trying to find what all is happening in your code. But it's never too late to care for your programs.

(a) Coding Fundamental :

- A proper naming conventions has to use insetad of bad code like an example :
  strUsername1 = "Code"; thats a bad syntax
  Instead Use
  string_Username = "Code";



```
    def pre_process_image(self, img, skip_dilate=False):

        # Gaussian blur with a kernal size (height, width) of 9.
        # Note that kernal sizes must be positive and odd and the kernel must be square.
        proc = cv2.GaussianBlur(img.copy(), (9, 9), 0)

        # Adaptive threshold using 11 nearest neighbour pixels
        proc = cv2.adaptiveThreshold(proc, 255,\
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)

        # Invert colours, so gridlines have non-zero pixel values.
        # Necessary to dilate the image, otherwise will look like erosion instead.
        proc = cv2.bitwise_not(proc, proc)
        if not skip_dilate:
            #Dilate the image to increase the size of the grid lines.
            kernel = np.array([[0., 1., 0.], [1., 1., 1.], [0., 1., 0.]])
            kernel = kernel.astype('uint8')
            proc = cv2.dilate(proc, kernel)
        return proc

    def get_extracted_digits(self, original, show_image=True):
        processed = self.pre_process_image(original)
        corners = self.find_corners_of_largest_polygon(processed)

        cropped_board = self.crop_and_warp(original, corners)
        squares = self.infer_grid(cropped_board)
        #self.show_image(cropped_board,"cropped_board")
        digits = self.get_digits(cropped_board, squares, 28, show_image)

        model_path = os.path.join(os.path.dirname(__file__), 'best-model', 'model.ckpt')
        digit_recogniser = DigitRecogniser(model_path)
        board_int = [0] * 81
        board_int = digit_recogniser.predict_digit(digits)
```

- Class Names — Classes and objects should have noun or noun phrase names like Customer, WikiPage, Account, and AddressParser. Avoid words like Manager, Processor, Data, or Info in the name of a class. A class name should not be a verb.

```
import operator
import os
import numpy as np
from matplotlib import pyplot as plt
import cv2

from neural_net.DigitRecogniser import DigitRecogniser


class ComputerVision:

    def __repr__(self):
        return "Module of image processing tasks."

    def read_image(self, path):
        return cv2.imread(path, cv2.IMREAD_GRAYSCALE)

    def display_rects(self, in_img, rects, colour=255):
        img = in_img.copy()
        for rect in rects:
            img = cv2.rectangle(img, tuple(int(x) for x in rect[0]),\
            tuple(int(x) for x in rect[1]), colour)
        self.show_image(img, "With squares")
        return img

    def infer_grid(self, img):
        '''Infers 81 cell grid from a square image.'''
        squares = []
        side = img.shape[:1]
        side = side[0] / 9
        for i in range(9):
            for j in range(9):
                top_corner = (j * side, i * side)#Top left corner of a bounding box
```

- Method Names —Methods should have verb or verb phrase names like postPayment, deletePage, or save. Accessors, mutators, and predicates should be named for their value and prefixed with get, set. (above picture)

- Code should explain everything. Modern programming languages are english like through which we can easily explain our point. Correct naming can prevent comments. However there are also documentation comments and clarification comments.

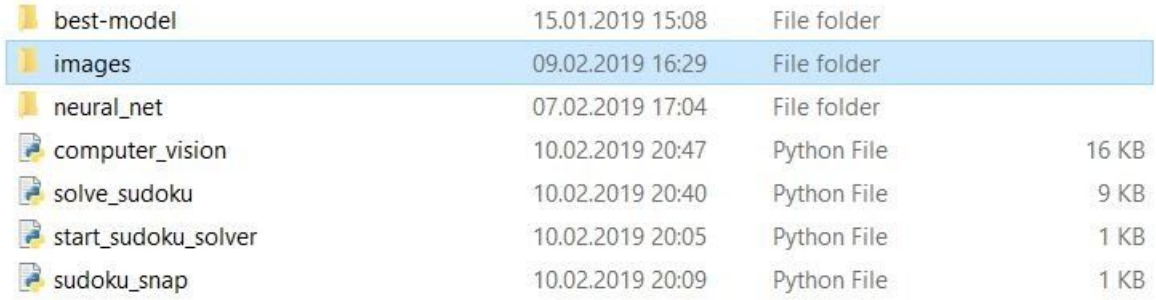
```
        return top, left, right, bottom

    def find_largest_feature(self, inp_img, scan_tl=None, scan_br=None):

        '''
        Uses the fact the `floodFill` function returns a
        bounding box of the area it filled to find the biggest
        connected pixel structure in the image.
        Fills this structure in white, reducing the rest to black.
        '''

        img = inp_img.copy() # Copy the image, leaving the original untouched
        height, width = img.shape[:2]

        max_area = 0
        seed_point = (None, None)

        if scan_tl is None:
            scan_tl = [0, 0]

        if scan_br is None:
            scan_br = [width, height]

        seed_point = self.get_max_area(img, scan_br, scan_tl, max_area)

        mask = self.get_main_feature_in_grayscale(img, height, width, seed_point)

        top, bottom, left, right = height, 0, width, 0

        for x in range(width):
            for y in range(height):
                self.hide_featureless_area(img, x, y, mask)
                top, left, right, bottom = self.get_bbox_of_feature(img, x, y, [top, bottom, left, right] )
```

Clarification comments are intended for anyone (including your future self) who may need to maintain, refactor, or extend your code. Documentation comments are intended for anyone who is likely to consume your source code, but not likely to read through it.

- Verbs to be used for function names and a function shouldn't do more than one task.

```
    def get_max_area(self, img, scan_br, scan_tl, ):
        for x in range(scan_tl[0], scan_br[0]):
            for y in range(scan_tl[1], scan_br[1]):
                if img.item(y, x) == 255 and x < width and y < height:
                    area = cv2.floodFill(img, None, (x, y), 64)
                    if area[0] > max_area:
                        max_area = area[0]
                        return (x, y)

    def get_main_feature_in_grayscale(self, img, height, width, seed_point):
        
        self.dilate_featureless_area(img, height, width)
        mask = np.zeros((height + 2, width + 2), np.uint8)

        if all([p is not None for p in seed_point]):
            cv2.floodFill(img, mask, seed_point, 255)
        return mask
```

(b) Folder Structure : Always create folder according to the file name and if we have more than one file create separate folder for like neural_net here.



6. Build Management:

PyBuilder is a software build tool written in 100% pure Python, mainly targeting Python applications. PyBuilder is based on the concept of dependency based programming, but it also comes with a powerful plugin mechanism, allowing the construction of build life cycles similar to those known from other famous (Java) build tools.

To build a project using this first one have to create a build file.

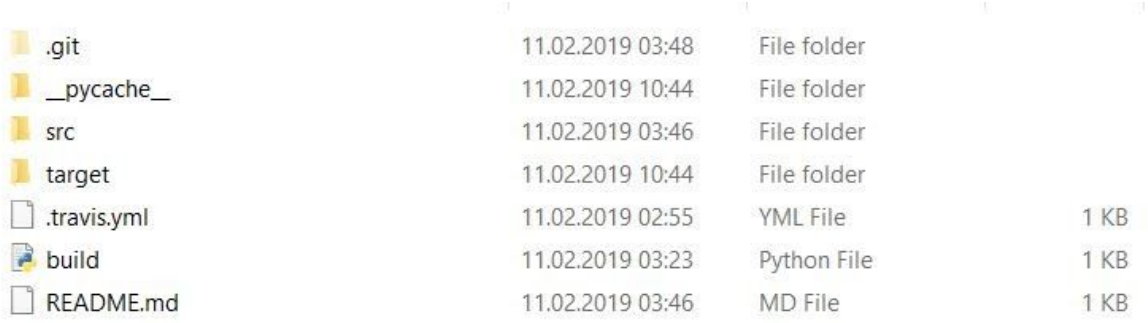
```
from pybuilder.core import init, use_plugin

use_plugin("python.core")
use_plugin("python.unittest")
use_plugin("python.install_dependencies")
use_plugin("python.distutils")

default_task = "publish"

@init
def initialize(project):
    project.build_depends_on('os')
    project.build_depends_on('numpy')
    project.depends_on('matplotlib')
    project.depends_on('opencv-python', "==3.4.4.19")
    project.depends_on('tensorflow', "==1.12.0")
    # File is installed relative to sys.prefix
    project.install_file("Lib/site-packages/neural_net","neural_net/DigitRecogniser.py")
    project.install_file("Lib/site-packages/neural_net","neural_net/NeuralNetwork.py")
    project.install_file("Lib/site-packages/best-model","best-model/model.ckpt.data-00000-of-00001")
    project.install_file("Lib/site-packages/best-model","best-model/model.ckpt.index")
```

Now it's time to tell *PyBuilder* to

- execute the tests on each build
- break the build (i.e. produce a failure) when a test fails.

```
$ pyb
PyBuilder version 0.11.17
Build started at 2019-02-11 10:44:03
------------------------------------------------------------
[INFO]  Building SudokuKiller version 1.0.dev0
[INFO]  Executing build in E:\1st_sem_Documents\Python work directory\python programming\Adv Soft Eng\project\SudokuKiller
[INFO]  Going to execute task publish
[INFO]  Running unit tests
[WARN]  Not forking for <function do_run_tests at 0x000001F6D0F307B8> due to Windows incompatibilities (see #184). Measurements (coverage, etc.) might be biased.
[INFO]  Executing unit tests from Python modules in e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\src\unittest\python
2019-02-11 10:44:05.817104: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
[INFO]  Executed 2 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\target\dist\sudokukiller-1.0.dev0
[INFO]  Copying scripts to e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\target\dist\sudokukiller-1.0.dev0\scripts
[INFO]  Writing MANIFEST.in as e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\target\dist\sudokukiller-1.0.dev0\manifest.in
[INFO]  Writing setup.py as e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\target\dist\sudokukiller-1.0.dev0\setup.py
[INFO]  Building binary distribution in e:\1st_sem_documents\python work directory\python programming\adv soft eng\project\sudokukiller\target\dist\sudokukiller-1.0.dev0
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
            Project: SudokuKiller
            Version: 1.0.dev0
     Base directory: E:\1st_sem_Documents\Python work directory\python programming\Adv Soft Eng\project\SudokuKiller
       Environments:
              Tasks: prepare [267 ms] compile_sources [0 ms] run_unit_tests [2386 ms] package [234 ms] run_integration_tests [0 ms] verify [0 ms] publish [6689 ms]
Build finished at 2019-02-11 10:44:12
Build took 9 seconds (9608 ms)
```

After a successful build one can easily find the republished code inside the target folder.



.git	11.02.2019 03:48	File folder	
__pycache__	11.02.2019 10:44	File folder	
src	11.02.2019 03:46	File folder	
target	11.02.2019 10:44	File folder	
.travis.yml	11.02.2019 02:55	YML File	1 KB
build	11.02.2019 03:23	Python File	1 KB
README.md	11.02.2019 03:46	MD File	1 KB

7. Continuous Delivery:

Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted at GitHub. It will build your app and run your tests each time you make a push to the repository.

```
$ git clone --depth=50 --branch=master https://github.com/rashikcs/SudokuKiller.git rashikcs/SudokuKiller

$ source ~/virtualenv/python3.6/bin/activate
$ python --version
Python 3.6.3
$ pip --version
pip 9.0.1 from /home/travis/virtualenv/python3.6.3/lib/python3.6/site-packages (python 3.6)
$ pip install -U pip setuptools
$ pip install pybuilder
$ pip install numpy
$ pip install -U pip matplotlib
$ pip install opencv-python==3.4.4.19
$ pip install tensorflow==1.12.0
$ pyb
PyBuilder version 0.11.17
Build started at 2019-02-11 02:52:35
------------------------------------------------------------
[INFO]  Building SudokuKiller version 1.0.dev0
[INFO]  Executing build in /home/travis/build/rashikcs/SudokuKiller
[INFO]  Going to execute task publish
[INFO]  Installing plugin dependency pypandoc
[INFO]  Installing plugin dependency unittest-xml-reporting
[INFO]  Running unit tests
[INFO]  Executing unit tests from Python modules in /home/travis/build/rashikcs/SudokuKiller/src/unittest/python
2019-02-11 02:52:40.203263: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
[INFO]  Executed 2 unit tests
[INFO]  All unit tests passed.
[INFO]  Building distribution in /home/travis/build/rashikcs/SudokuKiller/target/dist/SudokuKiller-1.0.dev0
[INFO]  Copying scripts to /home/travis/build/rashikcs/SudokuKiller/target/dist/SudokuKiller-1.0.dev0/scripts
[INFO]  Writing MANIFEST.in as /home/travis/build/rashikcs/SudokuKiller/target/dist/SudokuKiller-1.0.dev0/MANIFEST.in
[INFO]  Writing setup.py as /home/travis/build/rashikcs/SudokuKiller/target/dist/SudokuKiller-1.0.dev0/setup.py
[INFO]  Building binary distribution in /home/travis/build/rashikcs/SudokuKiller/target/dist/SudokuKiller-1.0.dev0
------------------------------------------------------------
BUILD SUCCESSFUL
------------------------------------------------------------
Build Summary
             Project: SudokuKiller
             Version: 1.0.dev0
      Base directory: /home/travis/build/rashikcs/SudokuKiller
        Environments:
               Tasks: prepare [2274 ms] compile_sources [0 ms] run_unit_tests [2539 ms] package [39 ms] run_integration_tests [0 ms] verify [0 ms] publish [4550 ms]
Build finished at 2019-02-11 02:52:44
Build took 9 seconds (9435 ms)
The command "pyb" exited with 0.



Done. Your build exited with 0.
```

For that all you need to do is write a *'.travis.yml'* file on the root path of your project. This file

```yaml
language: python
python:
  - "3.6"
# command to install dependencies
install:
  - pip install -U pip setuptools
  - pip install pybuilder
  - pip install numpy
  - pip install -U pip matplotlib
  - pip install opencv-python==3.4.4.19
  - pip install tensorflow==1.12.0
# command to run tests
script: pyb
```

will contain all the configurations Travis needs to build and run the tests written in your project.

Video Demonstration of the project