Still Pending
Bachelor Report

May 16th, 2019

IS-304, University of Agder

Ali Al-Musawi, Kent Daleng, Eirik Fintland, Osman Younas

Abstract

The Norwegian Coastal Administration is looking to create a new version of an already existing dataset. The existing dataset defines the coastline using segmented lines with a terrain type associated with each segment, but the dataset contains a lot of classification errors. To that end, we were asked by Norkart and the Norwegian Coastal Administration to look into the possibilities of using machine learning for terrain classification to help create a new, more correct dataset. In this report we will explore a possible approach to this task and describe how we used work methods like Kanban and Scrum as well as Anaconda, QGIS and TensorFlow as tools for our process. As it stands at the time of writing, with the achieved result, we find it likely that machine learning is a viable method for classifying terrain from aerial photos.

Table of contents

1.0 Introduction **4**

1.1 Project description 4

1.2 Why is this an interesting and valuable project 5

1.3 What is the use of the product 6

1.4 Complexity 6

1.5 Terminologi 6

2.0 Project management **7**

2.1 Methodology 7

2.2 Quality assurance 8

2.3 Time constraints 8

2.4 Technology 9

2.4.1 Trello 9

2.4.2 QGIS 10

2.4.3 Anaconda 11

3.0 Process **12**

3.1 Sprint example 12

3.2 Realism in the project 14

3.3 Challenges 14

3.3.1 Technical challenges 14

3.3.2 Organizational challenges 15

3.4 User stories 15

4.0 Product **17**

4.1 Ensuring the existence of data 17

4.1.1 Folder structure 17

4.1.2 Data files 18

4.2 Generating truth data 19

4.3 Creating a model 20

4.3.1 How an ImageDataGenerator works 20

4.4 Fitting the model 21

4.5 Evaluating accuracy 23

5.0 Conclusion **25**

6. Literature **26**

List of figures

Figure 1 4
Figure 2 10
Figure 3 11
Figure 4 13
Figure 5 18
Figure 6 19
Figure 7 20
Figure 8 21
Figure 9 22
Figure 10 22
Figure 11 23
Figure 12 24

1.0 Introduction

This report will present the findings of the group's work done in IS-304. The bachelor project is first and foremost in collaboration with Norkart. The Norwegian Coastal Administration is the owner of the dataset and will thus also be collaborating with the group. Both organisations are stakeholders. Norkart was founded in 1961 and is the market leader within solutions for municipal engineering, as well as map and property information for towards both the public and private sector. *(Norkart, n.d.)*

The Norwegian Coastal Administration is an agency of the Norwegian Ministry of Transport and Communications responsible for services related to maritime safety, maritime infrastructure, transport planning and efficiency, and emergency response to acute pollution. (*Kystverket, 2018*)

1.1 Project description

The Norwegian Coastal Administration is looking to create a new version of an already existing dataset, which should be expedient and quality assured. The existing dataset defines the coastline using segmented lines. Each segment is categorised by the coastal type, but a lot of data errors exist.

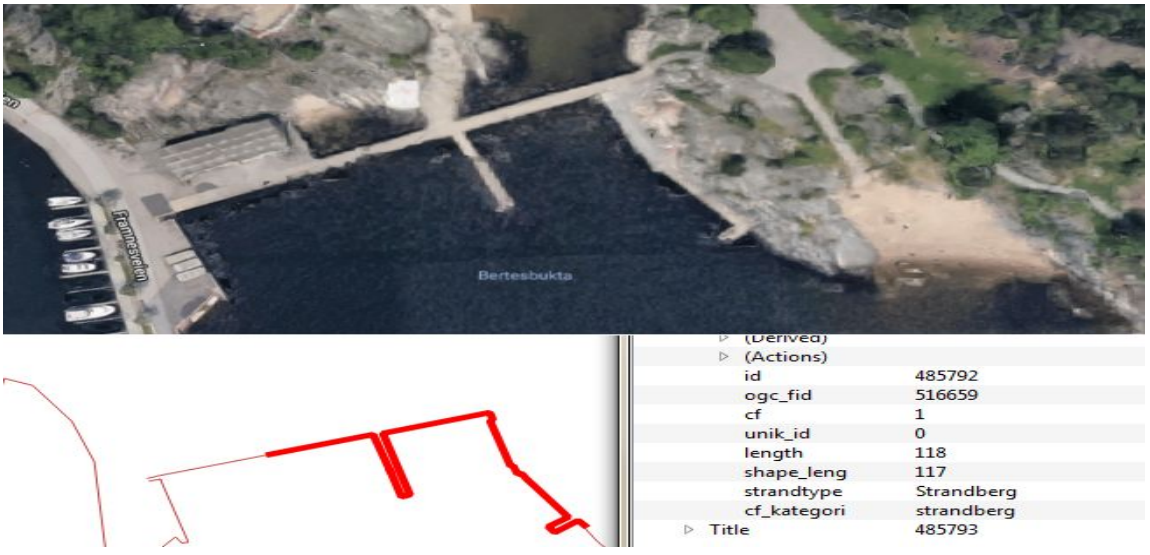

Figure 1: Example of data error. Man-made structure has been classified as bare rock.

The goal of the project is to research potential solutions to developing the dataset. Initially, this involves classifying coastal types by using an array of sources such as

satellite imaging, aerial photos, drone pictures, and by potentially using crowdsourcing where using the prior sources has not been able to produce a conclusive classification. In other words, a finished dataset is not the goal, but rather that the group examines methodology and whether image recognition is a viable approach.

The bachelor group Still Pending consists of the following members:

Member	**Role**
Ali Al Musawi	Responsible for stand ups, primarily responsible for work tasks
Eirik Fintland	Responsible for documentation
Kent Daleng	Responsible for maintaining Git repositories, and setup of test environments
Osman Younas	Responsible for documentation

Generally speaking the group has a flat structure. The report and the technical side is a shared responsibility by all.

1.2 Why is this an interesting and valuable project

The project is interesting because attempting to classify coastal types in sequential layers has not been tried within the organisation before. For instance, given a classification from satellite images with high confidence, it may be possible to skip going down to aerial photos to gain more confidence. Another example would be to use crowdsourcing in cases where image recognition has not provided results of high confidence, and thus let people manually classify areas. Additionally, approaches with machine learning and more traditional image recognition will be compared. Machine learning has not been used in a large scale with the client before, and as such the research conducted will be valuable. The client will also be following the project closely, and will therefore be able to gain experience with machine learning which can potentially be carried over to other projects.

1.3 What is the use of the product

This dataset by the Norwegian Coastal Administration will be used as a lookup for the entire coastline of Norway. An example of use is in the case of oil spills where the oil will behave differently on different surfaces, such as sand beaches or rôche moutonnée. The dataset will then provide an essential foundation for how a clean up crew will prepare for the mission.

1.4 Complexity

This project will require an in-depth understanding of image recognition, both through using traditional algorithms and through machine learning. An ability to modify existing solutions as well as to develop own solutions according to the client's needs is essential to the project. At the same time, the project will require an understanding of data structures and algorithms to best organise the data to a set that is generated.

1.5 Terminologi

Throughout this report, some terminology will be used that might otherwise be confusing. These terms will be clarified in the table below:

Terminology	**Description**
Enkai	The name of the application
Dataset	Pictures, video and other data of geographical areas that are used for the purposes of training Enkai
Epoch *(Weenink, D., 2004)*	A term that is often used in the context of machine learning. An epoch is one complete presentation of the *data set to be learned* to a learning machine.
The group / we	The group members responsible for the project development
NCA	Norwegian Coastal Administration, or Kystverket in Norwegian
Agile software development *(Collier, K. W., 2011, pg. 121)*	Agile software development is an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s).

Open source (*Open Source Initiative, n.d.)*	The term "open source" refers to something people can modify and share because its design is publicly accessible. It is software with source code that anyone can inspect, modify, and enhance.
Scrum *(Boehm, B. & Turner, R. 2012, 8th ed.)*	Scrum is an agile framework for development designed for teams to break down a project into one to four week iterative cycles called Sprints. Using Scrum, one must begin with distributing roles: Scrum master (SM), product owner (PO) and team members (TM).
Kanban *(Radigan, D., n.d.)*	Kanban is a framework used to implement agile software development. It requires real-time communication of capacity and full transparency of work. Work items are represented visually on a kanban board, allowing team members to see the state of every piece of work at any time.
Slack & Discord	Slack is the communication tool we used in collaboration with the representatives from NCA and Norkart. Discord is also a tool for communication that we used internally for group members only.

2.0 Project management

In this chapter we will present our work methodology and approach for quality assurance, in addition to the time constraints and technology involved in the project.

2.1 Methodology

Generally speaking, an agile approach is what's most appropriate for this project. The group will assume to maintain a mixture of Scrum and Kanban. It is desirable from all parties that the client takes a part of the project. In addition to that the client wishes to contribute to the technical side of the project, the group also wishes that the client gets regular updates on the project in general. This will be achieved through having regular presentations, for instance at the end of a Sprint period.

Initially, the group wishes to set the length of Sprints to three or four weeks, with a small presentation with the client at the end of each period. Then a general assessment of the work will be made, which is in accordance with Scrum.

According to Kanban however, new work tasks can be created on demand, but this in this case it will be natural to create the bulk of new work tasks after an assessment as previously mentioned. Work tasks will be described like user stories. If at any point a group member thinks a task is too general, it can be discussed and split into smaller parts, as long as it is documented.

In summary, the group does not wish to strictly use Scrum, but rather perform work more based on Kanban and continuous delivery. Discussion and changes to work tasks can occur at any time. The Sprint periods will as such be more to summarise all of what has happened in the period, both for the group and the client.

2.2 Quality assurance

The most central part towards ensuring quality is to document well and to be in close contact with the client. It is primarily the actual surveying of methodology and whether image recognition is a viable approach which will deliver the most value to the client. An example of something specific to ensure quality is that the group has a goal of discussing documents and work tasks with equal priority in standups.

As a way to ensure quality we decided to use test-driven development in order to identify and tackle problems as early as possible during the development cycle. If you integrate regularly — there is much less to check while looking for errors. That results in less time spent for debugging and more time for adding features.

At one point in the project, Enkai had a low and unsatisfactory terrain classification accuracy. In order to figure out if the problem lied with the dataset or with the codebase, we used binary classification of cats and dogs in an attempt to reduce the amount of factors involved. The result of the binary classification ended up with a significantly higher average accuracy, which indicated that the problem lied with the dataset and not the codebase.

2.3 Time constraints

The bachelor group consists of four members. The project also involves two members from Norkart as well as 3 members from the Norwegian Coastal Administration. Even though there has been expressed interest from all parties to participate in the project in various ways, it’s impossible to quantify how many hours the client will be able to

put into project work. This project is due on the 16th of may 2019, therefore the group assumes to work a minimum of 26 hours per week, following the standard workload for 20 credits according to UIA. *(University of Agder, n.d.)*

2.4 Technology

There are several technologies involved in the project. The project was developed using Python. We used Anaconda as a package environment for Python libraries. The following packages were used: GDAL, GeoPandas, LibGDAL, Keras, RasterIO and Tensorflow. QGIS was used to classify and visualize terrain data in order to train the algorithm.

Project work will be performed using collaboration tools with version control. Git, through Bitbucket will in large part be the arena for the technical side of the project. Google Drive and related tools for collaboration will be used for documents and resource files. Both the group and the client will have access to Git and Google Drive. Work tasks will be kept track of using Trello, which is integrated into Bitbucket, as well as documented in Drive.

For the purpose of communication, the following technologies were used. For internal communication, planning group meetings and sharing general information the group used Discord. When communication with the product owner and Norwegian Coastal Administration, the group used Slack, as both parties was familiar with it.

2.4.1 Trello

Trello is an collaborative tool used for organizing projects and subtasks into boards. (*Finnegan, M., n.d.)* At its most basic, a board is separated into three columns: to-do, in progress, and completed. This reveals what the team is working on, which group member is working on what, and which tasks have been completed.

Trello was used to influence and infuse our work process with Kanban elements, and it kept us focused on the tasks at hand and was easily adjustable throughout the project. The boards were created to keep tabs on different tasks in IS-304, IS-305, arranging meetings with the product owner, or even to individual taks for each group member.

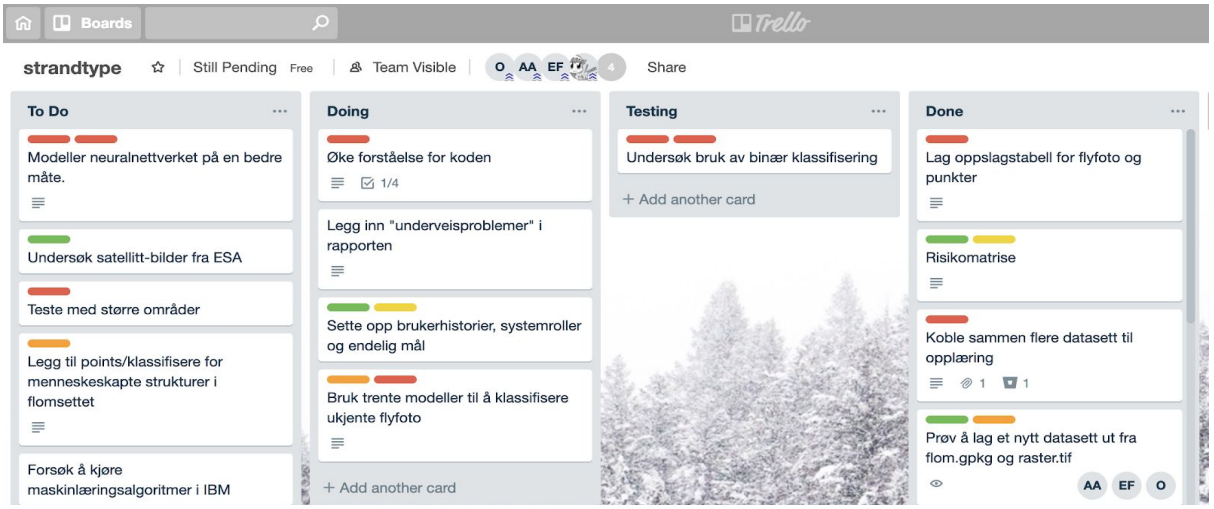

Figure 2: Trello task board

2.4.2 QGIS

QGIS is a user friendly Open Source Geographic Information System (GIS) licensed under the GNU General Public License. QGIS is an official project of the Open Source Geospatial Foundation (OSGeo). It runs on Linux, Unix, Mac OSX, Windows and Android and supports numerous vector, raster, and database formats and functionalities. *(QGIS, n.d.)*

In the project QGIS was used for classifying terrain types such as sand, rocks and coastal bare rock in the dataset. This was needed to generate more data in order to train Enkai. Each member received an area on the map to tag and classify, before combining the files and feeding the information to Enkai.

At first it was difficult to learn the intricacies of QGIS as a tool. QGIS is fairly complex and many of the features included in the system are not immediately obvious in their use. Our first attempts at generating terrain markers were ineffective and did not achieve anything of note. After contacting the NCA, we were given guidelines on how to approach QGIS in an efficient manner.

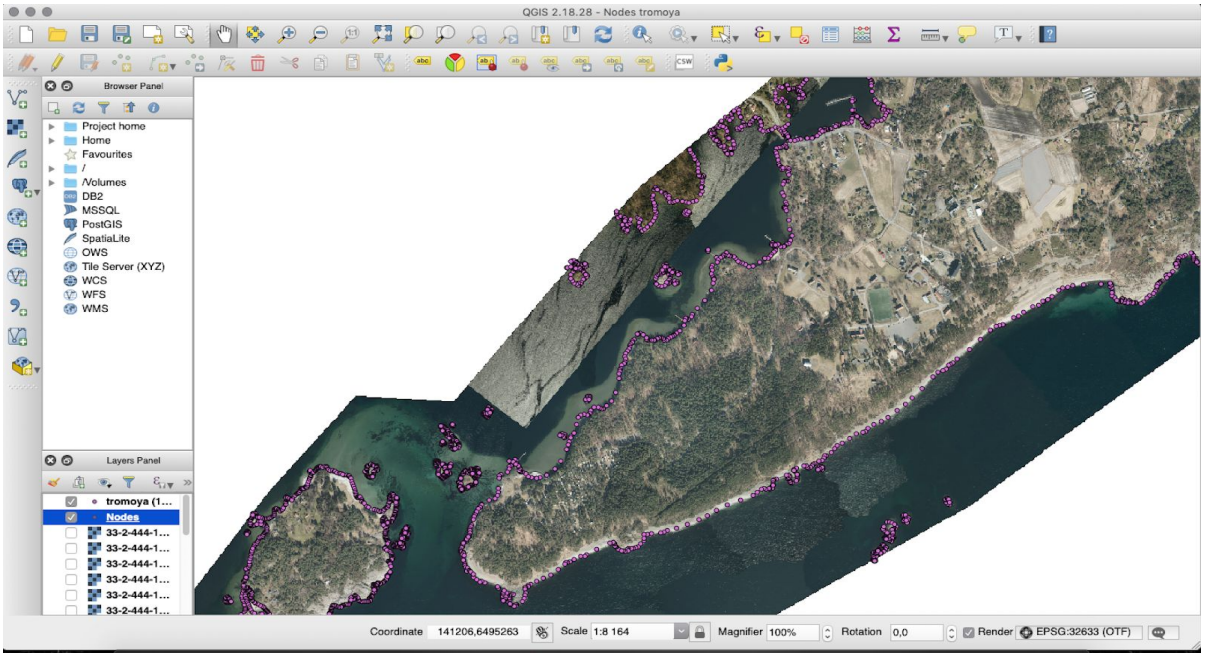

Figure 3: Example of QGIS points over "Tromøya" Dataset .

2.4.3 Anaconda

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS, and Linux. *(Anaconda, n.d.)*

One of the main reasons behind the group using Anaconda Navigator was because it allows for an easy 'point-and-click' method of working with packages and libraries. Instead of installing each package and plugin independently, Anaconda Navigator simplifies the process by allowing the user to install libraries with a universal method and through a common interface. By centralizing libraries through the Navigator environment, the risk of user error and faulty installation is reduced, and the work process is streamlined.

3.0 Process

In this chapter we will reflect on how we ran the project, our experiences and the challenges we faced, how we dealt with them and how we could have done things differently. We will also provide an example of a Sprint, as well as the user story which the project is based on.

3.1 Sprint example

The second Sprint was officially started 23th of January, a day after our first update meeting with our project owners. The focus for this sprint was to work on the remaining tasks from the prior sprints, in addition to downloading QGIS as a tool for classifying areas on our dataset.

3.1.1 Daily scrum

The first daily Scrum meeting of Sprint 2 was held on the 23rd of January, and meetings from this point onward were held semi-regularly, varying from 2 to 5 meetings a week.

3.1.2 Time boxing

The second Sprint was timeboxed from the 23rd of January until 25th of February. The group split up tasks into timeboxed subtasks with start dates and end dates. This sprint was used to work on QGIS and generating more data for our dataset, and also working on the Risk deliverable for IS-305. The Scrum Master delegated each member with tasks to wok on, similar to subtasks.

3.1.3 Sprint backlog

Priority	Task	Status	Estimate	Actual	Start date	End date
1	Work on IS-305 report	Done	8 hours	8 hours	23rd of January	30th of January
2	QGIS points	Done	8 hours	18 hours	25th of January	23rd of February
3	User stories	Doing	4 hours	5 hours	23rd of January	25th of February
4	Binary classification	Done	2 hours	2 hours	13th of February	13 of February
5	Connect points to aerial photos	Done	1 hour	1 hour	18th of February	18th of February
6	Use several point datasets to create a richer set of training images	Done	1 hour	1 hour	24th of February	24th of February

3.1.4 Burndown chart

All planned tasks that were delegated to this Sprint have been completed in a satisfactory manner. While it should be mentioned that some rounding in the resulting actual time spent has occurred, the planned estimates have mostly been accurate, with task 2 as an exception.

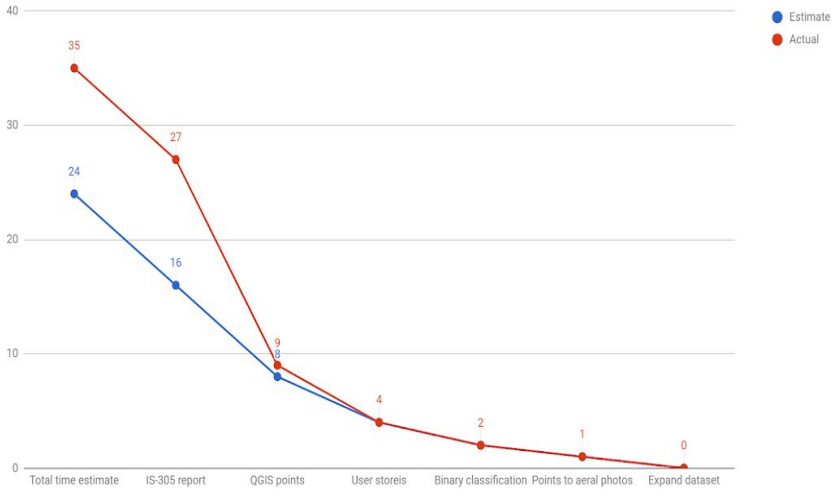

Figure 4: Burndown chart from Sprint 2

3.1.5 Sprint review

Sprint review 2 was held on the 25th of February. The tasks that were set in the Sprint backlog were all finished, although the actual time spent on some of these tasks were a lot higher than initially expected, with the main hurdle being task 2 in the backlog, “QGIS points”. This was due to the group being unfamiliar with QGIS as a tool, and the process of tagging points was more complex than initially expected. The rest of the Sprint went as expected and without any major issues.

3.2 Realism in the project

It is viewed as realistic and attainable to create a completed dataset generated through methods defined in the project, of a defined stretch of the coastline. This is however not essential to the success of the project. The project feels realistic as it is in line with the field of research & development, and in that there is a need for an accurate dataset. The work that will be performed in the project can lead to such a dataset being publicly available in the future.

3.3 Challenges

There are immediate challenges in the project, as machine learning and image recognition are both very complex fields. Even though the project members have some experience with them, this is rather superficial. Considering the research made in this project, it will be very important to document processes and experiences gained. As mentioned in 2.1 Methodology, the group has incorporated elements from Scrum and Kanban in running the project.

3.3.1 Technical challenges

When the initial learning process was started, we had few points of data that could be categorized as sand, so more data was requested. Unfortunately, the new datasets also had skewed ratios of terrain types. Upon running the algorithm learning process on the terrain datasets, the accuracy of the resulting model was not satisfactory. At this point it was uncertain if the poor accuracy was due to the codebase or the dataset itself. These results were discussed with the NCA, and the best route forward is to reduce the amount of factors involved in order to isolate the problem. Instead of

feeding the algorithm with aerial photos, an established picture set of cats and dogs was used, which resulted with an accuracy of 87.69% after 50 epochs; an acceptable result. This indicated that the problem did not lie with Enkai itself, and rather with the dataset, which means that either a low quantity of data or inaccurate terrain classification led to this problem.

3.3.2 Organizational challenges

At the start of the semester, roles and their responsibilities should have been more clearly defined, which most likely would result in better efficiency in terms of work done. As an example, at least 2 members should have had technical responsibilities, instead of just one. This would have divided the workload more fairly and effectively. In addition to this, we should have implemented a test-driven development process from the start of the project, which would have more clearly defined the necessary functions of the application early on, increased the quality of the code, and would have allowed us to implement Continuous Integration tools to verify that the application works as expected.

Our group meetings during the first few Sprints did for the most part not include breaks, which led to fatigue, a lack of focus and led to several group sessions ending prematurely, as discussion became increasingly off topic. We decided to start taking breaks every hour or so during the initial phase of Sprint 3, which turned out to be quite effective. It allowed for an improved workflow and led to more efficient group sessions.

3.4 User stories

In order to create a product that is grounded in realism the feature design of the application will be based on user stories.

In software development, user stories are used as an informal, natural language description of features of a system. *(Wiegers, K. & Beatty, J., 2013)*. The group will discuss and arrange the user stories, in addition to using the MoSCoW method, which is a prioritization technique that stands for four possible priority classifications for the requirements that will be used:

Must have	The requirements needed for the system to reach the minimum functionality required by customers to reach the project goal.
Should have	The requirements are important and should be included in the solution if possible, but it's not mandatory to reach the project goal.
Could have	It is a desirable capability, but one that could be deferred or eliminated. Implement it only if time and resources permit.
Won't have	This indicates a requirement that will not be implemented at this time, but could be included in future release.

The user stories are based on the following two formats.

Format 1	**Format 2**
As a X (role),	In order to Z (benefit),
I want Y (functionality)	As a X (role),
So that Z (benefit)	I want Y (functionality)

As mentioned above, user stories are based on format 1 and format 2. Format 1 is used in most of the user stories, with format 2 being used in a few user stories. Below is an example of our first user story, the rest can be viewed in the appendix.

User Story 1 (format 1)
As a **data scientist**, I want to train a coastline learning algorithm, for the purpose of classifying different coastal environment types.
MoSCoW classification: **Must have**
Reason: Using a machine learning for terrain classification will save time and resources compared to doing it manually.

4.0 Product

This section serves as a demonstration of the primary functions of enkai, an application which utilizes machine learning with geodata and aerial photos to build models to classify terrain types. For a more in-depth presentation of the product, please refer to product.html or product.ipynb in the appendix.

4.1 Ensuring the existence of data

Before any operation is performed, enkai will check the existence of necessary data according to a configuration file. This step is split into two distinct functions. First, ensure that the folder structure is set up, then ensure that data files exist and download them if not (aerial photos, heightmaps, and point sets with classified terrains for use in training).

4.1.1 Folder structure

Building the folder structure is done through a recursive function. The project structure is defined in a configuration YAML file:

```
- data:
  - models
  - subrasters
  - geodata:
      - gpkg
      - aerials
      - heightmaps
```

In Python, this gets represented as a list and dicts:

```
[{
      "data": [
      "models",
      "subrasters",
      {
            "geodata": [
            "gpkg",
            "aerials",
            "heightmaps"
            ]
      }
      ]
}]
```

We can have a look at how the function performs. Every print call below represents an attempted creation of directory. The recursion isn't perfect, as data is attempted to be created twice. However, that is not an issue in practice as it will be created once, only when it doesn't exist.

```
def ensure_folder_structure(structure, d=""):
    if isinstance(structure, list):
        for dir in structure:
            real_d = d
            if isinstance(dir, str):
                d += dir + "/"
            print(build_path(d))
            ensure_folder_structure(dir, d)
            d = real_d
    elif isinstance(structure, dict):
        for dir in structure.keys():
            d += dir + "/"
            print(build_path(d))
            if isinstance(structure[dir], str):
                print(build_path(d + structure[dir] + "/"))
                return
            ensure_folder_structure(structure[dir], d)

structure = [{"data": ["models", "subrasters", {"geodata": ["gpkg", "aerials", "heightmaps"]}]}]
ensure_folder_structure(structure)
```

```
data\
data\models\
data\subrasters\
data\
data\geodata\
data\geodata\gpkg\
data\geodata\aerials\
data\geodata\heightmaps\
```

Figure 5: How ensure_folder_structure works

4.1.2 Data files

The project's data files consist of GeoPackages and GeoTIFF files. These are both binary formats, and as Git commits are based on differences, having these files in the Git-repository would make the commits quite large whenever there's a change in any of these files. Therefore, it was chosen to use a more traditional file host for the data files, and since Google Drive was already in use for documentation, it also covers that use case.

The list of required files is contained in a configuration YAML file. The existence of each data file is checked, and in order to verify the integrity of the data files, a hash check is performed. If the file does not exist locally, the file is downloaded from Google Drive. If the hash does not match, the file is also downloaded from Google Drive again.

Here is an example of a file entry in the YAML file.

```
- &id001
  hash: 64b11160b52994b5443427a39f6af7c15e06728299eb2d19803b9d5fc98cbad7
  name: flom.gpkg
  type: gpkg
  url: https://drive.google.com/open?id=1vomZkF5qFJs-LE2xnxoGdWkLuzXYPYWs
```

`hash` is a sha256 hash of the file, `name` is a desired filename, `type` determines where in the directory structure the file should be saved, and `url` is the Google Drive share URL for the file.

4.2 Generating truth data

Truth data is a set of images, each associated with a terrain type. These images are cropped from the aerial photos, centered around points with terrain classifications.

To reduce bias, the process of creating truth data outputs an equal terrain distribution. Generating truth data is a necessary step as it forms a basis for later operations.

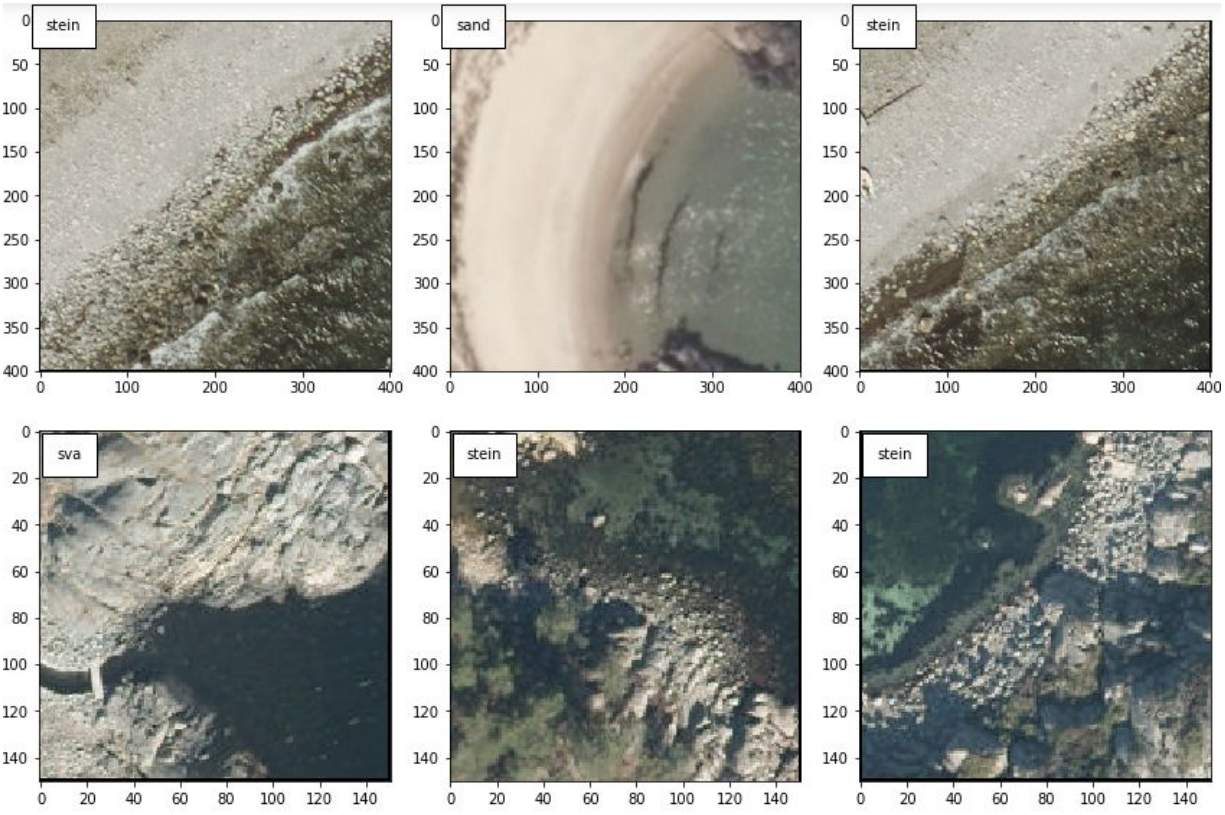

Figure 6: Examples of truth data

4.3 Creating a model

When truth data has been generated, enkai can be used to create a model for classifying new images. The program uses TensorFlow and Keras to create this model based on the truth data. In order to train the model, the input data must be set up in an X, Y fashion, image data becomes X, and associated category becomes Y.

The input data must then be split into training, testing and validation sets. This way, we can use data for training and validating while the model is being fitted, and testing to check the model accuracy after training.

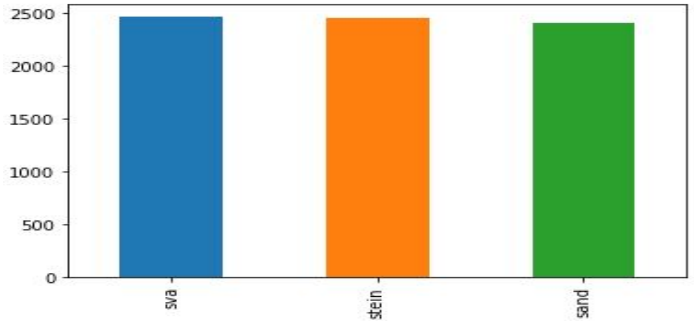

Figure 7: Distribution of categories in the training set

The sets are contained in each their DataFrame. Together with ImageDataGenerators from Keras, we can simply feed the images from the DataFrames into a layered, sequential model.

4.3.1 How an ImageDataGenerator works

ImageDataGenerator is a utility which can augment input images by performing certain operations on it such as flipping, rotating, and zooming. To a machine learning algorithm, an image with slight modifications is essentially a never before seen image. This results in a better foundation for training as there is "more" data to train on.

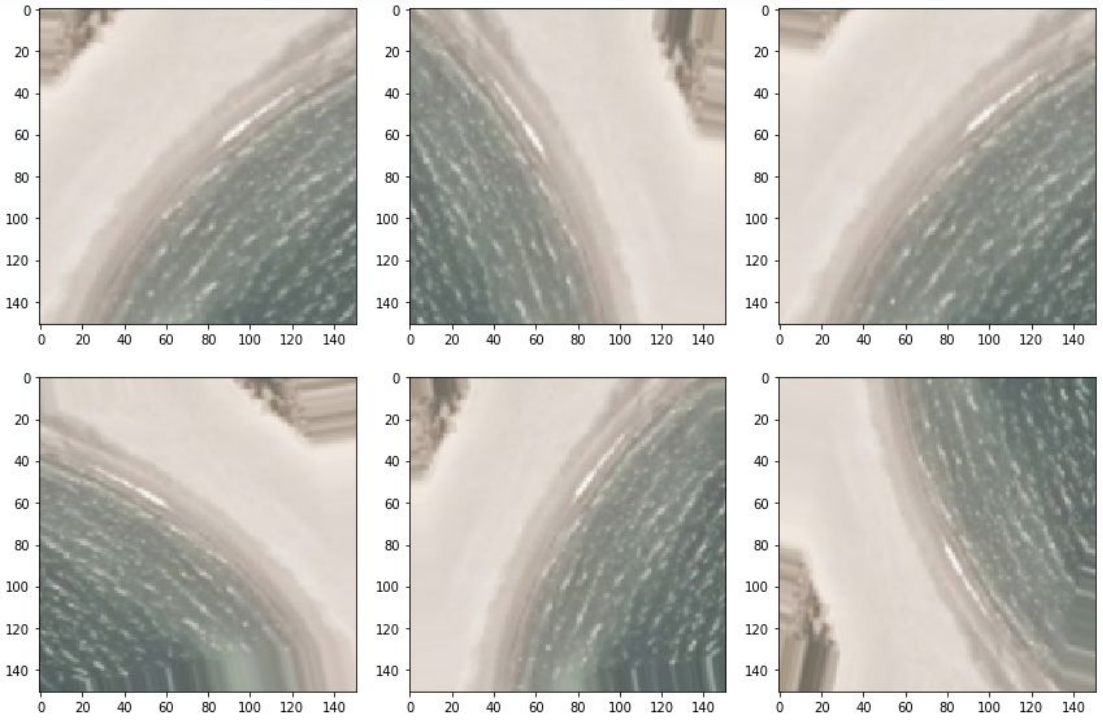

Figure 8: Effects of ImageDataGenerator on a single image

4.4 Fitting the model

After the input data has been set up properly, the next step in the process is to apply it to a sequential model. The particular model in use in this project has 4 primary sequential layers.

The way the model works is that it takes an image as input, the following layers performs feature extraction, or "learns" patterns from the input. At the output end, there is a layer for classification which outputs an array of length 3, with probabilities for each ("sva", "sand", "stein") terrain type.

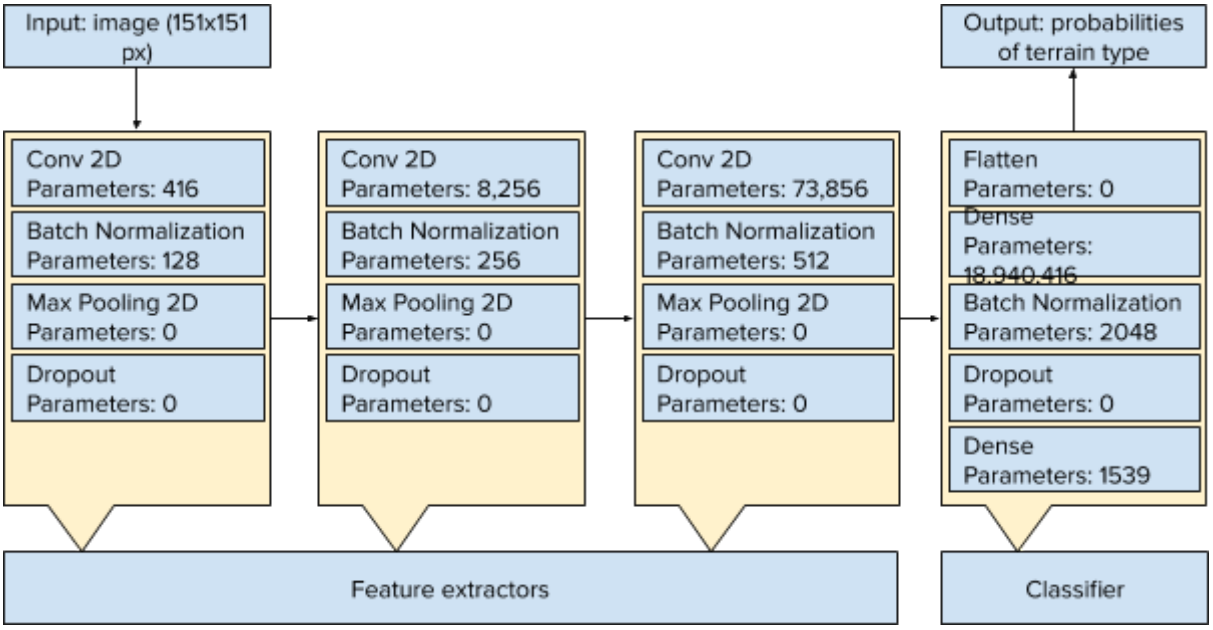

Figure 9: A summary of the model, most parameters are "trainable"

In addition to the model, the learning process is further optimized by having the fitting process stop if the validation loss doesn't decrease after 10 epochs. Likewise, the learning rate is reduced if validation accuracy doesn't increase after 5 epochs.

At this point, everything has been set up for the model to try and fit around the training data, with a validation step using the validation data every epoch. fit_generator runs for 50 epochs, or less if the above callbacks trigger an early stop. This is the *main* learning step for machine learning.

```
EPOCHS = 50
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]

history = model.fit_generator(
    train_generator,
    epochs=EPOCHS,
    shuffle=True,
    validation_data=validation_generator,
    steps_per_epoch=total_train//BATCH_SIZE,
    validation_steps=total_validate//BATCH_SIZE,
    callbacks=callbacks
)
```

Figure 10: The function call which fits the model around the training data

4.5 Evaluating accuracy

After the model has been fitted, it can be evaluated in several ways. The data collected during fitting is the foremost point of evaluation as it shows how accurate the model is against test and evaluation data. However, as this data is reused for every epoch, it is only an indication of accuracy.

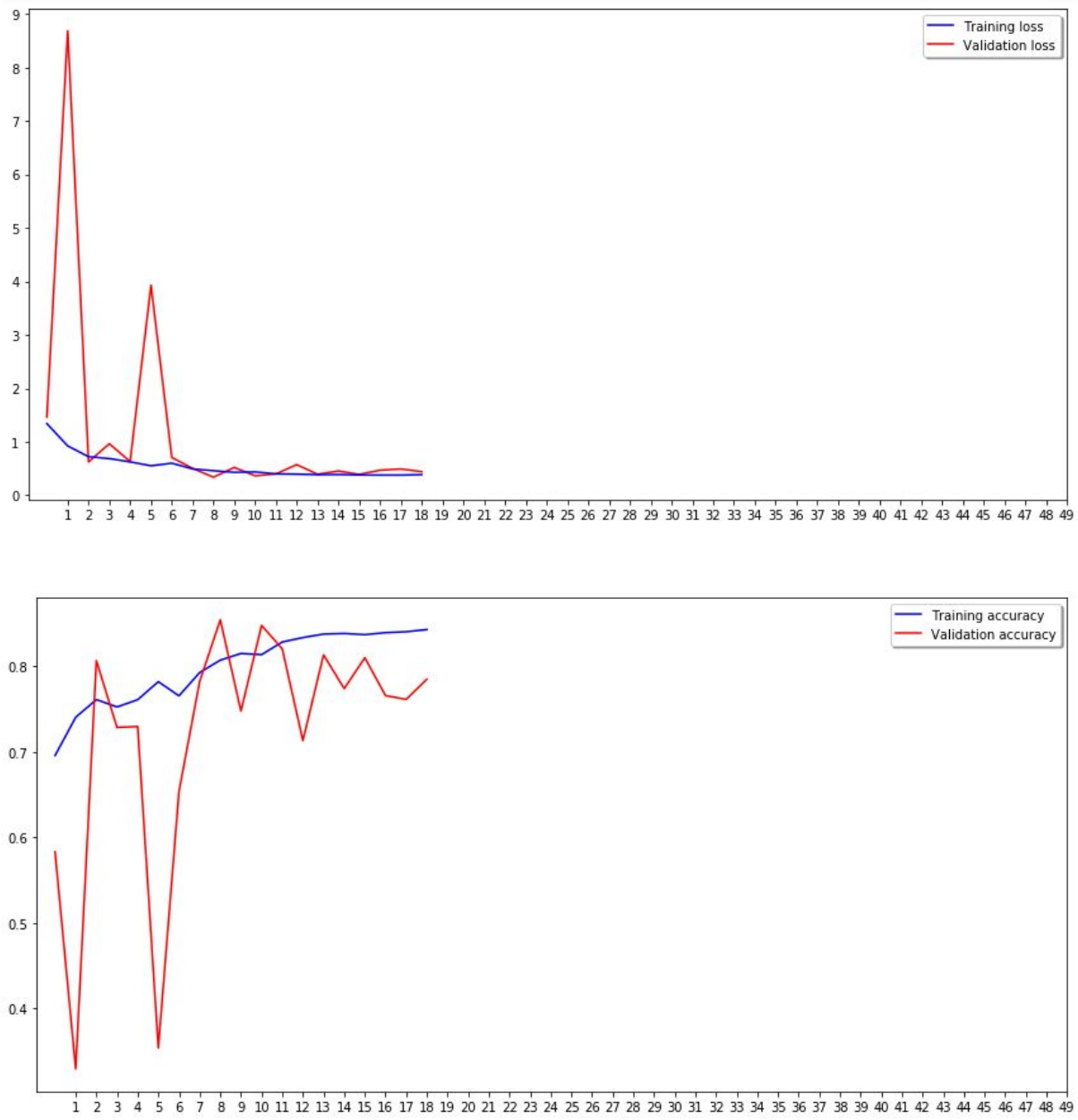

Figure 11: Training and validation loss and accuracy on a run which stopped early after epoch 18.

In order to properly perform evaluation, test data which has not been seen by the model should be used. The model can then run a classification on each image, and a predicted terrain type can be derived from selecting the class which has the highest probability. Unfortunately, even with the best performing model, an accuracy of roughly 38% is achieved on the test images.

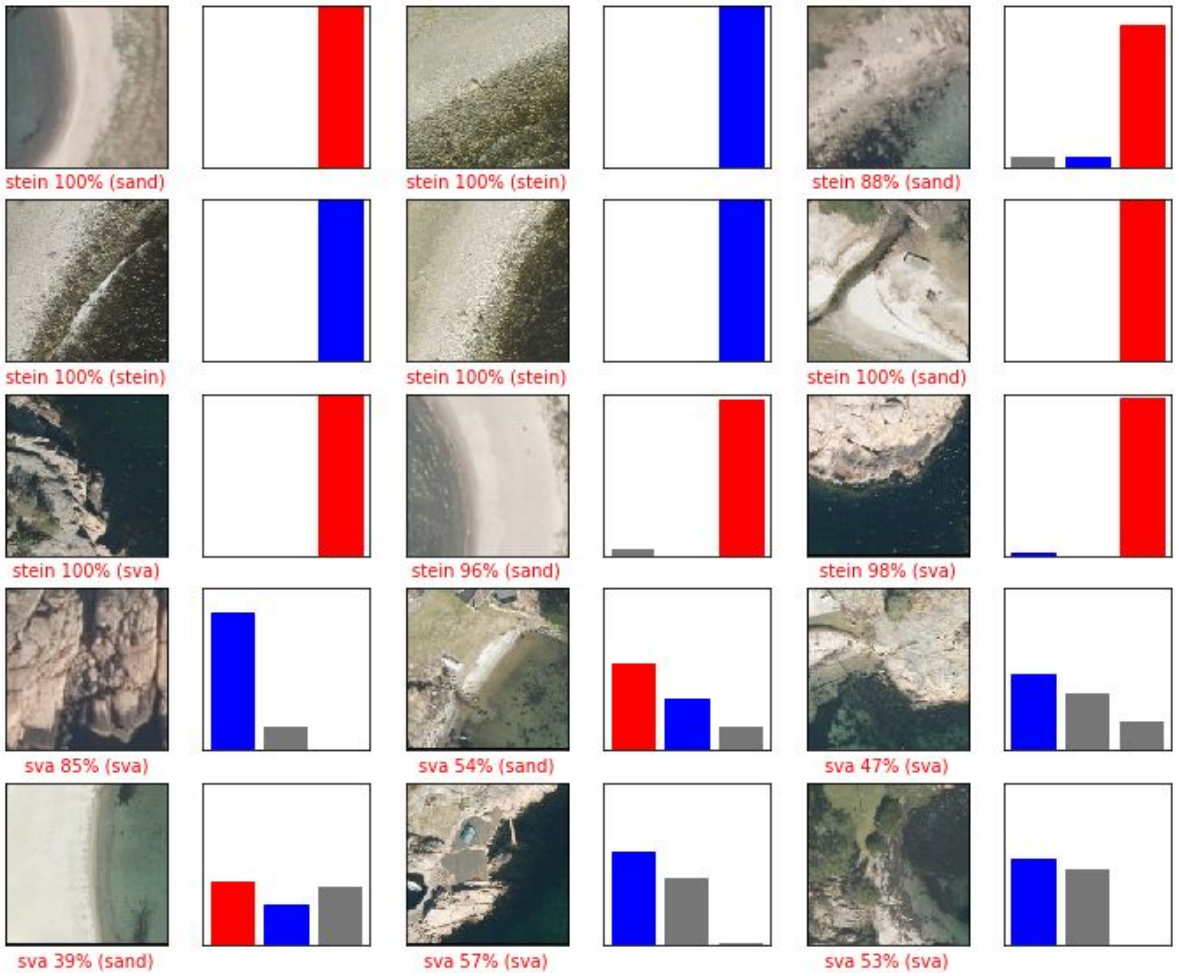

Figure 12: Various predictions made on test data which the model has never seen before. The actual classification is in parentheses.

Given an accurate model, it would be trivial to create a function which takes a set of points from a GeoPackage and aerial photos. The model can then predict the type of terrain at each point and write the prediction to the GeoPackage. This would then result in an accurate dataset with points and associated terrain types.

5.0 Conclusion

This project has allowed us to learn and provide a better understanding for machine learning and the use of a geographic information system. The project has had relatively few formal guidelines, which meant that we had to be more self-sufficient, resulting in novel approaches to the project.

The challenges we faced have been manageable to some degree, but were shaped by our shortcomings. In terms of having a concrete project plan, defining roles and responsibilities and delegation of workload, we have fallen short. The lack of variation in the dataset can therefore partly be thought of as an extension of these previous faults. If we managed to solve our shortcomings either in the planning phase or early in the project, the dataset would held a higher quality, as we would have had more time to look into the issue.

Despite the previously mentioned shortcomings, Enkai is in an acceptable state, and the framework is suitable for future use. The process of adding and using resource files is simple, and it is likely that better results can be achieved with more data.

6. Literature

All screenshots and figures in this document have been made by the group members.

Anaconda. (n.d.) *Anaconda Navigator*. Retrieved April 25th 2019, from https://anaconda.org/anaconda/anaconda-navigator

Boehm, B. & Turner, R. (2012, 8th ed.). *Balancing Agility and Discipline*. Boston: Addison-Wesley Longman Publishing

Collier, K., W. (2011). *Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing.* Pearson Education Publishing

Finnegan. M. (n.d.). *What is Trello? A guide to Atlassian's collaboration and work management tool*. Retrieved April 27th, 2018 from https://www.computerworld.com/article/3226447/what-is-trello-a-guide-to-atlassians-collaboration-and-work-management-tool.html

Norkart. (n.d.). *Om oss.* Retrieved December 7th, 2018, from https://www.norkart.no/om-oss/

Open Source Initiative. (n.d.) *The Open Source Definition (Annotated)*. Retrieved April 25th, 2019, from *https://opensource.org/osd-annotated*

QGIS. (n.d.) *Discover QGIS*. Retrieved April 25th, 2019, from https://qgis.org/en/site/about/index.html

Radigan, D. (n.d.) *What is Kanban?* Retrieved April 12th 2019, from *https://www.atlassian.com/agile/kanban*

University of Agder. (n.d.). *Bacheloroppgave i informasjonssystemer*. Retrieved December 7th 2018, from https://www.uia.no/studieplaner/topic/IS-304-1

Weenink, D. (2004, April 28th). *Epoch*. Retrieved from http://www.fon.hum.uva.nl/praat/manual/epoch.html

Wiegers, K. & Beatty, J. (2013). Software Requirements. Washington, USA: Microsoft Press