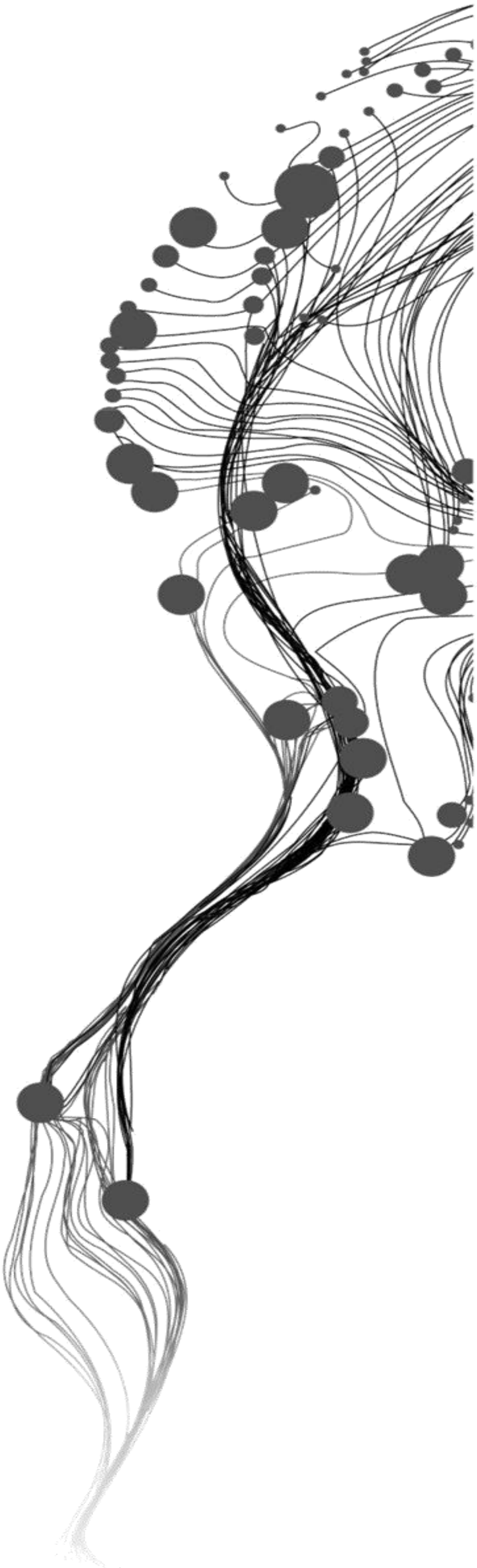


THERMAL SATELLITE FUSION (Downscaling Sentinel-3 LST images using Sentinel-1 SAR images)

SUPERVISORS:

dr. Milad Mahour (Gemeente Rotterdam)
dr. ing. H. Aghababaei (ITC)

Name: Nishit Patel
Student No.: s2872129
Specialization: Geoinformatics
Faculty ITC, University of Twente
Email: n.t.patel@student.utwente.nl

Internship Duration: 15th January 2023 – 15th April 2023

UNIVERSITY OF TWENTE. | Gemeente Rotterdam

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to the Municipality of Rotterdam (Gemeente Rotterdam) for providing me with the opportunity to undertake my internship with them. I am also grateful to Dr. Milad Mahour and Dr. ing. Hossein Aghababaei for their constant guidance and support. This experience has been invaluable in enhancing my professional skills and preparing me for future career opportunities.

Table of Contents

List of Figures

1. Introduction and Motivation

Land Surface Temperature (LST) defined as the radiative skin temperature of the Earth is an important variable used in various studies and different applications such as monitoring and analyzing: urban heat islands (Zakšek & Oštir, 2012), soil moisture (Bai et al., 2019; Zhang et al., 2014), droughts (Wan et al., 2004), vegetation (Julien & Sobrino, 2009), and heat loss from individual buildings. LST is fundamentally different from Air Temperature. Air Temperature is usually measured 10 m above the ground using in-situ sensors.

Ground weather stations are one of the conventional ways to collect LST information. These sparse stations, however, cannot provide reliable information over large areas (Z. L. Li et al., 2013). On the other hand, remote sensing techniques, as an alternative method to estimate LST, allows users to indirectly deduce surface temperature of large areas from space. Although space-based LST can be retrieved from both Thermal Infrared (TIR) and Passive Microwave (PMV) satellites (Wu et al., 2021), the majority of LST information produced by space agencies are derived via TIR-based satellites. For instance, one of the most widely used satellites for deriving LST products is the Moderate Resolution Imaging Spectroradiometer (MODIS) that belongs to National Aeronautics and Space Administration (NASA). MODIS provides daily images with a spatial resolution of 500 m for Visible (VIS), Near Infrared (NIR), and Short-Wave Infrared bands (SWIR) and 1000 m for Thermal Infrared band (TIR). Moreover, in recent years, European Space Agency (ESA) has launched Sentinel-3 SLSTR with similar spatial and temporal characteristics to MODIS. Additionally, for comparatively finer resolution, Landsat-8 offers TIR channel at spatial resolution of 100 m to compute LST information. However, the temporal resolution of Landsat-8 is 16 days.

As seen from above, currently available, and operational remote sensing sensors produce TIR products (e.g., LST) at a much coarser resolution than VIS, NIR, and SWIR products (e.g., spectral indices) from the same satellite. The main reason of this disparity in the band resolutions is because the TIR bands capture radiation at longer wavelengths (Guo et al., 2022). Also, as the emitted wave from the ground surface (TIR) carries less energy compared to the reflected wave (VIS) from the visible region, the spatial resolution of TIR channels is often coarse in order to get enough level of emitted energy at each pixel. It is also important to note the tradeoff between the products' spatial and temporal resolutions (i.e., high spatial resolution information generally corresponds to low temporal information and vice versa). Thus, irrespective of the selected product, the user would have to deal with uncertainties either related to the spatial or temporal domain. These constraints can be removed by using aerial methods like airborne or UAVs to collect data on demand, however this technique can easily become costly. As a result, it becomes interesting to investigate techniques that can assist users in overcoming these constraints and '*Spatial Downscaling*' is one such technique.

Spatial Downscaling, in simple terms, is the process of translating spatial information from coarse to fine resolution. With the help of spatial downscaling, the user can remove the constraints caused by the tradeoffs between the spatial and the temporal resolution and can achieve high spatio-temporal information. The need for LST products with a high spatio-temporal dimension is obvious in urban studies because these areas have high heterogeneity in terms of land surface information, which correlates to high variability in LST (Z. L. Li et al., 2013). Furthermore, with high resolution LST data, it would be possible to observe and monitor accurate temperature information of distinct objects such as fields, roads, buildings which may not be feasible using LST products having coarse spatial resolutions (e.g., 1000m).

2. Related Works and Research Gap

‘Spatial Downscaling’ isn’t a new field of study. There has been a lot of research going on in this domain for the past two decades in the context of LST. Kustas et al. (2003) applied the method called DisTrad (which is based on the least square expression between Normalized Difference Vegetative Index *(NDVI)* and LST) to downscale MODIS LST products from 1000 m to MODIS NDVI resolution (250 m). Later on, Agam et al. (2007) refined the DisTrad algorithm and produced a new method called TsHARP which helped decrease the RMSE value of the DisTrad algorithm from 1.5 °C to a range of 0.67 – 1.35 °C. Additionally, the TsHARP algorithm was also applied to sharpen the Landsat resolution from 120 m to 30 m. The reported RMSE was 2.4 °C.

These techniques were based on traditional statistical methods. However, in the last decade, researchers have tried to use machine learning and deep learning models to downscale LST. For instance, Bindhu et al. (2013) applied a Non-Linear DisTrad (NL-DisTrad) algorithm which was a combination of a hot edge model and an Artificial Neural Network (ANN), to downscale MODIS LST from 1000 m to 60 m. The achieved RMSE was less than 0.96 °C. Analogous to previous statistical methods, NL-DisTrad uses NDVI as a predictor to downscale LST. Further, Li et al. (2019) applied popular machine learning algorithms such as Support Vector Machines (SVM) and Random Forest (RF) to downscale MODIS LST from 1000 m to 90 m. The obtained RMSE values were in the range of 2 – 3 °C. However, in this study, besides NDVI other spectral indices, terrain factors, and land use land cover (LULC) components were also used as predictors. Additionally, there have also been studies by Luo et al. (2021) and Mahour et al. (2017) which integrates statistical methods like Geographically Weighted Regression (GWR) and Co-Kriging to downscale LST products respectively.

A main observation from literature review is that irrespective of the applied method (statistical, machine learning, or deep learning), the predictors (such as Normalized Difference Vegetative Index (NDVI), Normalized Difference Building Index (NDBI), Bare Soil Index (BSI), terrain factors, and much more) used to downscale LST are derived from optical imagery. Some of the significant disadvantages of predictors derived from optical imagery is their reliance on clear weather and failure to provide observations during night-time. There could be a lot of incomplete data due to cloudy conditions, which could be a constraint when building a downscaling model. There is also a need to estimate night-time LST because, alongside day-time LST, night-time LST also acts as an important parameter in analysing urban climatology (Yoo et al., 2022). Furthermore, the observed spatial pattern of LST during day and night is also quite different (Yoo et al., 2022). However, because there are no night-time observations for the optical predictors, estimating night-time LST using such optical predictors would not be feasible. In a nutshell, the above discussed models fail to produce results during night-time and in cloudy conditions due to the nature of the optical imagery.

In order to overcome these limitations imposed due to the nature of optical imagery, radar data can be used for deriving predictors. Radar can penetrate through clouds, and because it is not dependent on solar reflectance, the user can acquire information during both daytime and night-time (J. Li et al., 2018). With the help of freely available Sentinel-1 Synthetic Aperture Radar (SAR) data, we try to model the relationship between the predictors derived from the backscatter of the radar and LST and then use these derived predictors to downscale LST to fine resolution.

3. Methodology

Below flowchart describes the developed framework of the entire internship project. The corresponding sub-sections break down the individual elements of the flowchart and describes them in detail.

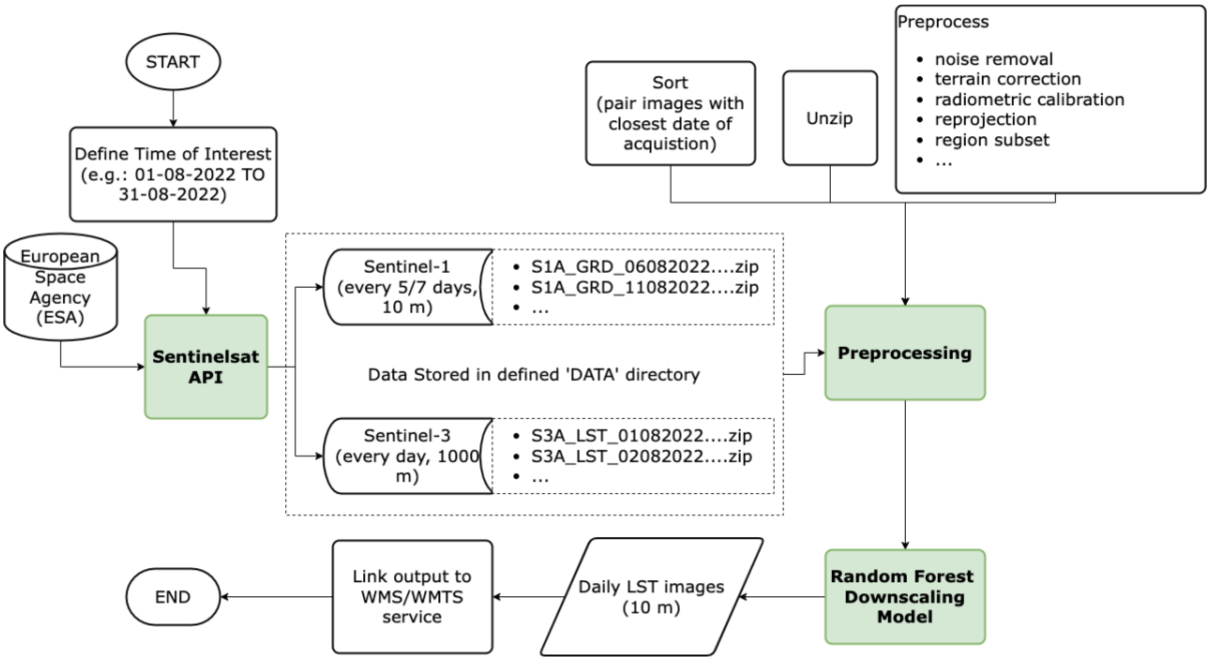


Figure 1: Developed framework for the 'Thermal Satellite Fusion' internship project

As can be inferred from the above flowchart, primarily, the first step is to specify the time range for which the downscaled products are needed. This information is passed to the Python script 'Sentinelsat API', which retrieves the corresponding Sentinel-1 and Sentinel-2 data from the European Space Agency (ESA) database. The collected data must be preprocessed in the following phase. The 'Preprocessing' python script performs three major tasks: sorting data based on acquisition dates, unzipping, and performing various preprocessing functions such as noise removal, terrain correction, radiometric calibration, reprojection, and much more. After the data has been preprocessed, it is ready to be used as input for the downscaling model. The downscaling model generates images of LST with a resolution of 10 m. These LST images are then linked and disseminated as a Web Map Service (WMS)/Web Map Tile Service (WMTS). The interested users can avail of these LST images in their front-end applications such as QGIS, ArcGIS, and various JavaScript-based frameworks.

Sentinelsat API:
The Sentinelsat API (https://sentinelsat.readthedocs.io/en/stable/) is a powerful and adaptable tool for accessing the massive amounts of Earth Observation data collected by the European Space Agency's (ESA) Sentinel satellites. This API, written in Python, allows users to easily search for and download satellite data from the Sentinel missions. As we are interested in Sentinel-1 and Sentinel-3 products, this API serves as an essential tool to the developed framework. Below image shows the various parameters that can be adjusted in the query function of the API.

```
sentinel_1_data = api.query(
                            area_of_interest,
                            beginposition='[2022-08-01T00:00:00.000Z TO 2022-08-31T00:00:00.000Z]',
                            platformname='Sentinel-1',
                            producttype='GRD',
                            filename='S1A_IW*',
                            orbitdirection='ASCENDING'
                           )
```

Figure 2: Example of a Sentinel-1 query using sentinelsat API

```
sentinel_3_data = api.query(
                            area_of_interest,
                            beginposition='[2022-08-01T00:00:00.000Z TO 2022-08-31T00:00:00.000Z]',
                            platformname='Sentinel-3',
                            producttype='SL_2_LST___',
                            filename='S3A*',
                            orbitdirection='DESCENDING'
                           )
```

Figure 3: Example of a Sentinel-3 query using sentinelsat API

As can be seen from the above image, we are interested in Sentinel-1 Ground Range Detected (GRD) and Sentinel-3 SLSTR level-2 LST products. The 'Ascending' orbit direction in the case of Sentinel-1 query collects the data at ~17:30 pm, whereas the 'Descending' orbit direction in the case of Sentinel-3 query collects the data at ~10:30 am.

The developed API python script will collect the queried data and save the data in a zip format at the specified location. It is important to the note that a lot of past data on https://scihub.copernicus.eu/ is stored in a long-term archive (LTA) and is not available to download instantaneously. For the data that is stored in the LTA, the script automatically triggers retrieval of this data from the LTA. Thus, depending upon the input time range, the user might have to wait for some time before all the products are downloaded.

Preprocessing:
Once the data is downloaded, the next step is to preprocess the data. The preprocessing script involves three major tasks: sorting, unzipping, and preprocessing.

- Sorting:

To achieve an accurate mapping function between the predictors and the target variable, we need to appropriately pair up the predictors and the target based upon the closest dates of acquisition. For instance, in the case of month of August 2022, Sentinel-3 image taken on 7th August 2022 should pair up with the Sentinel-1 image taken on 6th August 2022, as Sentinel-3 images are available daily and Sentinel-1 images are taken every 5/7 days. This step decreases the accumulation of uncertainties due to temporal mismatch. However, for some of the images there is indeed some inherent temporal mismatch since Sentine-1 data is not available daily.

- Unzipping:

Here we simply unzip the data.

- Configuring snappy on AzureML:

Before moving on to the main preprocessing steps, first we require to configure snappy module. ESA SNAP (Sentinel Application Platform) is a comprehensive software package designed for the processing and analysis of data collected by Sentinel satellite missions. SNAP is written in Java; however, ESA also provides a python module called 'snappy' built on top of SNAP that allows users to perform same processing tasks but with python. However, since it is built on top of Java, setting up snappy is not straightforward.

Following are the developed steps to configure snappy module on AzureML:

1. *Create a conda environment 'python_36' (or whatever name) with python version 3.6 (environment should be saved in /anaconda/envs/<env_name>)*
2. *Download the ESA SNAP linux .sh file from https://step.esa.int/main/download/snap-download/ using 'wget' command in terminal ('wget https://download.esa.int/step/snap/9.0/installers/esa-snap_sentinel_unix_9_0_0.sh')*
3. *Change the permissions of the file to make sure you have rights to execute the file using 'chmod 777 <file_name>' command*
4. *Run the installation process using './<file_name>' command*
5. *Once the installation runs, you'll get a lot of choices (similar to when you install an application)*
6. *Proceed with default choices except when asked 'Configure SNAP for Python' select 'NO' as we'll do this manually later. Alternatively, you can provide a path here to your python interpreter which should be saved in the /anaconda/envs/python_36/bin/*
7. *The setup should be complete and when asked 'Run SNAP desktop and configure Python' select 'NO'*
8. *The required SNAP files to configure snappy are now downloaded and should be available at '/home/azureuser/snap'*

```
azureuser@cpu-nishit:~/cloudfiles/code/Users/160911$ cd /home/azureuser/
azureuser@cpu-nishit:~$ ls -a
.                                     .bash_logout   .ipython          .ssh
..                                    .bashrc        .java             .vscode-server
.Rprofile                             .cache         .jupyter          .wget-hsts
.azure                                .conda         .local            cloudfiles
.azureml                              .config        .profile          localfiles
.azureml_remote_websocket_server      .dotnet        .python_history   readme
.bash_history                         .gnome         .snap             snap
```

9. *If you check the bin folder in '/home/azureuser/snap/bin', you will find a 'snappy-conf' command which is the command we will use to configure snappy*

```
azureuser@cpu-nishit:~$ cd snap
azureuser@cpu-nishit:~/snap$ ls -a
 .               'SNAP Configuration Optimiser.desktop'   bin   platform   s3tbx
 ..              'SNAP Desktop.desktop'                   etc   rstb       snap
 .install4j       THIRDPARTY_LICENSES.txt                 ide   s1tbx
 LICENSE.txt      VERSION.txt                             jre   s2tbx
azureuser@cpu-nishit:~/snap$ cd bin
azureuser@cpu-nishit:~/snap/bin$ ls -a
.                 gpt              pconvert.vmoptions    snappy-conf
..                gpt.vmoptions    snap                  uninstall
SNAP_icon_48.jpg  pconvert         snap-conf-optimiser
```

10. *Now all we must do is run this command with our python interpreter, so, 'snappy-conf /anaconda/envs/python_36(or whatever env_name you selected)/bin/python3.6'*
11. *The configuration should be done without any errors and a corresponding 'configuration done' message should be displayed. If there are some errors, you can google them as https://forum.step.esa.int/ has good resources on how to deal with certain errors*
12. *Once snappy is configured, you'll find a '.snap' folder located at '/home/azureuser/' (please use 'ls -a' instead of 'ls' to find the hidden '.' files)*
13. *Inside the '.snap' folder should be /snap-python/snappy located*

```
azureuser@cpu-nishit:/anaconda/envs/python_36/bin$ cd /home/azureuser/.snap/snap-python
azureuser@cpu-nishit:~/.snap/snap-python$ ls -a
.  ..  build  snappy  snappy.properties
azureuser@cpu-nishit:~/.snap/snap-python$ cd snappy
azureuser@cpu-nishit:~/.snap/snap-python/snappy$ ls -a
.            jdl.cpython-36m-x86_64-linux-gnu.so  jpyconfig.py  snappyutil.log
..           jpy-0.9.0.dist-info                  jpyutil.py    snappyutil.py
__init__.py  jpy-0.9.0.jar                        lib           testdata
__pycache__  jpy.cpython-36m-x86_64-linux-gnu.so  setup.py      tests
examples     jpyconfig.properties                 _snappy.ini
```

14. *Here, you can launch snappy at this level by calling in your python interpreter (remember the conda environment we created) through terminal (skip this if you don't know what this means)*
15. *However, your python interpreter still doesn't recognise the configured snappy folder on your system. To do so, change directory to the snappy folder, i.e., 'cd /home/azureuser/.snap/snap-python/snappy' and then type '/anaconda/envs/python_36/bin/python3.6 (the path to your python interpreter) setup.py install'*
16. *Now, the snappy library should be available in the 'site-packaged' folder of the conda environment and should be recognisable to the python interpreter*
17. *Now you can launch this environment anywhere (vscode, jupyterlab) and 'import snappy' should work*
18. *Bear in mind vscode searches for python interpreters and environments in different way than jupyterlab. Thus, if you want to work on jupyterlab, you'll need to define the kernel for the created conda environment. To do so, first you activate the conda environment using 'source activate python_36 (or whatever env name you selected)' and then run 'python -m ipykernel install --user --name=<kernel_name> --display-name=<display_name>'. Once you refresh the kernel, you should be able to see your kernel with the chosen display name and this kernel should have python 3.6 and snappy.*

As a final note, it is possible to install snappy for python versions greater than 3.6, but that requires a manual build of 'jpy' (java-python bridge) (more information: https://github.com/jpy-consortium/jpy). Here, you'd have to download the repo and build a wheel with openjdk 8 and then copy the built wheel to the '~/.snap/snap-python/snappy' directory. After this you can run the configuration command again and you should have working snappy for higher python versions (this was tested only on local computer and not on azureML. It might be possible that using this you can modify the existing 3.8/3.10 python environments to recognise snappy)

- Preprocessing:

Now that snappy has been configured, we are ready to proceed with preprocessing of Sentinel-1 and Sentinel-3 data using snappy. The undertaken steps for preprocessing Sentinel-1 and Sentinel-3 are as follows:

A. Sentinel-1 GRD preprocessing:
   1. Subset region of interest
   2. Apply Orbit File
   3. Thermal Noise Removal
   4. Radiometric Calibration
   5. Speckle Filtering
   6. Terrain Correction
   7. Linear to db conversion

B. Sentinel-3 level-2 LST preprocessing:
   1. Subset region of interest
   2. Reprojection
   3. Spectral Subset

Downscaling Model:
Now that the data has been preprocessed, predictors that explain spatial variability in LST can be derived from the preprocessed data.

- Predictors:

As seen in the 'Related Works' section, Normalized Difference Vegetation Index (NDVI) has been frequently used as a predictor in downscaling models. However, due to the nature of optical imagery, it is not prudent to use NDVI as a predictor in these models. As an alternative, we propose to use Radar Vegetation Index (RVI) developed during this internship as a predictor to downscale LST.

The effect of radar parameters such as incidence angle, wavelength, and polarization, as well as surface parameters such as surface roughness, soil moisture, and surface dielectric constant, on radar backscatter, has long been established in the studies done by Benallegue et al. (1995), Hoeben et al. (1997), and Ulaby (1974). Models like the 'Oh' Model (Oh et al., 1992) and the 'IEM' model (Fung et al., 1992) have been well-researched for soil moisture retrieval studies. All these studies make use of backscattering coefficients derived from different polarization modes (HH, VV, HV, VH), where polarization modes refer to the direction in which the electric field of radar signal is transmitted and received (i.e., V for Vertical and H for Horizontal). Variables such as vegetation cover, soil moisture, and dielectric constant affect the radar backscatter and can also explain the spatial variability in LST. Thus, here we use our developed RVI, which tries to explain the spatial variability in LST as a predictor for the downscaling model.

The formula for RVI as developed during the internship (based on the work of Dr. Milad Mahour),

$$RVI = 0.5 \times \left(\frac{vv - 4 \times vh}{vv + 4 \times vh} + 1\right)$$

where, 'VV' and 'VH' bands are in linear units

Figure 4: Radar Vegetation Index (RVI) image over Rotterdam for 11th August 2022

The range of RVI is in between 0 to 1, where higher vegetation corresponds to low values of RVI, and urban areas correspond to higher values of RVI. Below image shows the correlation between LST and RVI at 1000 m.

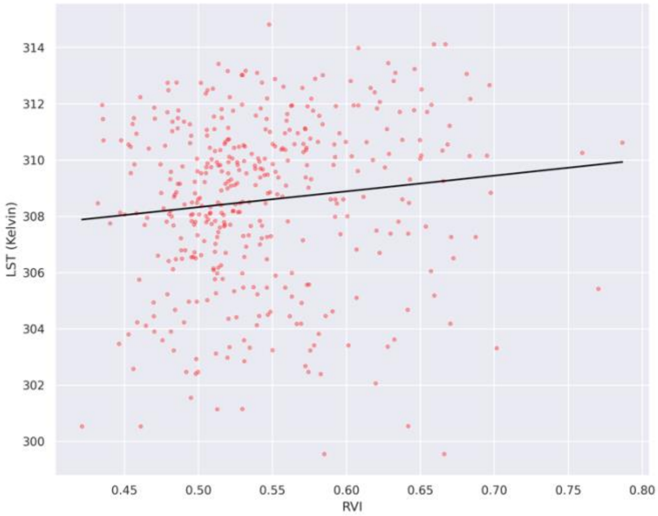
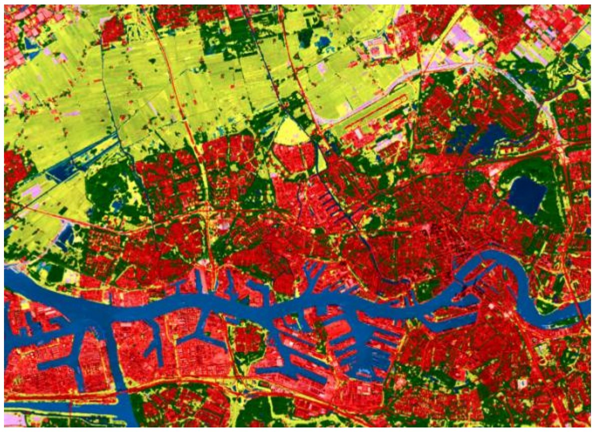


Figure 5: Scatterplot with regression line between LST and RVI at 1000 m for 11th August 2022

However, as one would rightly assume, single predictor is not enough to model a complex spatial phenomenon such as LST. Thus, some ancillary data needs to be added which could help the model recognize certain patterns. In this internship project, Digital Surface Model (DSM) and Land Use Land Cover (LULC) data was also added as predictors to the downscaling model. DSM integrates height and building footprint information to the model which is essential in estimating LST as the height of the objects could impact the amount of absorbed and/or radiated heat. On the other hand, LULC integrates information on different land use and land cover classes present in the study area. Different type of land use and land cover classes tend to have different values for LST and thus it is useful to add this information to the model so that the model can recognize the underlying patterns.

(a)



(b)

Figure 6: Ancillary data used as predictors

(a) Digital Surface Model (source: AHN3), (b) Land Use Land Cover (source: ESA)

- Downscaling Algorithm:

The crux of the downscaling algorithm can be explained in the following three steps:

1. Upscale/resample the predictors to the target variable's coarse resolution
2. Train a model to find the relationship between the predictors and the target at coarse resolution
3. Input the predictors at fine resolution to the trained model and estimate fine resolution LST

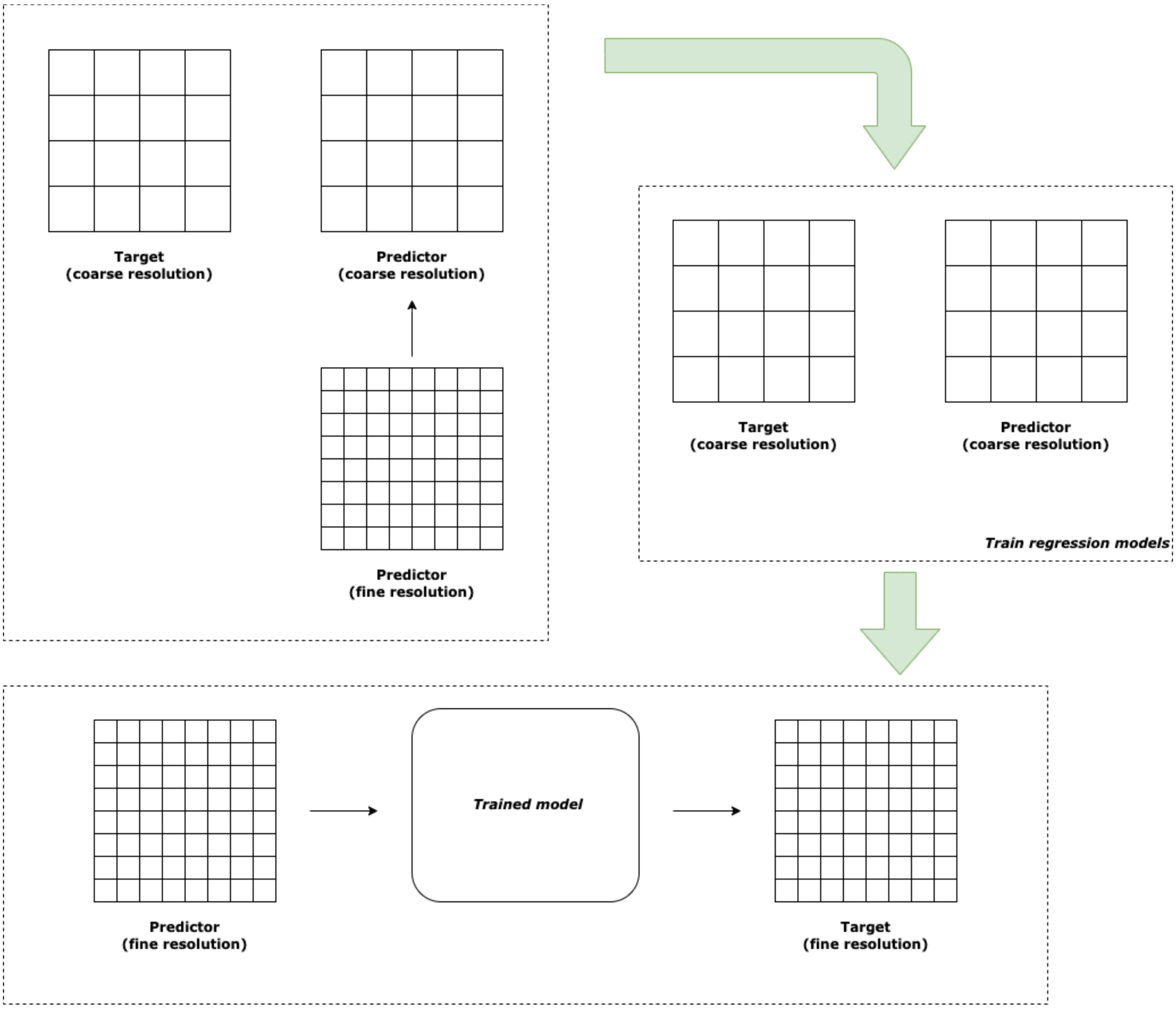


Figure 7: Basic Downscaling algorithm

The above-described downscaling algorithm can be implemented using any regression model/s. For this internship, Random Forest (RF) model was selected as the backbone algorithm to implement downscaling. The ongoing research has shown machine learning to be the state-of-the-art for downscaling problems. Random Forest is a traditional machine learning algorithm that uses an ensemble of decision trees for classification and regression tasks. It is a non-linear method that constructs a large number of decision trees by randomizing and de-correlating the data to produce diverse set of trees (Hutengs & Vohland, 2016). These decision trees are then averaged to form a robust and accurate prediction model. The Random Forest algorithm, widely used in classification and regression tasks, has demonstrated effectiveness in dealing with high-dimensional, noisy, and complex data sets. Due to these qualities, Random Forest was selected as the backbone algorithm to the downscaling model.

In line with standard practice, the dataset was divided into training and testing sets using a 0.2 ratio. However, due to limited amount of data, there wasn't enough data to set aside for a validation set to tune and determine the optimal hyperparameters of the model. RandomizedSearchCV was used as an alternative to identify the best hyperparameters of the model from a predefined range or grid of hyperparameter values. The use of RandomizedSearchCV allowed for efficient hyperparameter tuning while maximizing the utilization of the available data for training purpose.

4. Results

In accordance with the methodology outlined in the above section, the entire downscaling framework was implemented to downscale the Sentinel-3 1000 m LST product and generate daily 10 m LST images for the period of August 2022.

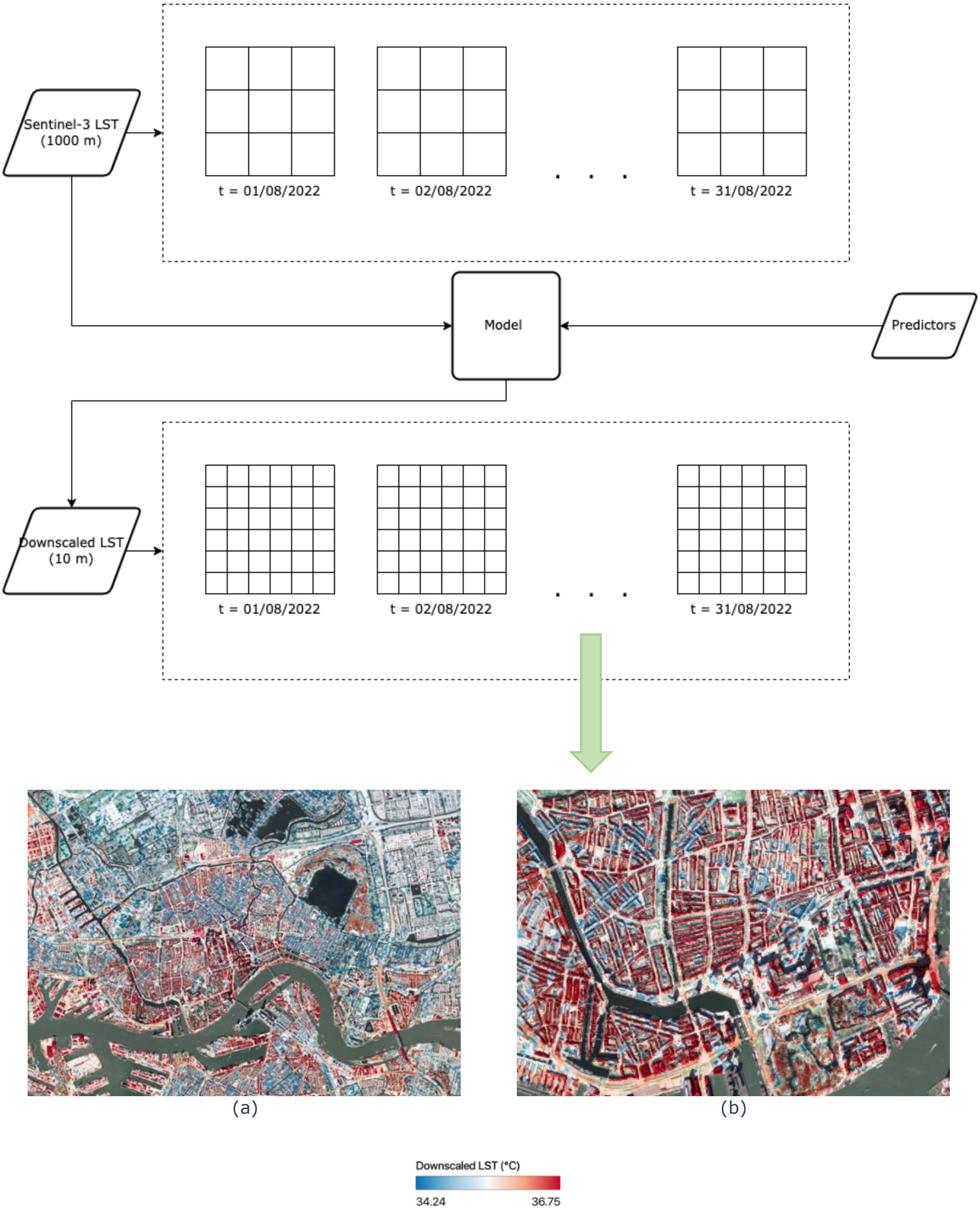


Figure 8: Schematic diagram of the inputs and the outputs of the downscaling mode

(a) Downscaled LST (11th August 2022) over Rotterdam, (b) Zoomed in view of (a)

The above results obtained from the developed downscaling model for the day of 11th August 2022 demonstrate that the buildings are effectively delineated, with higher temperatures observed in comparison to the surrounding areas. This observation is consistent with the anticipated urban heat island effect. Additionally, certain road networks also exhibit elevated temperatures. Conversely, areas with vegetation exhibit relatively lower temperatures. Through qualitative evaluation, the results can be deemed accurate.

Although qualitative analysis can provide insights into the performance of a model, it may not be sufficient to assess the model's overall accuracy. Therefore, it is recommended to employ quantitative metrics to evaluate the model's performance. In this regard, the output of the developed downscaling model was compared to Landsat-8 LST at 30 m (the original resolution of Landsat-8 LST product is 100 m, however, United States Geological Survey (USGS) also resamples and distributes the Landsat-8 LST data at 30 m). The obtained correlation coefficient was 0.4 (*Figure 9*), indicating suboptimal model performance. This indeed reveals some shortcomings of the model which are further discussed in the subsequent section. However, it should be noted that comparing results with different sensor, such as downscaled LST from Sentinel-3 to Landsat-8, may not be entirely accurate due to inherent differences in sensor characteristics, which in turn can significantly impact the quantitative metrics.

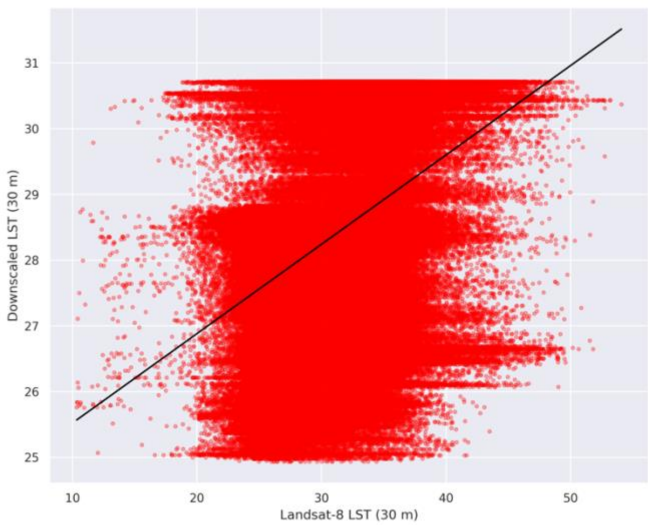


Figure 9: Scatterplot with regression line between Downscaled LST and Landsat 8 LST at 30 m

5. Discussion

As discussed in the previous section, the developed downscaling model has some limitations. Specifically, the range of the downscaled LST values is narrowed compared to the original Sentinel-3 LST range. The downscaled LST values range between 34 °C and 37 °C, while the original Sentinel-3 LST values range between 32 °C and 39 °C. This discrepancy of approximately 2 °C at both extreme ends is a common pattern observed across all generated downscaled products. This is due to the machine learning models' inherent nature, where extreme values lack sufficient training samples and thus these models tend not to predict these values to achieve higher accuracy. However, this issue can be addressed by applying residual correction, where residuals represent the information left behind by the model and can be added back to the model prediction to account for lost information. The developed downscaling model incorporates residual correction. Nonetheless, due to the significant difference in scale between the 1000 m and 100 m resolutions, these residuals may product a boxy effect that may or may not be desirable. An alternative approach to addressing this issue is to expand the number of features used in the model. Given the complexity and spatial variability of LST, it may not be possible to capture and account for all relevant factors using only three predictors, as was done in this study. To improve the model's performance, additional features, such as neighboring values, texture information, sky-view factor, and other relevant variables, can be incorporated.

In addition, a step-by-step downscaling approach may be utilized to enhance the results. Instead of directly transitioning from 1000 m to 10 m spatial resolution, the step-by-step method first downscales from 1000 m to 500 m, retrains the model with the predicted values at 500 m, and repeats this process until the algorithm reaches the desired resolution of 10 m. This technique was employed by X. Li et al. (2022) and yielded improved outcomes compared to methods based on traditional downscaling algorithm. Moreover, this approach mitigates errors that may arise from the 'scale-invariance' assumption (which is the base of the traditional downscaling algorithm), where the relationship between the predictors and the target is assumed to remain invariant with variation in scale, which may not always hold true in real-world scenarios.

It is also worth noting that deep learning-based regression models can also be employed for downscaling LST, although research on these models in this context has been limited. Another crucial aspect to consider is that the daily Sentinel-3 LST images are not always of optimal quality, as they are often contaminated by clouds. While the developed framework includes a system to identify and remove such contaminated pixels, a more effective approach would involve using machine learning-based time series interpolation methods to reconstruct full time series of Sentinel-3. This would not only yield more consistent results but also generate sufficient high-quality data to train a large-scale deep learning-based downscaling architecture.

6. Conclusion

The framework proposed in this internship project aims to generate high spatio-temporal LST products from Sentinel-3 1000 m LST data using RVI, DSM, and LULC as predictors. The results demonstrate that this framework is effective and can be used for practical applications. It is important to note that the current performance of the model is suboptimal; however, the framework’s design is flexible enough to incorporate improvements to the backbone model with ease, using some of the ideas mentioned in the aforementioned section. Overall, this study presents a promising approach for generating high spatio-temporal resolution LST products, which could aid the municipality in a lot of applications such as monitoring urban heat islands and energy loss from individual features such as buildings, parks, and roads. Furthermore, the generated products can also be used as a validation tool to measure the effectiveness of various heat and energy related sustainable development projects.

7. References

Agam, N., Kustas, W. P., Anderson, M. C., Li, F., & Neale, C. M. U. (2007). A vegetation index based technique for spatial sharpening of thermal imagery. *Remote Sensing of Environment*, *107*(4), 545–558. https://doi.org/10.1016/j.rse.2006.10.006

Bai, L., Long, D., & Yan, L. (2019). Estimation of Surface Soil Moisture With Downscaled Land Surface Temperatures Using a Data Fusion Approach for Heterogeneous Agricultural Land. *Water Resources Research*, *30 m*, 1105–1128. https://doi.org/10.1029/2018WR024162

Benallegue, M., Taconet, O., Vidal-Madjar, D., & Normand, M. (1995). The use of radar backscattering signals for measuring soil moisture and surface roughness. *Remote Sensing of Environment*, *53*(1), 61–68. https://doi.org/10.1016/0034-4257(94)00113-2

Bindhu, V. M., Narasimhan, B., & Sudheer, K. P. (2013). Development and verification of a non-linear disaggregation method (NL-DisTrad) to downscale MODIS land surface temperature to the spatial scale of Landsat thermal data to estimate evapotranspiration. *Remote Sensing of Environment*, *135*, 118–129. https://doi.org/10.1016/j.rse.2013.03.023

Fung, A. K., Li, Z., & Chen, K. S. (1992). Backscattering from a randomly rough dielectric surface. *IEEE Transactions on Geoscience and Remote Sensing*, *30*(2), 356–369. https://doi.org/10.1109/36.134085

Guo, F., Hu, D., & Schlink, U. (2022). A new nonlinear method for downscaling land surface temperature by integrating guided and Gaussian filtering. *Remote Sensing of Environment*, *271*(August 2021), 112915. https://doi.org/10.1016/j.rse.2022.112915

Hoeben, R., Troch, P. A., Su, Z., Mancini, M., & Chen, K. S. (1997). Sensitivity of radar backscattering to soil surface parameters: A comparison between theoretical analysis and experimental evidence. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *3*, 1368–1370. https://doi.org/10.1109/igarss.1997.606449

Hutengs, C., & Vohland, M. (2016). Downscaling land surface temperatures at regional scales with random forest regression. *Remote Sensing of Environment*, *178*, 127–141. https://doi.org/10.1016/j.rse.2016.03.006

Julien, Y., & Sobrino, J. A. (2009). The Yearly Land Cover Dynamics (YLCD) method: An analysis of global vegetation from NDVI and LST parameters. *Remote Sensing of Environment*, *113*(2), 329–334. https://doi.org/10.1016/j.rse.2008.09.016

Kustas, W. P., Norman, J. M., Anderson, M. C., & French, A. N. (2003). Estimating subpixel surface temperatures and energy fluxes from the vegetation index-radiometric temperature relationship. *Remote Sensing of Environment*, *85*(4), 429–440. https://doi.org/10.1016/S0034-4257(03)00036-1

Li, J., Wang, S., Gunn, G., Joosse, P., & Russell, H. A. J. (2018). A model for downscaling SMOS soil moisture using Sentinel-1 SAR data. *International Journal of Applied Earth Observation and Geoinformation*, *72*(July), 109–121. https://doi.org/10.1016/j.jag.2018.07.012

Li, W., Ni, L., Li, Z. L., Duan, S. B., & Wu, H. (2019). Evaluation of machine learning algorithms in spatial downscaling of modis land surface temperature. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *12*(7), 2299–2307. https://doi.org/10.1109/JSTARS.2019.2896923

Li, X., Zhang, G., Zhu, S., & Xu, Y. (2022). Step-By-Step Downscaling of Land Surface Temperature Considering Urban Spatial Morphological Parameters. *Remote Sensing*, *14*(13). https://doi.org/10.3390/rs14133038

Li, Z. L., Tang, B. H., Wu, H., Ren, H., Yan, G., Wan, Z., Trigo, I. F., & Sobrino, J. A. (2013). Satellite-derived land surface temperature: Current status and perspectives.

Remote Sensing of Environment, *131*, 14–37. https://doi.org/10.1016/j.rse.2012.12.008

Luo, X., Chen, Y., Wang, Z., Li, H., & Peng, Y. (2021). Spatial Downscaling of MODIS Land Surface Temperature Based on a Geographically and Temporally Weighted Autoregressive Model. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 7637–7653. https://doi.org/10.1109/JSTARS.2021.3094184

Mahour, M., Tolpekin, V., Stein, A., & Sharifi, A. (2017). A comparison of two downscaling procedures to increase the spatial resolution of mapping actual evapotranspiration. *ISPRS Journal of Photogrammetry and Remote Sensing*, *126*, 56–67. https://doi.org/10.1016/j.isprsjprs.2017.02.004

Oh, Y., Sarabandi, K., & Ulaby, F. T. (1992). An empirical model and an inversion technique for radar scattering from bare soil surfaces. *IEEE Transactions on Geoscience and Remote Sensing*, *30*(2), 370–381. https://doi.org/10.1109/36.134086

Ulaby, F. T. (1974). Radar Measurement of Soil Moisture Content. *IEEE Transactions on Antennas and Propagation*, *22*(2), 257–265. https://doi.org/10.1109/TAP.1974.1140761

Wan, Z., Wang, P., & Li, X. (2004). Using MODIS Land Surface Temperature and Normalized Difference Vegetation Index products for monitoring drought in the southern Great Plains, USA. *International Journal of Remote Sensing*, *25*(1), 61–72. https://doi.org/10.1080/0143116031000115328

Wu, P., Yin, Z., Zeng, C., Duan, S. B., Gottsche, F. M., Ma, X., Li, X., Yang, H., & Shen, H. (2021). Spatially Continuous and High-Resolution Land Surface Temperature Product Generation: A review of reconstruction and spatiotemporal fusion techniques. *IEEE Geoscience and Remote Sensing Magazine*, *9*(3), 112–137. https://doi.org/10.1109/MGRS.2021.3050782

Yoo, C., Im, J., Cho, D., Lee, Y., Bae, D., & Sismanidis, P. (2022). Downscaling MODIS nighttime land surface temperatures in urban areas using ASTER thermal data through local linear forest. *International Journal of Applied Earth Observation and Geoinformation*, *110*(May), 102827. https://doi.org/10.1016/j.jag.2022.102827

Zakšek, K., & Oštir, K. (2012). Downscaling land surface temperature for urban heat island diurnal cycle analysis. *Remote Sensing of Environment*, *117*, 114–124. https://doi.org/10.1016/j.rse.2011.05.027

Zhang, F., Zhang, L. W., Shi, J. J., & Huang, J. F. (2014). Soil moisture monitoring based on land surface temperature-vegetation index space derived from MODIS data. *Pedosphere*, *24*(4), 450–460. https://doi.org/10.1016/S1002-0160(14)60031-X

8. Self-Reflection

Description of the Host Organization:
My internship project was carried out at the Municipality of Rotterdam (Gemeente Rotterdam). I was privileged to work as a member of the Team Advanced Analytics under the guidance and supervision of Dr. Milad Mahour.

The Team Advanced Analytics plays a vital role in improving the management of the city and supporting colleagues within the organization. Specifically, the team leverages advanced analytical techniques and tools to aid the municipality in the policy and decision-making process. Furthermore, the team collaborates with other departments to identify data needs, ensure data quality and consistency, and promote data-driven practices across the municipality.

Project Description and Deliverables:
The 'Thermal Satellite Fusion'/'Thermal Data Distribution' project has the objective of providing users with highly detailed (in terms of both spatial and temporal resolution) Land Surface Temperature (LST) products. This project aims to aid the sustainability department of the municipality in identifying energy and heat loss from individual features such as buildings, roads, and green areas.

As a part of this project, my role involved developing the framework of the project, which encompassed various stages ranging from 'data collection' to 'preprocessing' to 'modelling' to 'distribution services'. Specifically, my contribution involved designing and implementing working Python scripts for each of the aforementioned stages, ensuring seamless integration between them and maximizing the efficiency of the project. The provision of highly detailed LST products is vital to various stakeholders, including urban planners, architects, and environmentalists. It offers unparalleled insight into the heat profiles of individual features, allowing for informed decision-making in the context of energy efficiency and urban heat islands.

Realization Process:
In the context of this internship project, I worked with European Space Agency's (ESA's) Sentinel satellite collection, with a focus on Sentinel-1 and Sentinel-3 satellites. The data from these satellites is freely available and provides a wealth of information that is highly relevant to the objectives of the internship project. While developing the preprocessing framework, I had the opportunity to work with snappy (SNAP python) module, which was developed by ESA. For the modelling aspect of the project, I worked with multiple regression models from the machine learning domain, which served as the backbone of the main algorithm. To ensure optimal performance and scalability, the entire framework was developed on the environment of AzureML.

Learning Process:
Prior to joining ITC, I was pursuing a bachelor's degree in Physics, which provided me with limited exposure to coding. However, after joining ITC, I have developed a keen interest in coding, and I enjoy building logical solutions to the presented problems with the help of coding. With this internship, I was fortunate enough to have the opportunity to put my acquired coding skills to the test. During my internship, I encountered components such as the snappy module and the environment of AzureML, which were unfamiliar to me. However, working with them improved my practical skills and provided me with valuable experience in coding in a professional environment, which differs significantly from coding in an academic setting. Automating the developed scripts was also a challenging aspect of the project, and it was a territory that was previously unknown to me. However, I was able to overcome this challenge and develop an automated workflow for processing large amounts of data and this is something I am really happy about.

In addition to gaining practical and domain specific skills, working on this project has also enhanced my time-management abilities. I have also had the unique experience of participating in daily team meetings, during which each member provides updates and overviews of their ongoing projects. This experience has improved my verbal communication skills, which I consider a valuable asset. However, some of the other meetings in which I participated at the municipality were conducted entirely in Dutch, which prevented me from fully engaging in those discussions. Nonetheless, this experience did provide me with an opportunity to improve my proficiency in the Dutch language, which I believe is a fortunate outcome.

Collaboration at a professional level was something that I was hoping to experience during my internship, however as this project was carried out individually, I did not have the opportunity to collaborate with other team members. Although this collaboration would have been the missing piece to the perfect picture, I am satisfied with the skills and experience gained through my internship project.

Communication Process:
During my internship, I was involved in several team meetings that were held on a daily and weekly basis. The daily meetings provided a space for the team members to update other team members about their respective ongoing projects and ask for assistance if faced with some issues. I also had weekly meetings with my supervisor, Dr. Milad Mahour, to evaluate the progress made and plan the next steps of the project. Additionally, I was part of the 'Sprint Review' and 'Sprint Planning' meetings ('sprint' refers to a period of three weeks). During the 'Sprint Review' meetings, interested team members present their progress on their respective projects to the other interested departments/teams of the municipality. I presented my work twice during these review meetings. 'Sprint Planning' meetings were held after the 'Sprint Review' meetings, where the team would plan the next steps for their respective projects. Moreover, towards the end of my internship, a one-hour long demo session was organized by our team to showcase my project, where I had the opportunity to present the project in detail. It was a pleasant experience to see people attending the session and showing interest in the project.

Overall Self-Assessment:
This internship has been of immense value to me, particularly in terms of getting practical experience working in a professional setting. Throughout the course of the internship, I was able to acquire and enhance my technical skills, as well as grow as an individual. I am grateful for the opportunity to have participated in such a rewarding experience.

Moreover, as a side note, the travel to the internship location via train instilled a new habit in me of reading physical books which I have grown to be quite fond of. Though not directly related to my academic pursuits, this habit has been beneficial in fostering intellectual growth about topics outside my feel of study. The internship, thus, not only contributed to my professional development, but also influenced my personal growth.

Recommendation:
I had a positive experience working at the Municipality of Rotterdam. It might be difficult at start to switch to their working environment like AzureML for writing codes; however, all the team members are kind and will be happy to assist you if you run into some issues. The onboarding process could have been better as the environment created for me to work in was not ready when I joined, so initially I had to work locally and then transfer everything to the assigned working environment.