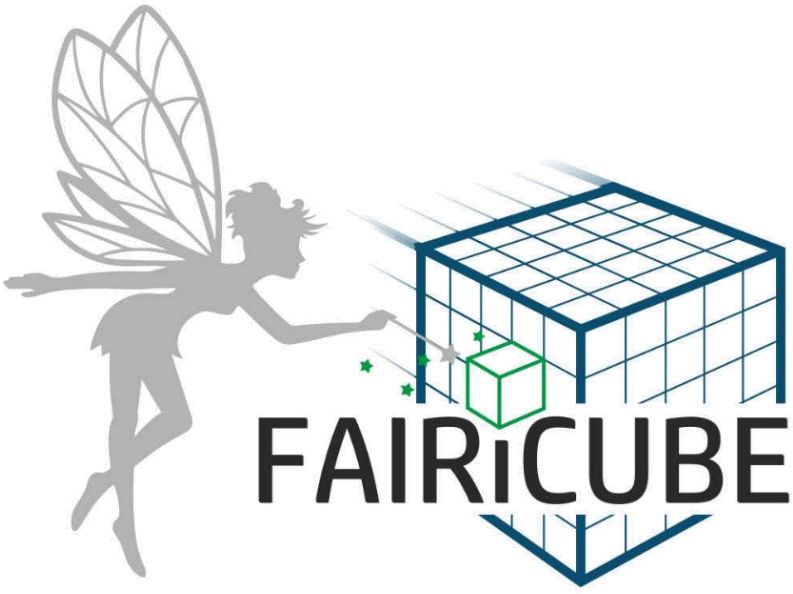

FAIRICUBE – F.A.I.R. INFORMATION CUBES

Project Number: 101059238

WP 3 Process

D3.2 Machine learning strategy specific for each use case

Deliverable Lead: NIL
Deliverable due date: 30/06/2024

Version: 2.4
13/05/2024

Document Control Page

Document Control Page	
Title	D3.2 Machine learning strategy specific for each use case
Creator	NIL
Description	D3.2 Machine learning strategy specific for each use case
Publisher	"FAIRICUBE – F.A.I.R. information cubes" Consortium
Contributors	NIL, WER, NHM, S4E
Date of delivery	30/06/2023
Type	R — Document, report
Language	EN-GB
Rights	Copyright "FAIRICUBE – F.A.I.R. information cubes"
Audience	☒ Public ☐ Confidential ☐ Classified
Status	☐ In Progress ☐ For Review ☒ For Approval ☐ Approved

Revision History			
Version	Date	Modified by	Comments
0.1	16/05/2023	Stefan Jetschny, NIL	Draft setup, headings and partner/contributor assignments
	27/05/2023	Rob Knapen, WER	Use case 2 contribution
0.2	11/06/2023	Stefan Jetschny	Ready for partial review, only the UC4 contribution missing
1.0	21/06/2023	Stefan Jetschny	Ready for review, minor comments still open
1.1	27/06/2023	Jaume Targa and Stefan Jetschny	Review and minor modifications according to review comments.
2.0	07/11/2023	Stefan Jetschny	Extraordinary update to reflect the progress of the work, read-through version for assigning writing tasks
2.1	26/01/2024	Jaume Targa	Initial review
2.2	07/02/2024	Jaume Targa	Final review
2.3	23/04/2024	Stefan Jetschny	Review with feedback from General Project Review Consolidated Report, UC5 chapter has been added as a placeholder for future progress
2.4	12/05/2024	Mirko Gregor (S4E)	Internal review

Disclaimer

This document is issued within the frame and for the purpose of the FAIRICUBE project. This project has received funding from the European Union's Horizon research and innovation programme under grant agreement No. 101059238. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Table of Contents

List of Figures

List of Tables

1 Introduction

WP3 aims to provide guidance, recommendations, technical expertise, and implementation support expertise to all use case efforts in terms of data analysis and processing. While the tasks will be executed by the use case developers, support will be given to assist in all data handling steps after ingestion and provision on both the Rasdaman- and EOxHub services as part of FAIRiCUBE's overall data and model services. Special emphasis is given to the data driven machine learning (ML) model generation.

This deliverable needs to be seen as one item of a classical and logical execution of a machine learning application. Given availability/ingestion of data, we first perform an exploratory data analysis to get familiar with the data, analyze statistical parameters and distribution, check for completeness, outliers and other characteristics which could be relevant for the choice of the machine learning. This in-depth data analysis is covered by the deliverable *D3.1 UC exploratory data analysis*.

Subsequently, the raw data might require conversion into features through a data engineering step. This could imply a combination of several input data sources or applying simple mathematical operations to enhance the meaningfulness of the raw data given the relationships that are to be revealed. The more prior information is available, the better the feature engineering process can be performed. Based on the findings from the exploratory data analysis, the formulation of the research question and the relationship between raw data sources/features, machine learning algorithms can be recommended to establish a baseline model if this is not provided by use case owners. Starting from the most efficient machine learning algorithm, more advanced ML methods can be identified to form a machine learning strategy. Several different methods might also be tested to recommend a method based on computational demands and the accuracy of the ML output. Typically, the testing of ML algorithms is performed on a subset of the original input data or on selected cases. The feature engineering process, testing of ML algorithms and the recommendation of a cascade to ML algorithms, as well as analyzing the output of ML methods is covered by this deliverable *D3.2 Machine learning strategy specific for each use case*.

As the FAIRiCUBE Hub ultimately wants to also provide resource estimations and guidance for ML applications, we want to collect and share computational parameters, timings, requirements and give an outlook on the expected scalability of the ML problems defined by the use cases. For each ML algorithm identified and executed as described in D3.2, we collect information on e.g., disk storage, CPU runtime, and main memory consumption, describe the hardware and environment where the ML algorithm is executed on and list essential libraries that are needed to exactly replicate the ML application. This technical documentation of the ML execution is covered in the deliverable *D3.3 Processing and ML applications*.

In summary, the exploratory data analysis (D3.1) can be seen as an essential input to the development of a UC specific machine learning strategy (D3.2) whereas the technical description in D3.3 acts as a reference to follow up on the execution and serves as valuable input to estimate the demands for other ML applications. In the following, the machine learning strategy is described for each use case.

2 Machine learning strategies

Developing a sound machine learning strategy can be considered as matching the findings from the data exploratory analysis, the formulation of the data analysis objective, the defined accuracy expectations from use case owners with the available & suitable machine learning algorithms. This can be a cascade of algorithms sorted after computational costs and/or transparency of the results. Usually, the more complex a ML algorithm is, the better the expected performance can be at a cost of higher computational efforts and less possibilities to debug/reproduce the exact numerical operations that lead to the output of the ML algorithm. Weighting available resources, the numerical efforts and the quality of the ML application by analysis of output metrics yields the most optimal ML algorithm that addresses the scientific problem that was formulated upfront.

In the following, we will provide details on the machine-learning strategies of all use cases (UCs). Note that UC5 started delayed and is not yet at the data processing stage/application of machine learning stage.

2.1 UC1 Urban adaptation to climate change

This use case aims to provide data to cities and other stakeholders (such as European institutions or city networks) to support the decision-making process. Some of the data need to be processed and analysed to make sense for decision makers (e.g., extract useful patterns instead of presenting the whole data). For example, the land use/land cover dataset (the five classes of level-1 i.e., *1. Artificial surfaces 2. Agricultural areas 3. Natural and semi-natural areas 4. Water 5. Wetlands*) can be processed to identify the cities that are similar w.r.t land use and hence provides a clearer picture to decision-makers.

Figure 1 presents the expected processing workflow in UC1. The flowchart describes the data flow and processing steps for the first part of the use case, namely the analysis of cities across the EU. In this part, a preparatory step is the calculation of descriptive indicators for a large number of cities across the EU and for several years. These indicators are partly derived from EU-wide spatial datasets (land-based and climate data) and partly are already available as socio-economic indicators. All the indicators are eventually collected into a data cube where the spatial coordinates are not the traditional geographic coordinates, but rather the city identifiers, which can in turn be linked back to geographic coordinates by using either the city boundaries or the city centre point coordinates.

There are three different types of data sources:

- Land-based data, mostly derived from the Copernicus Land Monitoring Service (CLMS).
- Climate data, mostly originating from the Copernicus Climate Change Service (C3S); and
- Socio-economic data from the Urban Audit database hosted by Eurostat.

Additional spatial datasets already publicly available in the EDC Hub will also be used.

The following processing steps are being implemented:

- Land and climate data are harmonised and ingested in a spatial data cube (spatial coordinates are lat/lon). Harmonisation consists for example in ensuring that the dimensions all share the same projection and geographical extent.
- The spatial data cube is used to compute various city indicators. One indicator has one value per city/timestamp.
- The spatial data cube will also be used in the subsequent spatially aware analysis (e.g. green distribution with cities) and for visualization purposes.
- These land- and climate-based indicators are then fed into a city data cube (technically implemented as a Postgres database for the time being). The city data cube differs from the spatial data cube because the spatial dimension (coordinates) are not the traditional geographic

coordinates (e.g. lat/lon) but rather the city identifiers. The spatial dimension in the traditional sense can be at any time recovered by linking the identifiers to the city boundaries or centre point coordinates. This data representation is less memory-intense and is more practical for further analysis.

- At the same time, socio-economic data is ingested. Socio-economic data do not have spatially explicit coordinates, but rather they are indexed by the city identifier. Therefore, indicators are directly fed into the city data cube, and derived indicators are computed.
- Attempts at ML-based gap filling have been made on the socioeconomic data but with limited success due to the large extent of the gaps. Rather the original dataset has been heavily filtered down to the indicators that have sufficient data.
- Cluster analysis is then carried out on the city data cube. The goal is to generate different clustering scenarios driven by different themes/questions (e.g. urban heat, flood retention, green infrastructure). This can be achieved by weighted clustering, where a weight is assigned to each feature (indicator) to control its influence on the clustering. An advantage of weighted clustering is also that it can be easily tuned to different needs, thereby generating multiple scenarios relatively quickly.
- Ultimately, this framework can be used to run simulations, by tuning the indicators values and measuring their influence on the outcome.
- The city data cube can be linked to visualisation tools to create dashboards and city fact sheets

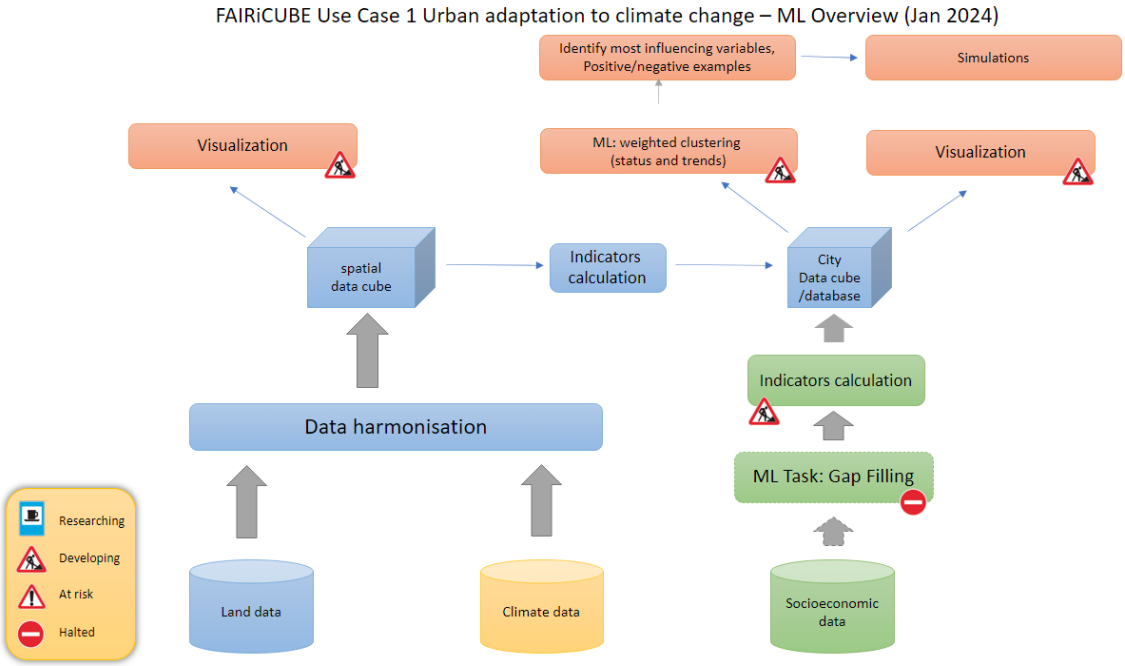

Figure 1 : UC1 data analysis and processing workflow

2.1.1 Clustering based on land use

To get a general overview of the cities that are similar with respect to area coverage, clustering can be used. Clustering is a type of Unsupervised Machine Learning tool that aims to put a given data into different clusters, each containing data points that are similar with respect to a set of features.

K-means is one of the most commonly used clustering algorithms due to its simplicity and efficiency in practice. Other algorithms include Mean-Shift, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation–Maximization Clustering and Agglomerative Hierarchical Clustering (AHC)

each useful on some type of data/problem. Additionally, Deep Learning can be used to reduce the number of features prior to implementing any clustering. This does not apply to this specific problem, because we only have 5 features (classes' ratios) as input. In addition, the features are already normalized, so there is no need for encoding/normalization steps.

For the task of clustering cities with respect to coverage ratios, we have experimented with three different clustering algorithms: k-means, Mean-Shift and AHC. Below, we will provide a report on the results obtained from each approach:

K-means is a clustering algorithm that starts with randomly selected K mean points and associates every data point to the closest mean. The means are incrementally updated until no significant change is noticed. A limitation of the k-means algorithm is the requirement to specify the number of clusters (K) as a parameter, which can be a drawback in some cases. However, this can be overcome using the Elbow Method. The elbow method runs k-means using multiple K values and selects the minimum K for which the inertia (squared sum of the distance between each point and its mean) starts to converge to a minimum value. For example, in this specific problem, a good choice of K would be 4, see Figure 1. Applying k-means with K = 4, has yielded four clusters of cities distributed as presented in Figure 2.

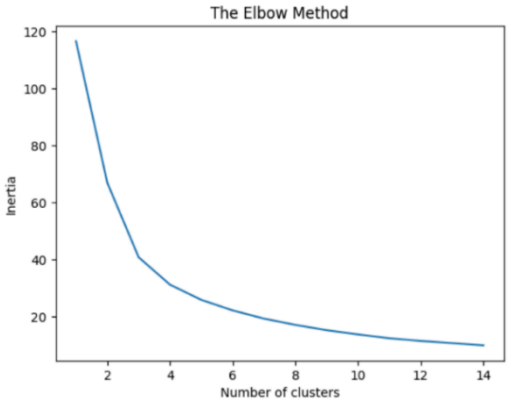

Figure 2 : Elbow metho to select k, the number of clusters.

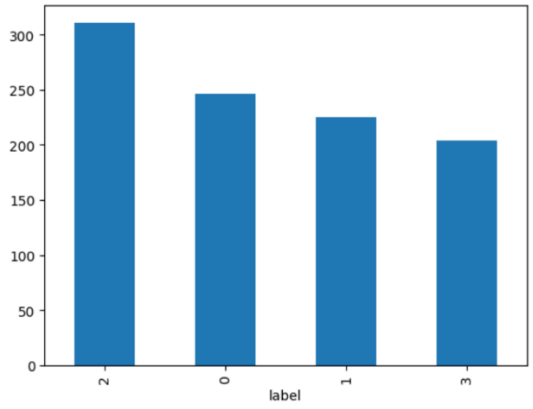

Figure 3 : Distribution of the four clusters using k-means.

Mean-Shift is based on shifting a window (with a pre-defined size) to the mean of the data points that are within the window. When it is no longer possible to shift the window (the centre of the window is the mean itself), the visited points are grouped into a single cluster and start constructing the next cluster by placing the window into a randomly selected data point. The main advantage of this algorithm is that there is no need for defining the number of clusters beforehand. However, we need to define the size of the window. In this study we tried different sizes and discovered that with size = 0.25, the number of clusters is acceptable (4 clusters). With a higher value, we get a lower number of clusters (a single cluster with size = 0.35). On the other hand, we get a higher number of clusters with smaller size value (9 clusters with size = 0.2). The distribution of the four clusters using Mean-Shift is presented in Figure 4. We clearly see that Mean-Shift puts most points (93% of the data) in a single cluster. This shows that this algorithm is not suitable for this specific study.

Agglomerative Hierarchical Clustering (AHC) builds a hierarchy of clusters by constructing a parent cluster from two closest sub-clusters (children). The Algorithm builds a hierarchy until every point is associated to a cluster. The user then can select any level of the hierarchy to be considered as clusters outcome. For this specific problem, we have built the hierarchy and selected 4 as the number of clusters (to be consistent with k-means). The distribution of the clusters is presented in Figure 5.

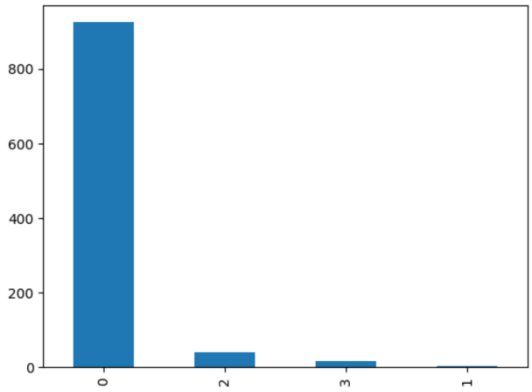

Figure 4 : Distribution of the four clusters using Mean-Shift.

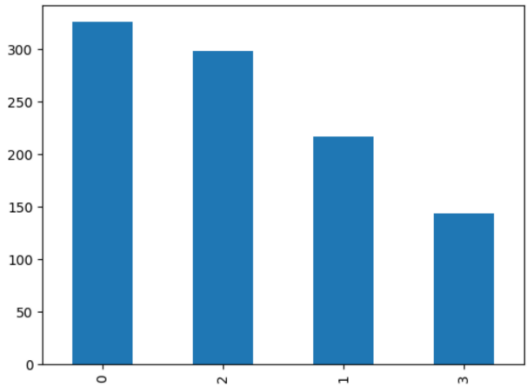

Figure 5 : Distribution of the four clusters using AHC.

In order to obtain an overview of each approach/cluster, we have cross-plotted the data points using each pair of features, as it is impossible to visualize a 5-dimensional clustering. We notice that very few cities have high water and wetland areas (see Figure 5), hence we consider the other three features (Artificial surfaces, Agricultural areas, and Natural and semi-natural areas) for plotting clusters (see Figure 6). The first observation is that Mean-Shift did not cluster the cities in an equitable way. It seems to group most of the cities into a single cluster, which is not meaningful in the context of this study.

On the other hand, k-means and AHC have yielded almost similar clustering. However, we can see some few overlapping using AHC (see Figure 6). Hence, based on the resulting clusters, the k-means algorithm seems to produce more meaningful clusters.
Following the plots in Figure 6, a possible significance of each cluster using k-means is the following:

- Cluster 0 represents cities with high artificial surfaces (e.g., Vienna, Austria).
- Cluster 1 represents cities with high natural and semi-natural areas (e.g., Stara Zagora, Bulgaria).
- Cluster 2 represents cities with a balance between artificial surfaces and agricultural areas (e.g., Linz, Austria).
- Cluster 3 represents cities with high agricultural areas (e.g., Rugby, Great Britain).

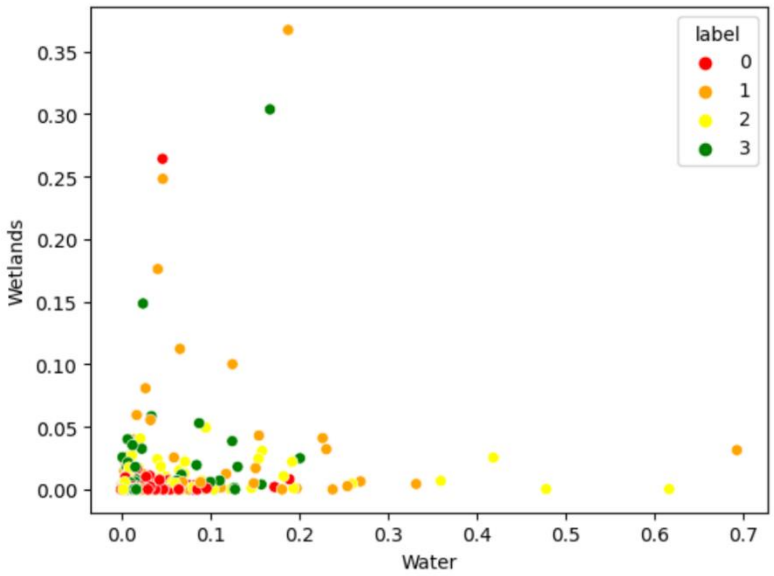

Figure 6 : Clustering with respect to Water and Wetlands

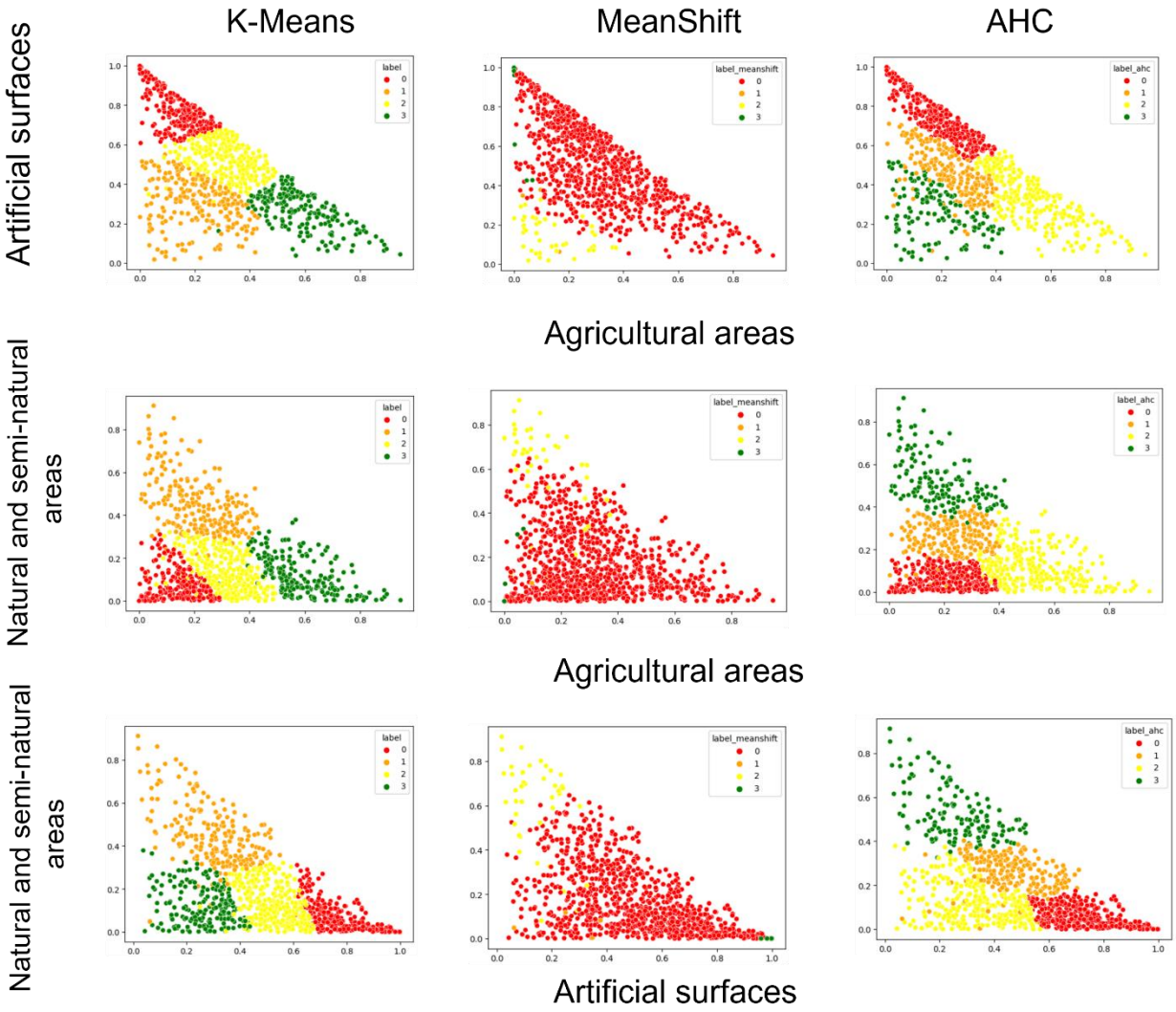

Figure 7 : Plotting clusters with respect to the three main features.

In this Machine Learning application, we have generated clusters of cities based on the input of five level-1 land use classes (*1. Artificial surfaces; 2. Agricultural areas; 3. Natural and semi-natural areas; 4. Water; 5. Wetlands*). We have used three different clustering algorithm (k-means, Mean-shift and AHC), and identified k-means as the baseline giving its consistent output (see clusters significance above). For further analysis, we will consider more features to have a multi-dimensional cluster of cities. For example, socio-economical features (e.g., population density), or climate features (e.g., heights temperature) can give more insight of cities from different perspectives.

2.1.2 Socioeconomic data gap filling

Obviously, clustering using only the land cover does not give a general insight into cities. Hence, using indicators from the socioeconomic data in addition will be useful. Unfortunately, the socioeconomic data reported by Eurostat is very limited and contains many gaps. In what follows, we propose several possible Machine Learning (ML) strategies to recover the missing data.

2.1.2.1 Using time series:

A possible direction into gap fills in missing indicator values of some specific years is to use the information of the other years i.e., the time series. For example, if the total population of a given city is increasing in the few last years, one can predict an increase -w.r.t some ratio- for the next year. Clearly, some time series are very complex and hardly predictable. One of the most known ML strategies for learning from time series is LSTM, for Long Short-Term Memory. LSTM is a type of Recurrent Neural Networks (RNNs) that learns from sequences of connected data and retain information over long sequences. Given a sequence size s, the LSTM model is trained to predict a value at time t using the previous s data points in the time series. Furthermore, **Bidirectional LSTM** (BLSTM) is an extension of LSTM, that learns both in forward and backward directions.

As an example, we have used a Bidirectional LSTM for 'Total population' prediction.

In the Eurostat data, the 'Total population' of some cities is available over 31 years (from 1991 to 2022). We selected two cities, one with simple linear increasing time series (Helsinki in Figure 8), and one with a more complex time series (Bari in Figure 9).

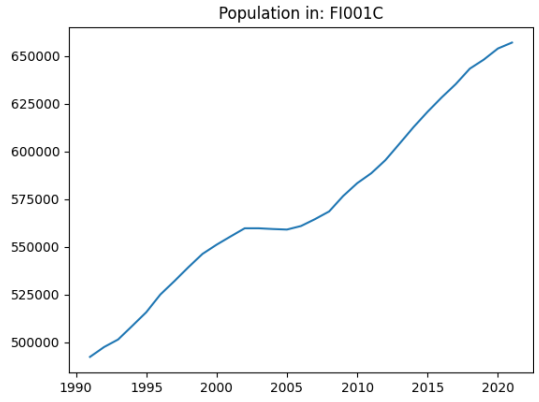

Figure 8 : Total population over the years in Helsinki

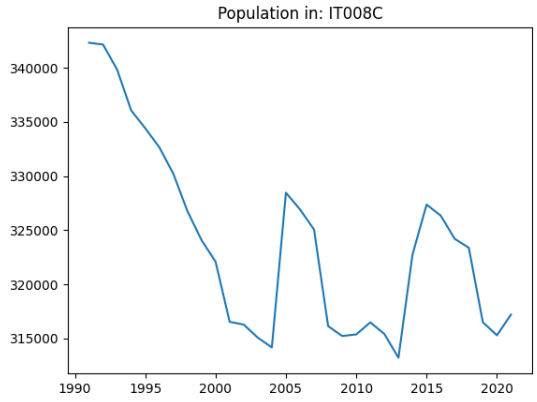

Figure 9 : Total population over the years in Bari

We have trained two BLSTMs, one for each city. We have selected the first 70% years for training, with sequence length of 3 (e.g., predict 2010 using 2009, 2008 and 2007), and the last years for testing.

After training over 100 epochs, the predicted time series vs the actual for Helsinki and Bari are presented in Figure 10 and Figure 11, respectively. We can clearly see that the prediction for Helsinki is close to the real values (with a Mean Absolute Percentage Error of 0.63%), this is thanks to the simple linear trend of the time series. On the other hand, for Bari, the prediction does not follow a linear pattern, but it is far from exactly following the actual trend even with a Mean Absolute Percentage Error of 0.53%.

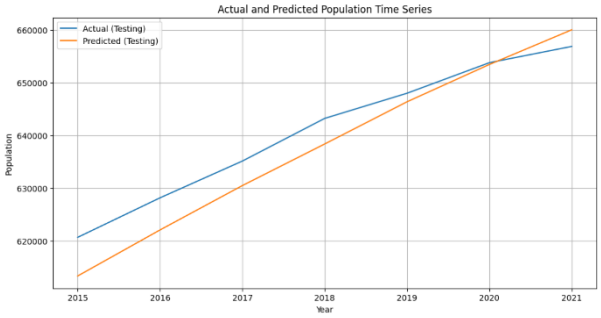

Figure 10 : Predicted vs Actual for Helsinki

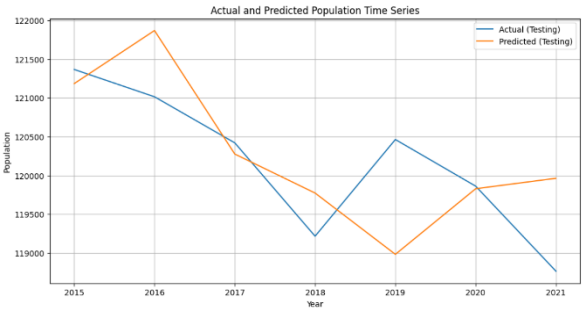

Figure 11 : Predicted vs Actual for Bari

Limitations: This approach is limited in our context. First, the number of data points is small for an effective training (only 31 data points) in case of complex sequences. To have acceptable results, a time series need to contain at least 50 data points for training.
Most importantly, we can notice that the time series are city/indicator specific. In this case, we need to train a model for each time series which is computationally expensive. Furthermore, no model can be useful for cities with 0 data points (unless we use a model of some similar city, which is hard to determine).
Finally, in Eurostat gap filling, this approach can be useful for cities/indicators with a few missing years (between 1 and 5), that follow regular trends (e.g., linear).

2.1.2.2 Using correlations between indicators:

Some indicators are related to each other. For instance, the *'Proportion of population aged 75 years and over'* is related to *'Total Population'*. Hence, we can use ML regression models to approximate the value of a given indicator using other available inputs. Regression models are approaches that learn a function to estimate the value of a given feature using a set of input features. Regression models range from simple linear regression (that only learns linear functions) to complex deep learning models.

As show case, we have implemented an ML model to estimate: the *'Total number of hours of sunshine per day [EN1002V]'* using two inputs: *'Average temperature of warmest month - degrees [EN1003V]'* and *'Average temperature of coldest month - degrees [EN1004V]'*. To train the model, we need data points that have the three features available i.e., 1,673 data points in Eurostat. Then the trained model can be used to gap-fill data points where the two input features are available, but not the target feature i.e., 299 data points in Eurostat. In Figure 12 we present the correlation matrix (computed using the 1,673 data points) between the three features. Clearly, a positive correlation exists between the target feature (number of sunshine hours) and the input features (temperature of warmest month, and temperature of coldest month). This is especially true for 'temperature of warmest month' where the correlation reaches 74%, this is promising for a regression model to be efficient.

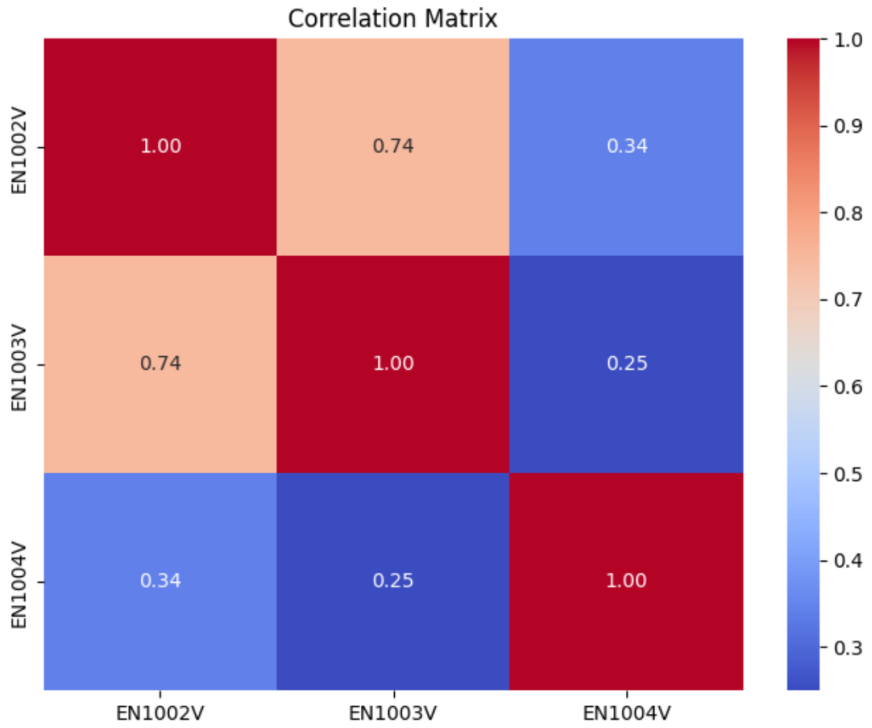

Figure 12 : Correlation matrix between 'Total number of hours of sunshine per day [EN1002V]', 'Average temperature of warmest month - degrees [EN1003V]' and 'Average temperature of coldest month - degrees [EN1004V]'

To predict EN1002V using EN1003V and EN1004V, we have trained several regression models on 70% of the selected data (the 1,673 data points). The **Gradient Boosting Regressor** (that uses an ensemble of decision trees for regression) was the most effective reaching mean squared error of 0.48 (error of less than 30 minutes of sunshine on average). Furthermore, the F1 score reached 0.677 indicating promising results. The distribution of the predicted vs the actual number of sunshine hours is presented in Figure 13. The distributions show close prediction to actual values.

Limitations: The results show that regression models can be effective in recovering missing data. The model above was useful to gap-fill in almost 300 data points. However, this approach requires designing regression models for each indicator and manually defining the potentially related input features. Furthermore, training data is not sufficient for each indicator. For example, it is not possible to predict the *'Proportion of households that are lone-pensioner households [DE3008I]'* using the input features: *'Lone pensioner (above retirement age) households [DE3008V]',* and *'Proportion of population aged 65-74 years [DE1028I]'* because 0 training data points are available (no data point that has all the three indicators available).

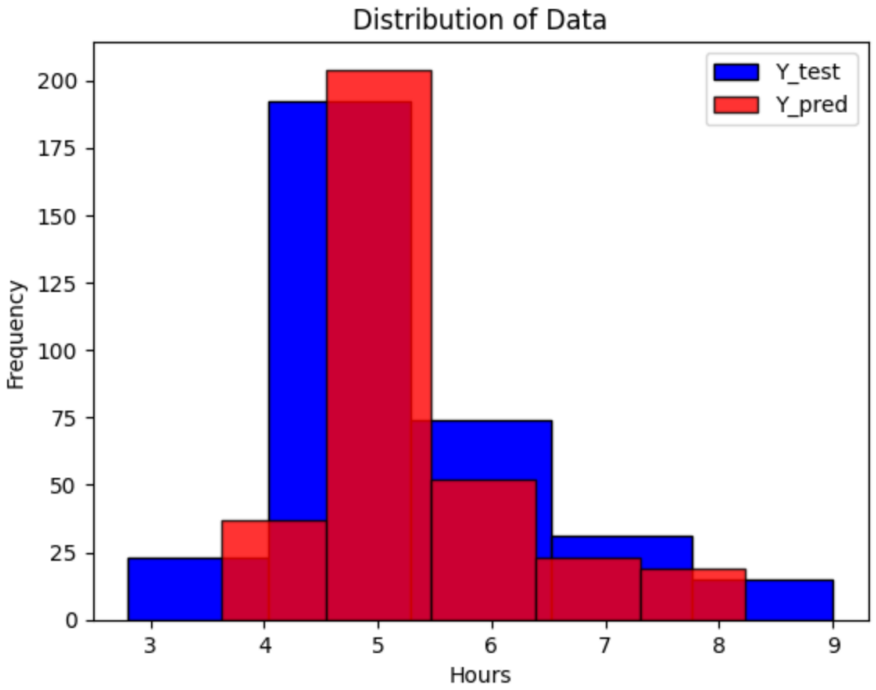

Figure 13 : Sunshine hours distribution (predicted vs actual)

2.1.2.3 A more general solution:

Above, we have shown the usefulness of both time series patterns and interdependence between indicators to gap-fill missing data. A more general solution is to take into consideration both the time and indicator dimensions in a single model to predict missing values. Furthermore, due to the absence of data for some cities, one may consider learning from cities with similar characteristics. In Figure 14 we present a potential ML architecture that considers all these information in a single model. Here, we first cluster cities w.r.t some pre-defined features, then we collect all available [indicators, years] matrices and feed it to a bidirectional LSTM architecture that takes into consideration the interdependence between indicators as well as the time series connections to predict values of a specific 'city cluster/indicator/year'. Implementing such a model was not possible due to the limitations mentioned below.

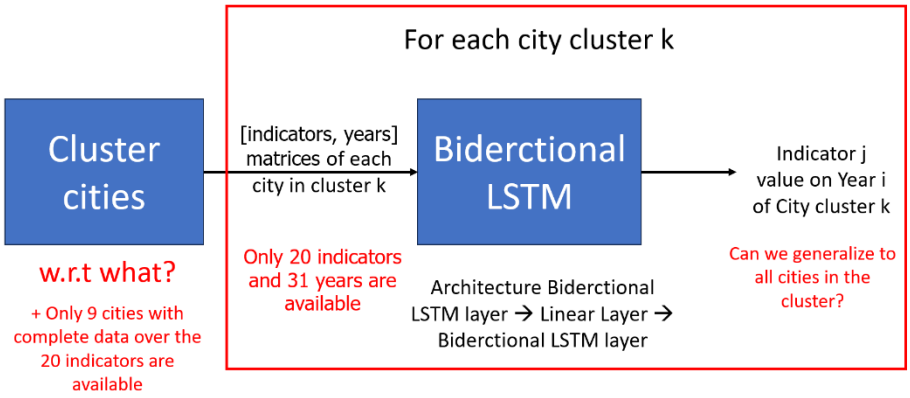

Figure 14 : Possible architecture for a general solution.

Limitations: Obviously, this model requires a large amount of training data that is not available in Eurostat. In Eurostat, we only have 9 cities that have 20 indicators with all available years. In addition, it is hard to pre-define features for which we should cluster cities at the first step, mostly, these features are the ones we want to gap fill. Finally, predicting values of a specific cluster of cities does not necessarily generalize to all cities in the same cluster.

Overall, gap-filling the Eurostat data is very challenging due to the absence of enough training data. Here, we have explored some low-hanging fruits (time series analysis, regression models). However, these strategies require some manual effort, and do not recover enough data. Finally, a more general solution requires more training data.

2.2 UC2 Agriculture and Biodiversity Nexus

2.2.1 General concept

This use case aims to study the effects of agriculture and farming activities on biodiversity, specifically in agricultural areas. As an overall guiding principle a detection and attribution approach will be followed, based on causal machine learning to relate (detected) changes in biodiversity to farm management practices (attribution), using (spatially detailed) data cubes as primary information sources.

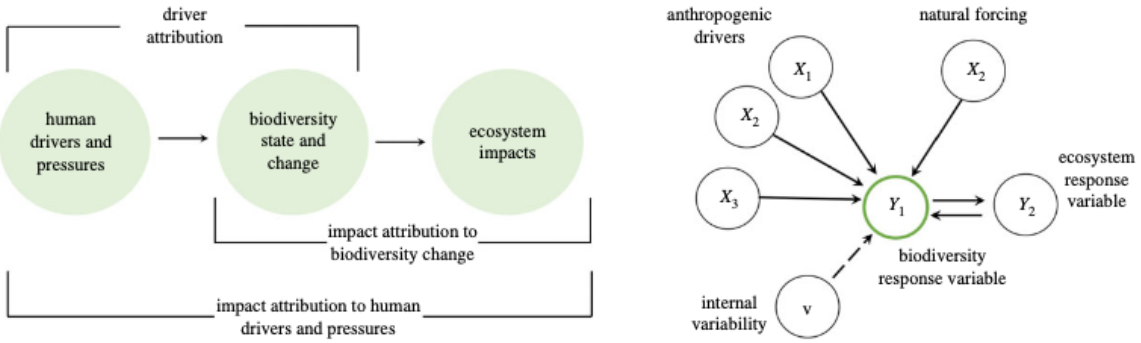

Figure 15 : Attributing biodiversity change to human drivers and pressure

Causal machine learning improves probabilistic approaches by adding causal reasoning, which for humans is a crucial element in learning about and understanding how the world works. Cause-and-effect relationships drive data, but regular statistical analysis alone is insufficient to recover those causal relationships from that data. The causality is part of the data generator, not of the data itself. In that sense, causal inference allows the discovery of the characteristics of the data generator. It is particularly

helpful in cases when fully Randomized Controlled Trials cannot be conducted, which clearly applies when working in the environmental / nature domain. Causal machine learning, including causal inference and causal discovery, can leverage provided causal graphs (e.g. for answering counterfactual ("what if?") questions) or learn causal graphs from observational data. It is a promising field of study in machine learning, but also still a very active research topic.

The concept of a framework for the detection and attribution of biodiversity change has been proposed by Gonzalez et al. (2023)[1]. Where possible for this use case we will try implement elements of this framework.

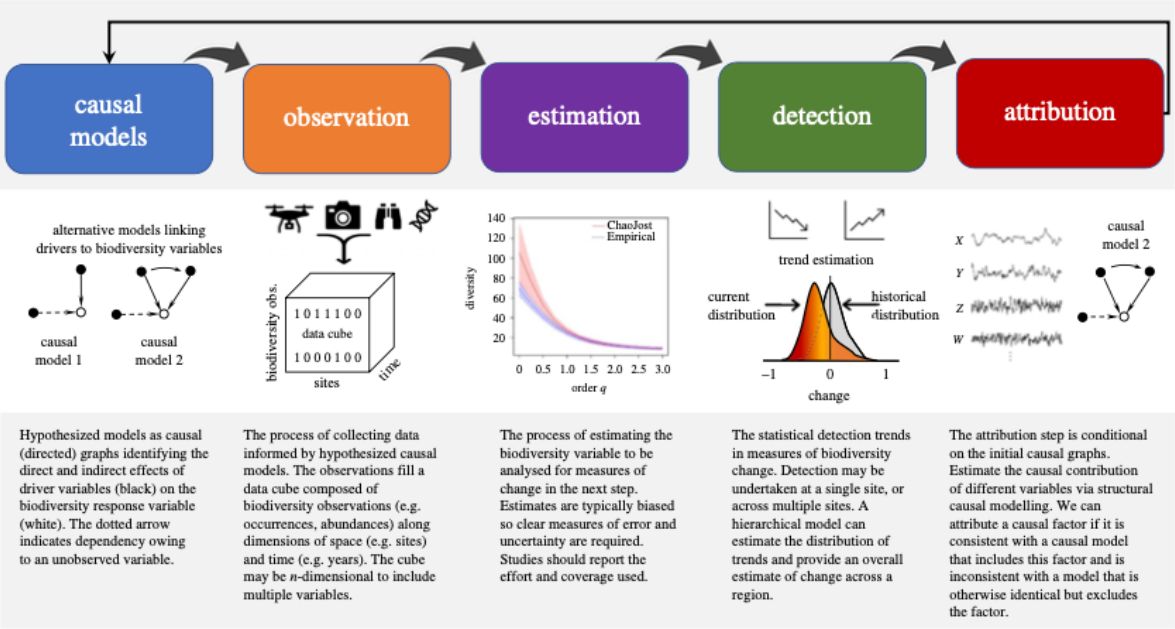

Figure 16 : Steps in the detection and attribution framework for biodiversity change

The causal machine learning approach in this use case is intended to identify the causal factors that contribute to changes in biodiversity within agricultural landscapes. This involves distinguishing between correlation and causation, which could lead to designing effective conservation and management strategies. Causal models facilitate counterfactual analysis, which (in this use case) involves comparing observed biodiversity outcomes with what would have happened in terms of biodiversity if a particular agricultural intervention or land management decision had, or had not, been implemented. This helps attribute changes in biodiversity to specific agricultural practices.

2.2.2 Conceptual ML and processing workflow

In the detection and attribution framework for biodiversity change, three main categories of data can be considered: biodiversity related data, environmental data, and agricultural data. Using these as pillars, a conceptual diagram of the proposed ML and processing workflow has been sketched for clarification and to serve as guidance for the various types of data engineering and ML tasks that might be needed (see Figure 17). It is worth noting that this workflow diagram might still change as the insights into the available data, data cube functionality, existing research (literature), and possible ML algorithms evolve. This is in line with the exploratory and experimental nature of data science and machine learning.

[1] Gonzalez A, Chase JM, O'Connor MI. 2023 A framework for the detection and attribution of biodiversity change. Phil. Trans. R. Soc. B 378: 20220182. https://doi.org/10.1098/rstb.2022.0182

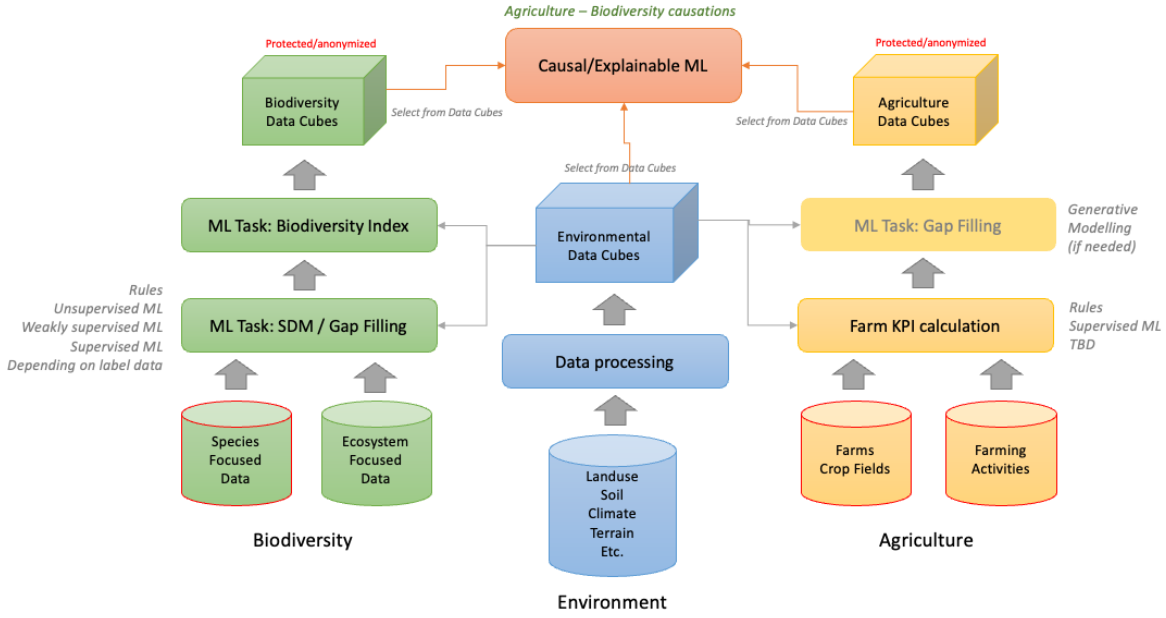

Figure 17 : UC2 data analysis and processing workflow

The diagram indicates the three pillars (biodiversity, environment, and agriculture) that play a role in the use case. The data engineering and ML tasks flow from the bottom of the diagram to the top, where they merge into the application of causal machine learning. By applying these types of ML algorithms, it is expected that not only associations in the data between agricultural activities and changes in biodiversity are detected, but also causality can be indicated and explained. For example, an increase in the presence of herb-rich grasslands causing an increased presence of a species under study. The datasets at the bottom marked with a red border note the inputs with additional requirements (such as restricted access and privacy sensitivity).

2.2.3 Status

In this section the workflow diagram is used to indicate the overall progress for the use case, as of November 2023. The traffic sign icons show the general state, work on items that are not marked by an icon still has to be started.

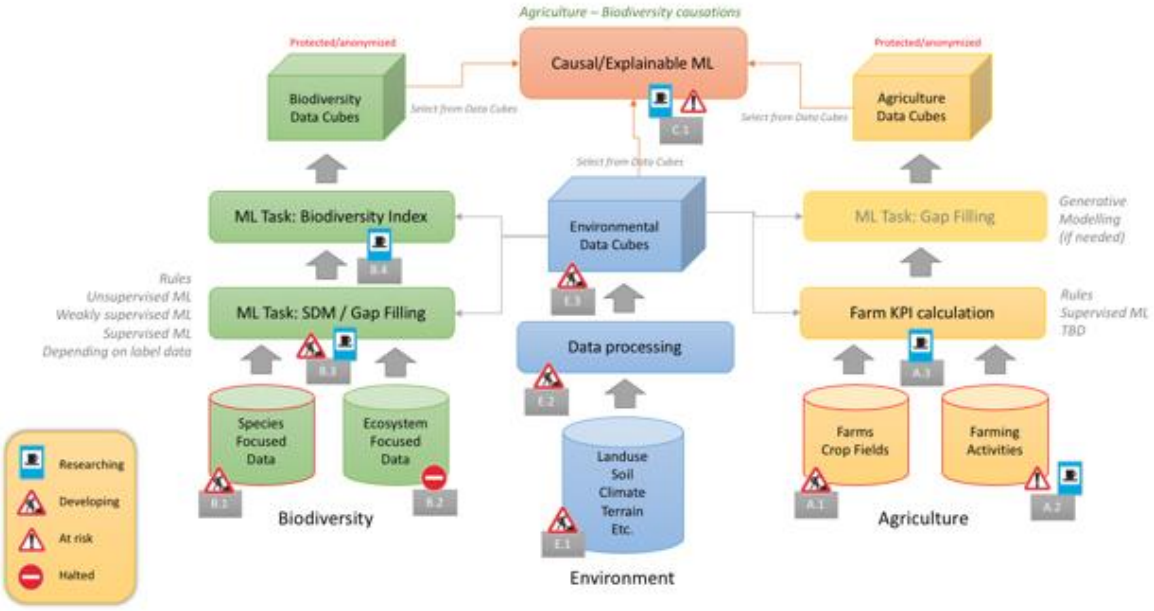

Figure 18 : Progress within UC2 data analysis and processing workflow

Tags associated with items and icons refer to further explanation, as follows:

Biodiversity (B)

- B.1: For species focused data the work on selecting species observations relevant to the use case is in progress, as well as the ingestion into rasdaman data cubes. Currently, two data sources are being used, the Dutch NDFF database (which requires a license to use the data), and the GBIF data (via our B-Cubed sister project). Both supply species observation data, NDFF as observations with geometries indicating ranges and GBIF as grid references for occupancy. Both need work to translate into spatial coverage-oriented data cubes.
- B.2: Due to project constraints the biodiversity aspect of the use case will be limited to species-focused data and leave ecosystem-focused data and variables for a possible follow-up as refinement.
- B.3: Given species observation data the work species distribution modelling (SDM) has started with some initial modelling using common Maximum Entropy (MaxEnt) and Random Forest type models. Since data is not fully ingested into data cubes and the de-facto Java-based MaxEnt model for SDM could not yet be run on the available processing environments, first experimenting will be done on local system configurations,
- B.4: Depending on the result of the species distribution modelling, a method still needs to be devised to calculate an index of biodiversity per spatial unit (i.e. grid cell of a chosen resolution). This can be one of the common biodiversity indices, such as the Shanon or Simpson index, but also a combination of a newly developed index (if needed).

Environment (E)

- E.1: Selection of needed environmental data is in progress. Some of this data is shared with other use cases. Every dataset requires discussion on how exactly the data needs to be represented in one or more data cubes. Such specifications must be provided by the use case to the data ingestion work package.
- E.2: Some of the environmental data needs processing before it is suitable for ingestion into a data cube, this work is in progress or will be started when needed. Since most of it is a one-time activity, no specific data pipelines are being build. Some of the work is performed as part of the use case work package using GIS software, depending on expertise and what is most efficient.

- E.3: Based on the specification provided by the use case and processed data (when applicable), rasdaman takes care of the ingestion of the data into their system and makes it available as data cubes.

Agriculture (A)

- A.1: In the Netherlands, as in many EU countries, farmers need to provide information about their parcels and the crops grown on them every year. Part of this data is publicly available. However, the ownership information of all parcels and data about farming activities in general is not. Also, the available crop parcel data is in vector (polygon) format and thus needs to be gridded with a usable spatial resolution first before it can be ingested into a data cube. Work on this is in progress.
- A.2: Detailed (per-farm) data about farming activities cannot easily be obtained, as it turned out and despite ongoing efforts. It basically requires data-sharing consent by each farmer involved, which is too impractical for this project. An alternative approach will have to be researched and might involve working with anonymized farm private data or synthesizing characteristic farm-type data that would still allow to showcase of the full detection and attributed framework approach the use case aims for.
- A.3: Based on the availability of crop parcel and farming activity data it needs to be investigated which Key Performance Indicators are sensible to derive and use. For example, the Hill-Shannon Diversity (indicating crop diversification) and Edge Density (indicating the spatial configuration of parcels). Values for these indicators then will have to be calculated in gridded (cell-based) format and made available for ingestion into data cubes by rasdaman.

Causal / Explainable Machine Learning (C)

- C.1: The ultimate step of the use case relates to the attribution part of the framework, in which biodiversity changes are causally related to agricultural activities if such a relation can be found. It still needs active research on ML toolkits and technologies that can be applied, but also requires the availability of the envisioned biodiversity, agriculture, and environmental data cubes. The delays in preparing those data cubes put this final activity at risk. It might not be possible to fully achieve it within the scope of the project.

2.2.4 Biodiversity Pillar Data Engineering and Machine Learning

Despite decades of research in biodiversity science, there still exists an information gap in biodiversity science, and it is a major obstacle for reducing the large uncertainties associated with answering those questions. Scientists, public administrations, and environmental organizations adopt a wide variety of data collection and monitoring protocols, while coordination is insufficient and multiple biases (including spatial, temporal, and taxonomic) are present in the data. To help resolve some of such issues, in 2013 a framework of *Essential Biodiversity Variables (EBVs)* was proposed[1]. Since then, not only has the acceptance of this concept grown, but research has also been published on the integration of in-situ observations and remote sensing data through modelling for the purpose of EBVs. In use case 2 we therefore have chosen to focus our biodiversity data related work around these variables as well. Figure 19 shows an overview of the six broad classes that the EBVs are grouped in[2] (note that the use case will be limited to only considering some of the species focused EBV classes).

[1] Pereira, H. M., Ferrier, S., Walters, M., Geller, G. N., Jongman, R. H., Scholes, R. J., Bruford, M. W., Brummitt, N., Butchart, S. H., Cardoso, A. C., Coops, N. C., Dulloo, E., Faith, D. P., Freyhof, J., Gregory, R. D., Heip, C., Höft, R., Hurtt, G., Jetz, W., . . . Wegmann, M. (2013). Ecology. Essential biodiversity variables. *Science*, *339*(6117), 277-278. https://doi.org/10.1126/science.1229931

[2] Fernández, N., Ferrier, S., Navarro, L. M., & Pereira, H. M. (2020). Essential Biodiversity Variables: Integrating In-Situ Observations and Remote Sensing Through Modeling. In *Remote Sensing of Plant Biodiversity* (pp. 485-501). Springer International Publishing. https://doi.org/10.1007/978-3-030-33157-3_18

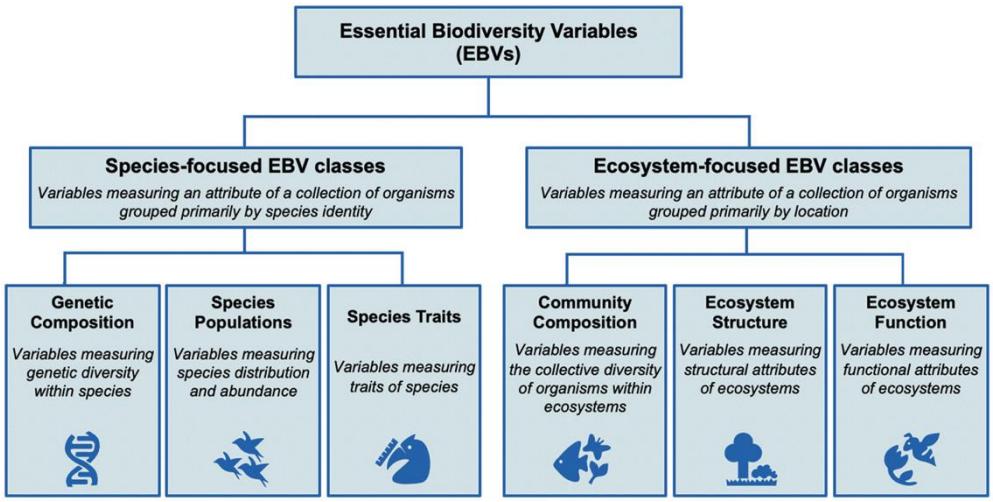

Figure 19 : Essential Biodiversity Variables

Fittingly, the data structure of an EBV is thought of as a space-time-biology hypercube, which should be a good fit for the multidimensional array data cubes used in this project. For this however we need the biology dimension to describe taxonomy and values to inform about presence/absence or population abundance. A key issue then becomes how to transform the in-situ observation data into 'gridded' data that matches the available data cube technology in the FAIRiCUBE project, in case of UC2 the rasdaman implementation of data cubes. The two foreseen steps in this transformation are: (1) *primary data aggregation*; and (2) *model-based estimation (gap filling)*.

2.2.4.1 Primary data aggregation

Following the EBV production workflow described in literature the steps to take for the aggregation of the primary data, i.e. the species distribution data acquired from the Dutch NDFF ("Nationale Databank Flora en Fauna") and from GBIF, include: (1) *data harmonization*, (2) *data aggregation*, (3) *uncertainty estimation*, and (4) *metadata annotation*.

In this case the required *data harmonization* is minimal, since it is part of the work already performed by the NDFF and GBIF organizations. Sufficient care must be taken though when selecting and filtering the species distribution data from both databases. Initially, we only had three NDFF datasets from 2016 to work with (nesting birds, other species of interest, and plants). In the second phase we got available extended dataset covering the same spatial extent of study area but including all available species records for the period of years 2014 – 2022 as described in deliverable D3.1.

The NDFF nesting birds dataset has been selected for an initial *data aggregation* study (note that GBIF can provide species occurrence data already aggregated into grid cells of choice). For the moment the aggregation procedure is being developed in a Jupyter Notebook, which is available on the project's GitHub site. The algorithm is as follows (for more details please see the notebook):

i. Load the CSV file with the species distribution data into a *Pandas DataFrame*.
ii. Perform initial clean-up and prepare the WKT (geometry) data to match *GeoPandas* requirements.
iii. Lift the Pandas *DataFrame* into a *GeoDataFrame* (**A**) (which supports spatial operations).
iv. Select only the observations inside the study area of the use case.
v. Extend the observation octagons by spatial buffering with the radius.
vi. Load a prepared grid (e.g. with 100m x 100m grid cells) into another *GeoDataFrame* (**B**).
vii. Calculate the spatial overlap (by union) (**C**) between **A** and **B**.
viii. In **C**, calculate the species abundance proportion for the overlapping geometries (by area).
ix. Merge (using a spatial join) the grid **B** with **C**, by covered geometries, creating dataset **D**.
x. Dissolve (aggregate) **D** by either counting the observations or summing the abundance shares.
xi. Assign the aggregated values to the cells of grid **B**, giving the final output.

Figure 20 shows a combination of multiple datasets created during the aggregation. The blue dots and hatched circles (octagons) are the individual observations, with the GPS accuracy octagon extended by the recorded radius. These get spatially intersected with the grid cells (see Figure 21), after which the intersections are used to calculate the shares of the abundance to assign to the cells. All shares are then summed per cell to get the final aggregated value. In the background the agricultural fields in the area are shown. The empty (not coloured) cells indicate 'unknown' abundance, since no species absence data is included in the processing.

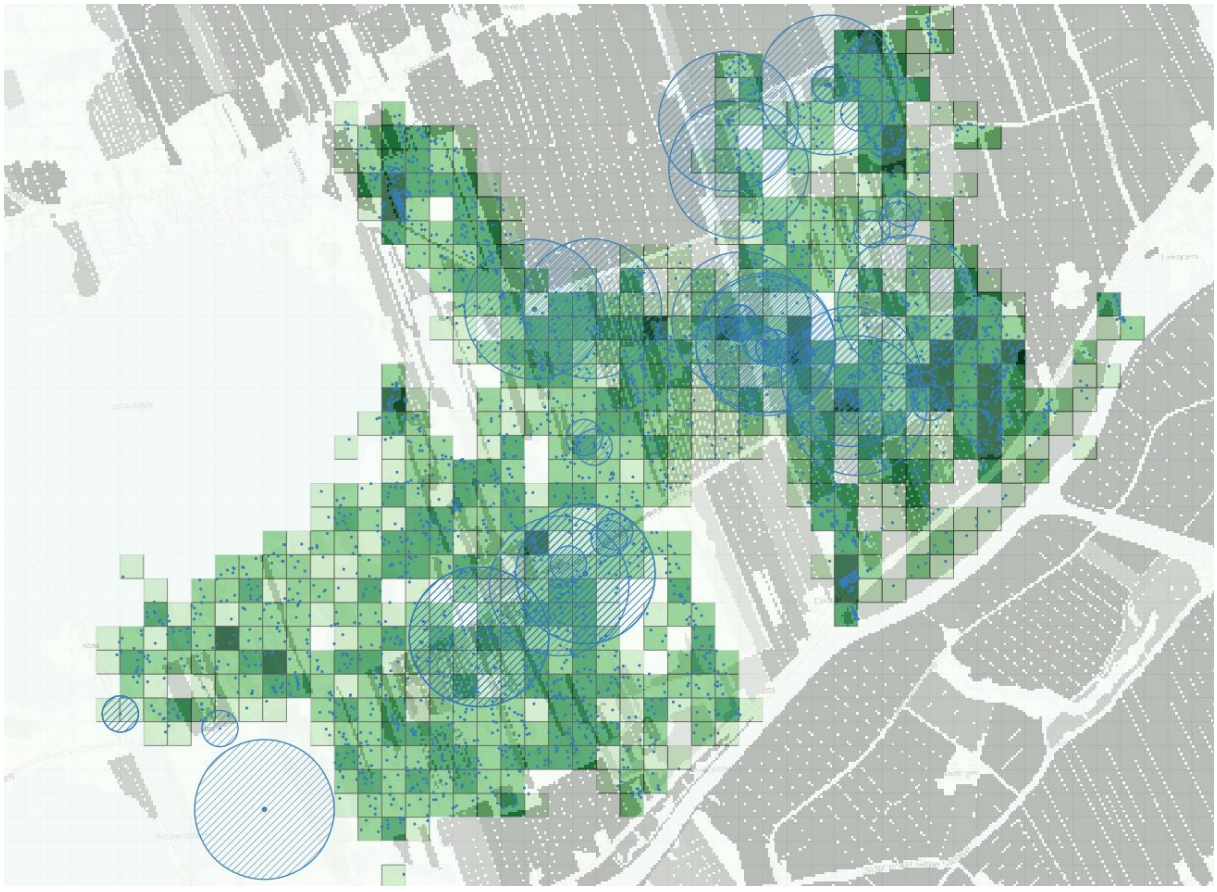

Figure 20 : A view of multiple datasets from the aggregation process

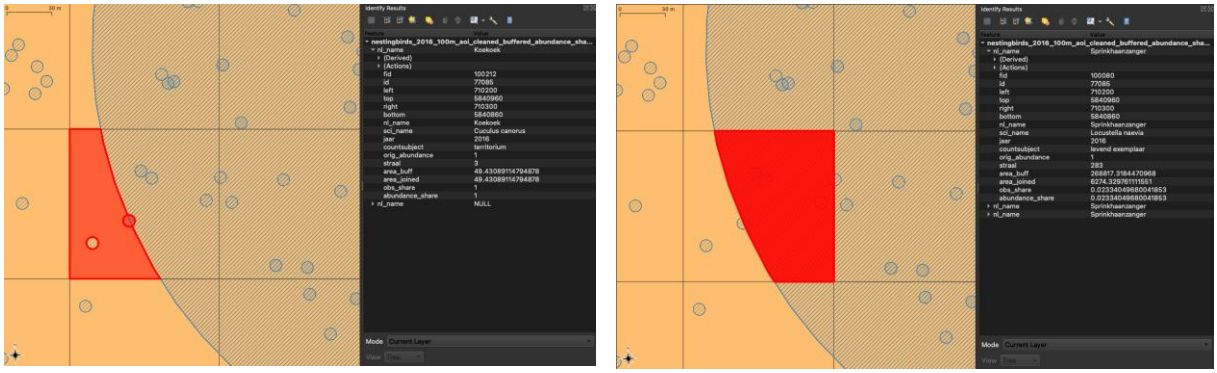

Figure 21 : Examples of intersections generated during processing

In the second phase of data aggregation effort, we focused on breeding bird species of year 2018 and specifically on selection of Farmland Bird species (please see further explanation in section 2.2.4.2 of this report). Outcome of this aggregation procedure serves as an input to species distribution models (section 2.2.4.2). Following steps were taken to reach aggregated data layer:

I. Load the Geopackage file with the species distribution data into a GeoPandas DataFrame (A).
II. Select only the observations of birds.
III. Select only the observations from year 2018.
IV. Select only the observations inside the study area of the use case.
V. Select only the observations of Farmland Birds and breeding birds.
VI. Create GeoDataFrame for each farmland bird species.
VII. Load a prepared grid (e.g. with 100m x 100m grid cells) into another GeoDataFrame (B).
VIII. Calculate the spatial overlap (by union) (C) between A and B for each farmland bird species.
IX. In C, calculate the species abundance proportion for the overlapping geometries (by area) for each farmland bird species.
X. Merge (using a spatial join) the grid B with C, by covered geometries, creating dataset D.
XI. Dissolve (aggregate) D by either counting the observations or summing the abundance shares.
XII. Assign the aggregated values to the cells of grid B and remove records with 0 value.
XIII. Create point dataset as centroids of grid cells of B and copy attribute values from b, giving the final output.

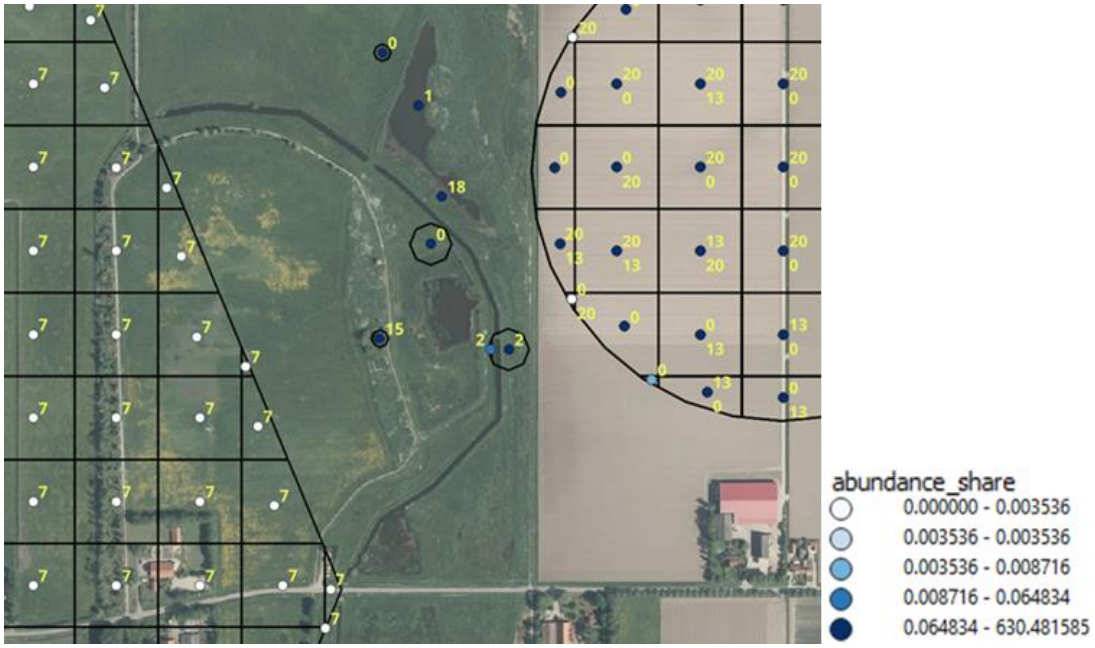

Figure 22 : Example of grid with abundance share and points generated per individual species

The approach described above, needs some expert input to judge its validity, how it can be improved, and how eventually the *uncertainty estimation* can be done. When available, we might use existing species suitability maps for that, although they typically are not more detailed than 1km grid cells.

This work is shared with similar ongoing developments in use case 5 for the validation of phyto-sociological (describing and classifying species composition of plant communities) methods through FAIRiCUBE Occurrence Cubes, which are split into *species cubes* and *plant community cubes*, that have common characteristics (such as the latitude, longitude, time, species (groups) dimensions). The 'cubing' approach can be largely the same as well, though with slight differences to cover both plants and animals. The concluding step of *metadata annotation* is expected to be part of the FAIRiCUBE data ingestion process where the result of the processing will be loaded into a rasdaman data cube.

It is worth to note that the spatial processing steps required in this algorithm can get time consuming. The species distribution data has many records, and the choice (need) to work with small grid cells (e.g. 10 – 100m) results in many (potential) spatial intersections that have to be checked and geometrically calculated. When processing times get too long (even when using spatial indexing such as R-trees or

similar), use of distributed processing can help. Possible options are *Dask*[1] or *PySpark*[2]. However, for both, it is recommended to check first if all needed spatial operations are supported.

2.2.4.2 Model-based estimation

After the primary data has been aggregated, the next step in creating biodiversity variables is usually some kind of model-based estimation of species distributions (species distribution modelling, SDM). SDM is a methodology – a set of procedures, definitions, and techniques about the relationship between species distributions (or other biotic response variables describing aspects of biodiversity) and the physical (abiotic) environment. SDMs are quantitative, empirical models of species–environment relationships typically developed using species location data (abundance, occurrence) and those environmental variables thought to influence species distributions.[3] One of the most popular tools for SDM are *MaxEnt*[4] (maximum entropy) models, that use presence-only occurrence data and relevant environmental variables such as remote sensing derived vegetation indices, land use, digital elevation model and climate (e.g. Worldclim[5] is a popular resource for global climate and weather data) to create (probabilistic) habitat (suitability) maps. MaxEnt models predict species occurrence by finding the distribution that is most spread out, or closest to uniform (i.e. having maximum entropy), while considering the limits of the environmental variables of known locations. In general, MaxEnt is a machine learning approach that can be used to estimate the probability of presence of a phenomenon, from known occurrence points and explanatory variables. The trained model can then be used to predict presence in different data if the corresponding explanatory variables are known. However, while MaxEnt models are generally very robust for presence-only data, they do (need to) make a lot of assumptions. When presence-absence data is available other types of models or ensemble modelling might give better results. And, while MaxEnt and Random Forest are established ML approaches for SDM, the recent advances made in deep learning, including generative modelling (and availability of compute resources) has made these interesting alternative approaches[6]. They can, for example, also learn to take species co-occurrence into account[7].

At the time of writing the focus is shifting from the selection, acquisition, and aggregation of the primary data to the follow-up steps of: (1) *Identification of confounders* (the environmental variables) from existing data cubes or ingesting new data when needed; (2) *Model learning and evaluation* using a few selected approaches (MaxEnt modelling results might already exist but perhaps only at a coarser resolution); (3) *Uncertainty estimation* of (the best possible) model; and (4) *metadata annotation*, which should again be covered by the FAIRiCUBE data ingestion process.

A first set of experiments is currently being run using the MaxEnt[8] model on observation data for 21 selected farmland bird species in our study region. These selected species are commonly used to calculate a 'Farmland Bird Indicator', that can serve as a proxy for assessing the biodiversity status of agricultural landscapes in Europe[9]. It includes birds that are dependent on farmland for feeding and nesting and are not able to thrive in other habitats. For these species the observation data from NDFF for 2018 is used.

As environmental variables the following datasets are used (all from 2018):

- LGN: Land use classes of the Netherlands, grouped into monitoring classes.
- OHN: Object Heights (in the Netherlands) on the land surface, derived from the AHN elevation dataset.

[1] https://www.dask.org
[2] https://spark.apache.org/docs/latest/api/python/
[3] Elith, J., & Franklin, J. (2013). Species Distribution Modeling. Encyclopedia of Biodiversity: Second Edition, 692–705. https://doi.org/10.1016/B978-0-12-384719-5.00318-X
[4] Steven J. Phillips, Robert P. Anderson, Robert E. Schapire. 2006. Maximum entropy modeling of species geographic distributions. Ecological Modelling, 190:231-259, 2006.
[5] https://worldclim.org
[6] Estopinan, J., Servajean, M., Bonnet, P., Munoz, F., & Joly, A. (2022). Deep Species Distribution Modeling From Sentinel-2 Image Time-Series: A Global Scale Analysis on the Orchid Family. *Front Plant Sci*, *13*, 839327. https://doi.org/10.3389/fpls.2022.839327
[7] Rademaker, M., Hogeweg, L., & Vos, R. (2019). Modelling the niches of wild and domesticated Ungulate species using deep learning. *bioRxiv*. https://doi.org/10.1101/744441
[8] Steven J. Phillips, Miroslav Dudík, Robert E. Schapire. [Internet] Maxent software for modeling species niches and distributions (Version 3.4.1). Available from url: http://biodiversityinformatics.amnh.org/open_source/maxent/.
[9] https://agridata.ec.europa.eu/Qlik_Downloads/InfoSheetEnvironmental/infoC35.html#_ftn1

- NDVI: Normalized Difference Vegetation Index, 4 seasonal images (February, May, August, September) derived from Sentinel 2 data.

Due to the used MaxEnt model software requiring a Java runtime, for which installation is under discussion at EOX, all processing has been done locally. In total 32306 species observation points derived from observation records have been used (0.8 MiB input file), and each species has been modelled individually (10 cross validation runs and 500 iterations of MaxEnt, taking about 17 hours on a M2 MacBook Pro with 16 GiB memory).To evaluate the performance of the model on the classification task per species the ROC (Receiver Operating Characteristics) curve can be used. It shows the true positive rate against the false positive rate at classification thresholds. The closer the curve is to the top left corner the better the model performs. The AUC (Area Under the Curve) indicates how well the model can distinguish between classes (Figure 21,22).

In this study the (10 run) average AUC value per species varied between 0.71 and 0.93, with an outlier of 0.56 which can probably be related to faulty/noisy input data. The relative contribution of each of the selected environmental variables also differs per species and needs to be further analysed. OHN always seems to be one of the least contributing variables. Below are examples of results of two species Gallinago gallinago (Common snipe) and Carduelis carduelis (European goldfinch) as obtained from MaxEnt.

MaxEnt results for Common Snipe

Species: Gallinago gallinago (Common snipe)
https://en.wikipedia.org/wiki/Common_snipe

Mean AUC: 0.93
AUC stddev: 0.02
Training samples: 98
Test samples: 11
Datapoints: 10098

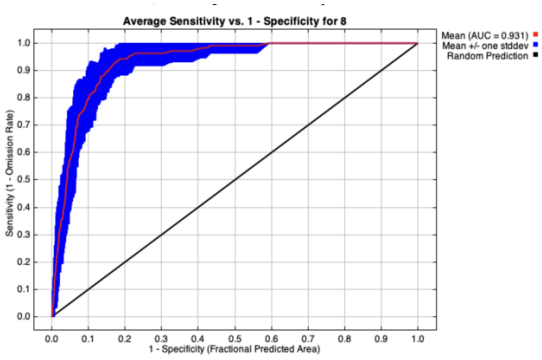

Figure 23 : 'Snipe' ROC and mean AUC

MaxEnt results for European Goldfinch

Species: Carduelis carduelis (European goldfinch)
https://en.wikipedia.org/wiki/European_goldfinch

Mean AUC: 0.79
AUC stddev: 0.03
Training samples: 418
Test samples: 47
Datapoints: 10418

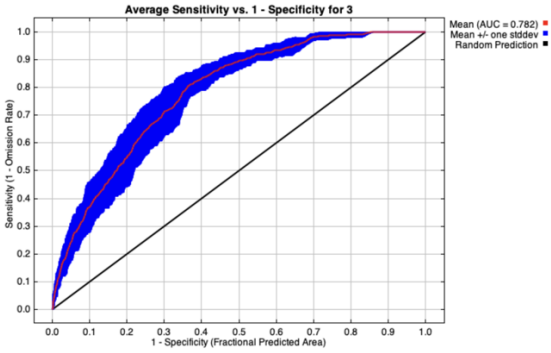

Figure 24 : 'Goldfinch' ROC and mean AUC

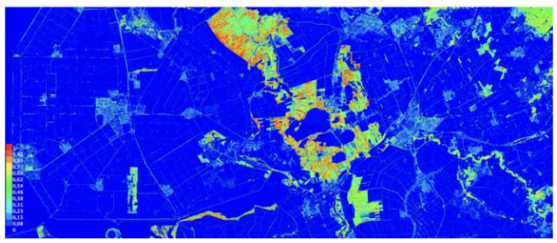

Figure 25 : 'Snipe' occurrence probabilities

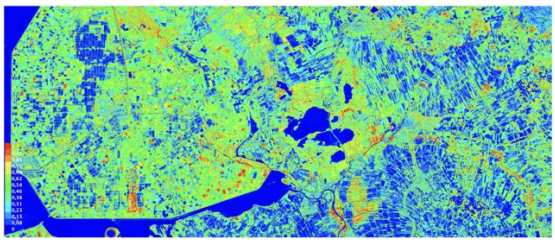

Figure 26 : 'Goldfinch' occurrence probabilities

Variable contribution (and permutation importance):
LGN: 69.3% [74.7%]
NDVI August: 10.2% [9.2%]
NDVI February: 8.8% [8.3%]
NDVI May: 8.6% [5.1%]
OHN: 1.7% [1.6%]
NDVI September: 1.4% [1.2%]

Variable contribution (and permutation importance):
NDVI February: 57.3% [40.3%]
NDVI May: 19.8% [26.2%]
NDVI September: 9.4% [15.1%]
NDVI August: 6.8% [15.1%]
LGN: 5.9% [6.6%]
OHN: 0.8% [1.2%]

Based on these initial study results we plan to further experiment with the MaxEnt model, e.g. by using other environmental variables (confounders) and looking at known characteristics of the bird species abundance or comparing the results with for example expert-based species distribution maps. Besides that, we intend to explore other machine learning models such as random forest and, if possible, run them on FAIRiCUBE Hub instead of locally.

2.2.5 Agricultural Pillar Data Engineering and Machine Learning

Regarding the agriculture aspect of the use case the intention was to make use of data from existing initiatives to reward farmers for more nature friendly agricultural management. The most challenging aspect of this data as it turns out is that it is privacy sensitive and not easily shared by farmers since it directly involves their business operation. Even when permission to use the data is given, proper pseudonymization / anonymization must be considered. Including any accidentally 'leaking' of such privacy sensitive data, e.g. by overfitted ML models from which data points can then still be extracted.

While attempts were made to obtain (data usage) consents from a sufficient large number of farmers, among others by cooperating with internal projects that also needed them, in practice this has turned out to be too time-consuming and thus impractical for this project. As an alternative we intent to look at synthesizing usable sample data representing realistic (but fictious) farms. For example, by using a farm typology (average arable, dairy, and mixed farms in the Netherlands) and available regional statistical agricultural data. Further details of this are still being explored.

Information about nature friendly farm activities can possibly still be derived from the internal dataset called ANLb ("Agrarisch Natuur- en Landschapsbeheer"), which records field management practices that are more nature friendly and improve sustainability. Other input can come from current studies that are establishing and collecting Key Performance Indictors (KPIs) for farms, both arable and livestock (mostly dairy farms now). While still in development and most KPIs also requiring farm specific and sensitive data, some can be derived from open data (however the link between crop parcels and farms will not be known). The KPIs related to biodiversity that are currently considered are the (1) *Hill-Shannon Diversity* (in this case measuring the crop diversification), and the (2) *Edge Density (m/ha)* which indicates how many field borders there are (fewer large fields are assumed to be better for biodiversity).

All currently known KPI calculations are formula/decision rule based, therefore no need for machine learning is expected. However, it might turn out that available data points are too sparse (e.g. not many farmers agreeing to use of their data for the project), in which case weakly supervised learning or generative modelling can be considered to improve and partially solve the data sparsity.

A further crucial aspect will be the way the data, which typically is in vector (geometry) format (e.g. parcel boundaries), is transformed ('gridded') into raster data. For the Dutch parcel registration data ('basisregistratie percelen') currently a spatial resolution of 10m x 10m is considered, which needs to be carefully aligned with the other types of data in use, as will also be described in the following section.

2.2.6 Environmental Pillar Data Engineering and Machine Learning

This pillar concerns the more 'classical' Earth Observation data storage and processing, which should be most native to the available FAIRiCUBE platform. No specific need for machine learning is expected now. Rasdaman offers extensive data processing functionality via its query language (*rasql*), and if needed it can be extended with *User Defined Functions* (UDFs).

Most interesting (and challenging) will be that the environmental data usually is available as raster data, while the other two kinds of data used (biodiversity and agriculture) are vector data (or 'plain' tabular data with columns for coordinates or a location indication). These need to be matched by choosing an adequate grid (including cell size), one (or more) suitable coordinate reference systems (CRS), and proper transformations. Specifically for machine learning, mis-aligning grid cells or deforming the data too much (e.g. by accumulating transformation errors) can easily lead to a garbage-in, garbage-out situation, that might go undetected or take a long time to figure out. For deep learning a possible way to counter can be to focus on models that generalize better, for example by including (or simulating) coordinate transformations as part of the regular data augmentations.

Ingestion of data into data cubes is ongoing, while procedures and validation approaches are being developed and tested. Common datasets, such as remote sensing-based data (e.g. Sentinel imagery) and products derived by Copernicus, are readily available (though in rather 'raw' form). For this use case specifically the Dutch land use dataset (LGN) for multiple years has been ingested and a virtual data cube created by rasdaman. Access to it has been tested with a Python Notebook (using the WCPS API and running on EOXHub). A required next step to use the data for machine learning / deep learning is to bridge between this WCPS API (common in the EO domain) and the Datasets and DataLoaders used by the frequently applied Python ML frameworks, such as PyTorch and Tensorflow (Keras). No existing implementations (e.g. software libraries or packages) that handle this have been found so far.

2.2.7 Causal / Explainable Machine Learning

The final goal of this use case is the discovery of causal associations (following already mentioned detection and attribution framework approach) between agricultural activities and changes in biodiversity, i.e., to not only find correlations in the combination of biodiversity, agriculture, and environmental data, but try to detect causalities. Its achievability depends on all previously mentioned steps to be successfully finished in time (which starts to get more critical now), without losing relevant details from the data, i.e., too much aggregation will prohibit detection of fine-grained relations.

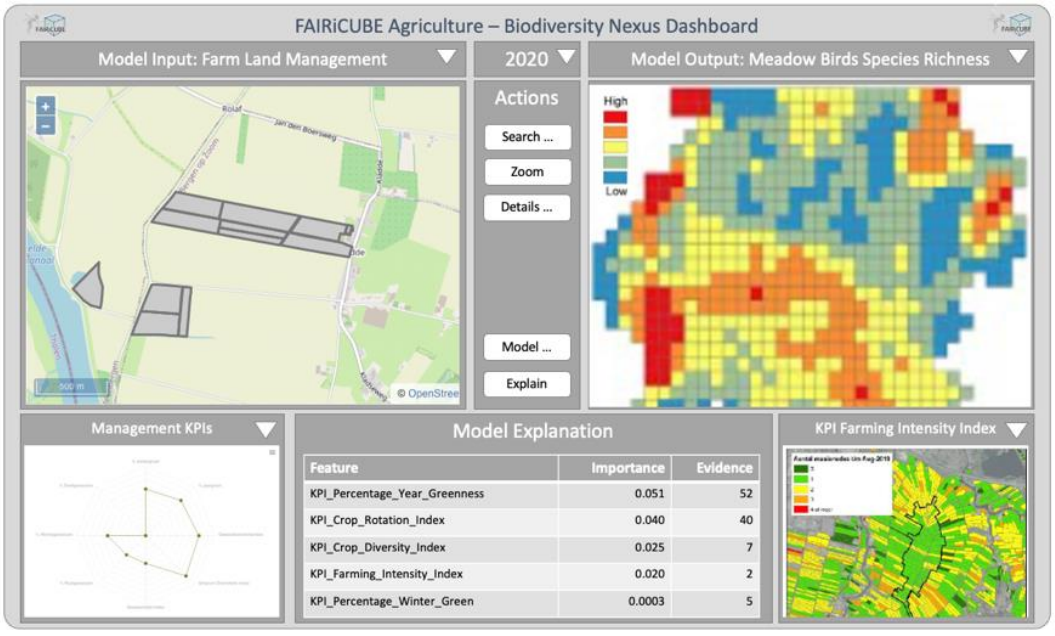

Figure 27 : UC2 Model serving - Dashboard application (proposed mockup, October 2022)

This topic is still a bit further away on the horizon with details about possible machine learning strategies hard to give. At a minimum the focus should be on interpretable models, the use of global and local model-agnostic methods (e.g. LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapely Additive exPlanations)) of explainability, feature visualization and saliency maps for deep learning models, up to causal association rules data mining, and causal machine learning. The latter,

however, will require the use of known causal graphs from the problem domain (agriculture – environment – biodiversity interactions), which might be difficult to find (in literature).

Meanwhile good progress is being made in setting up the processing environment for machine learning and deep learning. Specifically for the later, EOXHub has added headless execution of Python Notebooks that offers access to GPUs. As well as MLFlow Tracking for recording details on performed ML experiments and TensorBoard for the visualization of deep learning model training progress and results.

Decisions or requirements concerning the 'end' products (such as the dashboard mock-up in Figure 27) for the use case can impact choices to be made earlier on in the data engineering and machine learning. E.g., it might be good to 'reason backwards' from the end product as well when thinking about the grid, cell size, algorithms, and attributes to keep, when gridding data, in order to not lose relevant features and/or labels for machine learning purposes.

Further important aspects to consider for model serving and deployed applications are those of data, model, and concept drifts that over time can affect the operation (e.g., prediction accuracy). However, as products of a research project such concerns might be a bit less relevant.

2.3 UC3 Biodiversity occurrence cubes – Drosophila landscape genomics

UC3 aims to exploit the massive collection of DNA sequenced data of natural populations of the fruit fly *Drosophila melanogaster*. Apart from the challenge to adapt the data for storage as data cubes, the WP3 supports the processing and potentially application of machine learning algorithm to enrich the data set and reveal further insights while making advantage of the scalability and accessibility of data storage and processing capabilities of the FAIRiCUBE Hub.

As a first demonstrator task, gap filling of missing data in the genetic information of pools of individuals that were sequenced jointly (Pool-Seq) was identified. The provided data set DESTv2[1] comprises of more than 730 population-based samples of *Drosophila melanogaster* distributed over the globe. In a first step, we focus on populations in North America, which are predominantly collected along the East Coast. Many of these samples are densely collected across multiple seasons and years. This dataset, available at the so-called "Variant Call Format" (VCF), contains data entries for several millions of genomic positions for each population. These positions do not necessarily carry data for the entire number of populations. Additionally, missing data in one genomic position does not necessarily reflect which populations miss data in another position. To perform valid statistical analysis, positions with missing data cannot be included and the data used for various studies only represents a subset of the data available.

[1] https://dest.bio/

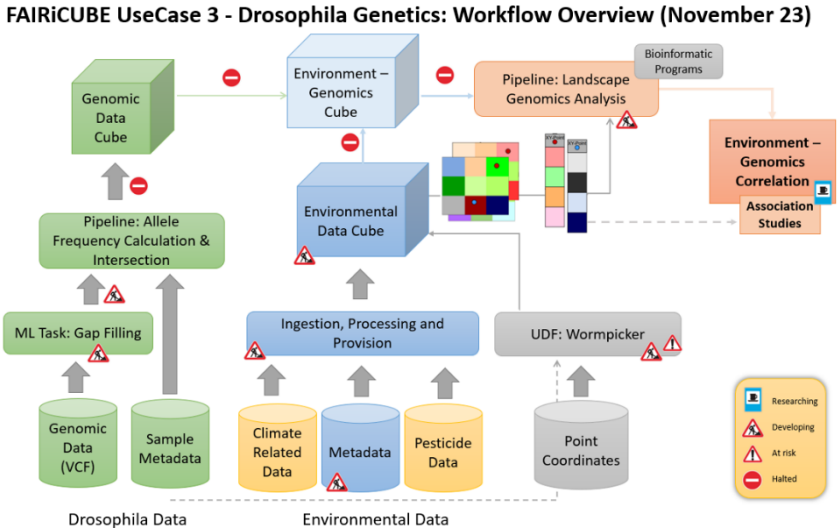

Figure 28 : UC3 data analysis and processing workflow. Machine Learning based gap filling methods can be applied to genomic data (lower left green cylinder: ”VCF”) to avoid non-usability of valid information and data on samples due to lack of data in other samples.

As described in the exploratory data analysis in deliverable D3.1, the provided test data that was generated to develop this method is without missing data, in contrary to the original DESTv2 dataset which characterized by different levels of missing allelic information in the population samples (D2.2). To evaluate the accuracy of an applied gap filling method, artificial missing data will therefore be introduced that mimic actual gap characteristics. The number of missing data sums up to about 5% of the total amount of allele frequencies and the distribution of the number of subsequent single nucleotide polymorphism (SNP) positions follows the observed exponential function of number of gap-occurrence with gap-size. Most of the gaps that have been introduce consist of single SNP locations.

A previous attempt has been made to gap-fill the data set by imputing missing allele frequency data based on information from close-by populations using inverse distance weighting based on geographic and temporal distance of neighbouring population. The DESTv2 dataset is particularly well suited for this, as many populations are closely sampled across space and time. To consider isolation-by-distance, due to limited dispersal among populations which reduces the relatedness (and potentially the similarity in allele frequencies), we restrict the inference to neighbours, which are within a user-defined geographic distance (in kilometres) and temporal distance (days between sampling dates). Moreover, we used inverse-distance-weighting (IDW) to average allele frequencies of neighbouring populations as a function of their proximity to the population with the missing allele frequency. This approach has been tested against a dataset who's artificially introduced gaps are distributed similar to the real dataset and provided satisfying results. However, assumption is that all neighbouring populations can have an equal relation regardless of geological boundaries. In the following this gap filling approach is addressed as *empirical baseline*.

An alternative approach will be provided through clustering of the existing populations regardless of origin of time and space. This is justified since mutation pattern can occur independently and recurrently. In preparation of the clustering data using machine learning techniques, the allele frequency data is loaded from TSV files, stripped off the header information, and converted to a NumPy array. In the context of the clustering approach, the populations are considered data points that are to be clustered, and allele frequencies are the features by which the clustering is performed. However, as there are 50,024 allele frequencies per population in the test dataset, we attempt a ML clustering in 50,024 dimensions and choosing a clustering algorithm by reviewing and visualizing cross plots of data is impossible. As an outcome of exploratory data analysis in deliverable D3.1, we have identified the position of allele frequencies with the highest variance across the populations which widens the

spectrum of differences in allele frequencies. After all, we do not want to cluster based on similarities of data but by different characteristics. By selecting only 99.5 % of the most variable allele frequencies, we reduce the dimensionality of our clustering to 251. Based on this, we can still not truly identify patterns of the data but can attempt a blind run of a *k*-means algorithm with application of the elbow method to find an optimal number of clusters *k* as input for the actual clustering.

k-means algorithms are usually the first choice in clustering attempts due to their simplicity and computational efficiency (see also chapter 2.1 for description of other clustering methods). After initialisation with random *k* centres of clusters, each data point is assigned to the closest cluster centre and an iterative process, the centres of the clusters are updated by re-calculating the means (centroids) of each cluster and re-assigning data points. The performance of *k*-means clustering depends on the number of chosen clusters *k*, the initialization of cluster centres and, most importantly, on the nature of data. Specifically, the dataset needs to exhibit higher-dimensional spheres as imposed by definition of the distance metrics to the nearest cluster centre and the calculation of means to update the new cluster centre. If data clusters shape differently, alternatives such as spherical k-means algorithms or DBSCAN (Density-Based Spatial Clustering of Applications with Noise) can be explored. Data with high dimensions can be further reduced in dimensionality using auto-encoder deep neural networks (DNN) which try to extract relevant characteristics instead of all dimensions of the data points – similar to our attempt to identify the most variable allele frequency positions. A subsequent application of e.g., k-means is then performing the actual clustering of lower-dimension data.

To find the optimal number of clusters *k* for our k-means clustering we apply the elbow method and perform several k-means clustering with a range of *k* from 1-15 and calculate the WCSS (Within-Cluster Sum of Square) for each run. When analysing Figure 29 and the plot of WCSS as a function of number of clusters *k*, we can observe a typical "elbow" shape of the curve which recommends k=5 for an optimal k-means clustering. A subsequent test with the silhouette method confirms this choice.

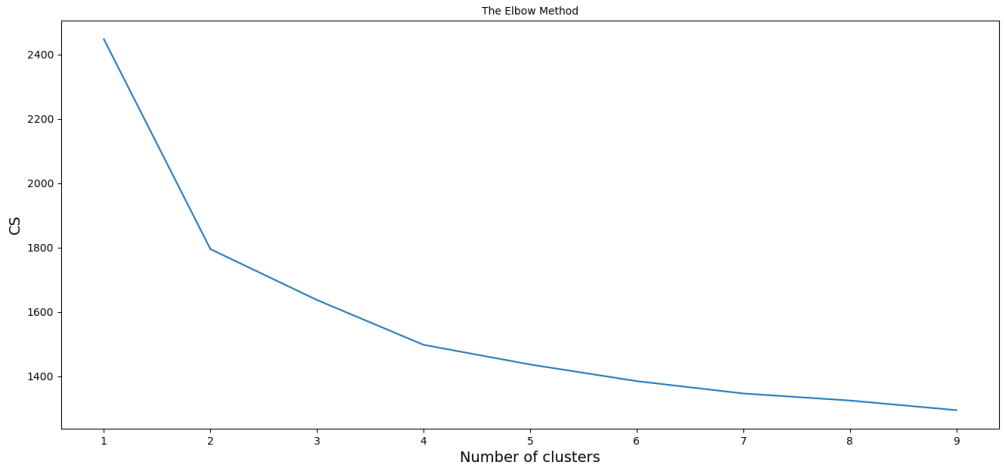

Figure 29 : Applying the elbow method to determine the optimal number of classes for the k-means clustering.

We can therefore perform a *k*-means clustering with *k*=5 and assign each population to a cluster number. Arguably, the recommended number of clusters is low which indicates large similarities in the allele frequency information across the populations and reflects the evolutionary history of world-wide *D. melanogaster* population very well (Kapun et al. 2020). The number and distribution of the clusters potentially thus reflects the geographic sampling and the evolutionary history of the investigated populations.

In the subsequent gap-filling step we can replace missing allele frequency data in one population with a mean value from populations belonging to the same cluster. The gap filling has been tested for selections of populations in Europe and North America separately. When assuming the introduced gaps

to be of zero values, we can calculate the summed difference and root mean square (RMS) error of data with and without gap. After gap-filling, we can compare the summed difference and RMS and calculate the ratio that indicates how well the gap-filling performed. The results of both the k-means clustering with *k*=5 and the previous inverse-distance weighting method are shown in Table 1. In the following the *k*-means gap filling approach is addressed as *machine learning (ML) baseline*.

Table 1: Statistics on the gap filling methods applied to selected populations.

	North American populations	European populations
Number of populations	121	332
Number of samples	6 152 952	16 708 016
Number of gaps	125 000	338 000
Total number of NaN	309 753	837 327
Percentage of gaps	5.03	5.01
Summed difference of gap data (rounded)	33794	90059
Summed difference after k-mean gap filling (rounded)	8216	256
RMS error of gap data	0.05189	0.05508

In order to improve the accuracy of the gap-filling we now employ sequential deep learning models. There are two types of models that we use: generative models and non-generative models. Generative models try to learn a probability distribution over the data and calculate missing values by sampling from the learned distribution. On the other hand, non-generative models try to impute the missing values directly from the data by exploiting the relationship between the known data points.

Table 2 shows the RMSE scores for the methods used to impute missing allele frequencies. The deep learning methods are benchmarked against the empirical and ML baseline. The best performing methods are the Masked Autoencoder, Variational Autoencoder and Generative Adversarial Network. A VAE operates within the framework of unsupervised learning, compressing data while maintaining the structure of input data. It introduces stochasticity in its encoding process, resulting in a structured, continuous latent space ideal for generating new samples. Similarly, Masked Autoencoders function in unsupervised settings, primarily focusing on reconstructing incomplete input data. They operate by masking a portion of the input data and learning to predict these masked values, effectively training the model to identify and fill in missing information. This approach is particularly useful in handling incomplete datasets, as it enables the model to learn the underlying structure of the data and accurately impute the missing values based on learned patterns.

GANs, on the other hand, comprise two neural networks: a generator and a discriminator, functioning in a competitive setup. The generator fabricates data aiming to pass as genuine, while the discriminator improves its ability to differentiate between real and synthetic data. Over time, the generator becomes increasingly proficient at producing realistic data.

Apart from MAE, VAE and GAN, we tried several other well-known machine learning and deep learning architectures such as: the KNN imputer that imputes missing values with K-nearest neighbours, Gaussian Process Regression (GPR) that imputes values by sampling from a learned distribution from the available data, Temporal CNN (TCN) is tailored towards sequential data based on supervised learning, LSTM/Bidirectional LSTM (LSTM/BiLSTM) are the go-to methods for sequential tasks and has the ability to learn sequential dependencies. These methods outperformed the inverse distance method and performed admirably.

Table 2 : RMSE on missing allele frequency imputation

Methods	RMSE NA	RMSE EU
No gap filling	0.05189	0.05508
Inverse distance (empirical baseline)	0.04330 = 19%	0.04511 = 18%
k-means clustering (ML baseline)	0.01137 = 79%	0.00131 = 98%
Masked Autoencoder (MAE)	0.00142 = 97%	0.00231 = 96%
Variational Autoencoder (VAE)	0.00467 = 92%	0.00093 = 98%
Generative Adversarial Networks (GAN)	0.00019 = 99%	0.00134 = 98%

VAE and GAN stand out in gap filling for genomic sequences due to their capability to generate plausible data points, crucial for handling the high-dimensional nature of genetic data. VAEs adeptly learn data distributions, enabling accurate imputations of missing values, while GANs, with their distinctive network architecture, create realistic sequences, enhancing data authenticity. However, the performance of GANs is hard to replicate as these models are notorious to train. It requires a lot of fine-tuning and balancing training between the generator and the discriminator. On the other hand, MAE also show comparable effectiveness in reconstructing incomplete data and predicting missing information, a key aspect in genomic studies where accurate and biologically plausible imputations are essential. All these models surpass traditional methods by innovatively maintaining genomic sequence integrity, however the family of autoencoder models are simpler, faster and easier to train.

Generally, even the fairly simple k-means ML clustering produce a highly accurate reconstruction of the missing data. Deep learning methods beyond the ML baseline model increase the accuracy as they have more layers and parameters which are capable of learning high dimensional interactions and correlations, especially in this case, where the input is a long sequence of 50,024 allele frequencies. The models defined above are capable of extracting features in such high dimensions. Also, another factor that contributes to DL methods' superior performance over ML baselines is that the DL models have been defined and implemented in a such a manner that the sequence of allele frequencies is preserved. This is imperative in this case as we are dealing with sequential data.

As a next step, we will focus on the actual DESTv2 with gaps that comprises of more than 730 population-based samples of Drosophila melanogaster distributed over the globe and more than 3 million SNP positions. This data will have gaps where we don't know the true values to compare against. Only the general distribution of Allele frequencies and a-priori information can be used to validate the results. It will further be numerically challenging to handle the whole records of more than 3 million SNP positions at once, we will therefore consider a window approach to input missing data piece by piece.

2.4 UC4 Spatial and temporal assessment of neighbourhood building stock

To create a building stock model that can be used to estimate the building energy performance, climate gas emissions and the building's material composition, we take the join outcome of the IEE Project EPISCOPE and TABULA[1] as a starting point (see also the final EPISCOPE report[2]). Country wise, a building classification is presented and a straightforward energy performance and building composition estimation is published there. Given the availability of the input parameters (construction year, building type and total floor area) we can thereby directly link public city data to our desired output parameters.

First, we will focus on the parameter *total floor area* which can be derived from the building volume being – in the simplest definition of the building Level of Detail 1(LoD) - the product of the ground area

[1] https://episcope.eu
[2] https://episcope.eu/fileadmin/episcope/public/docs/reports/EPISCOPE_FinalReport.pdf

and the building height. However, finding complete height buildings datasets is challenging. Gap filling is therefore a must and different methods need to be considered. Even though we strictly only need the number of floors and the ground area of a building to relate to the EPISCOPE building classification, we see the building height as a more widely available data source which can be further improved using gap filling.

Many approaches exist in literature for estimating building heights from data. Those include a simple multiplication of the number of floors by a constant ceiling height, machine learning tree-based approaches, estimation from digital elevation models and more advanced as, for example, estimations based on satellite data. In the context of our FAIRiCUBE use case, three approaches were used to estimate building heights. The city of Halle, Germany, was selected as a test case as we have available the ground truth building height data, the input data to all methods in focus (see also deliverable D3.1) and the published machine learning model could be applied without problems. For larger cities, like Vienna, Austria or Oslo, Norway which were identified originally as target cities, we ran into numerical issue of the published ML method most likely due to the size of the cities. Once we conclude from the city of Halle test case, we revert to the original selected European test cities to allow for synergies with other FAIRiCUBE use case, e.g. UC 1.

We now focus on the first method which is using Open Street Map (OSM) to extract the *number of building floors* and multiplying them by a *constant ceiling height*. Given the limited availability of ceiling height data for Halle, a region around Halle was selected to get a better representation of the ceiling height constant. The dataset contained information of 230,255 buildings of which only 20% (18,837) had the number of levels and just 850 both heights and number of levels. The average level height for the dataset was 2.5 m. In a brute force attempt, we tested different constants ranging from 2.4 m to 4.3 m as potential ceiling heights and calculated the mean absolute error (MAE) and root mean squared error (RMSE). The optimal ceiling height was given with 3.0 m with a MAE of 2.7 m. Later, the dataset was cropped to the outline of the city of Halle and building heights were computed. The results are shown in Figure 30.

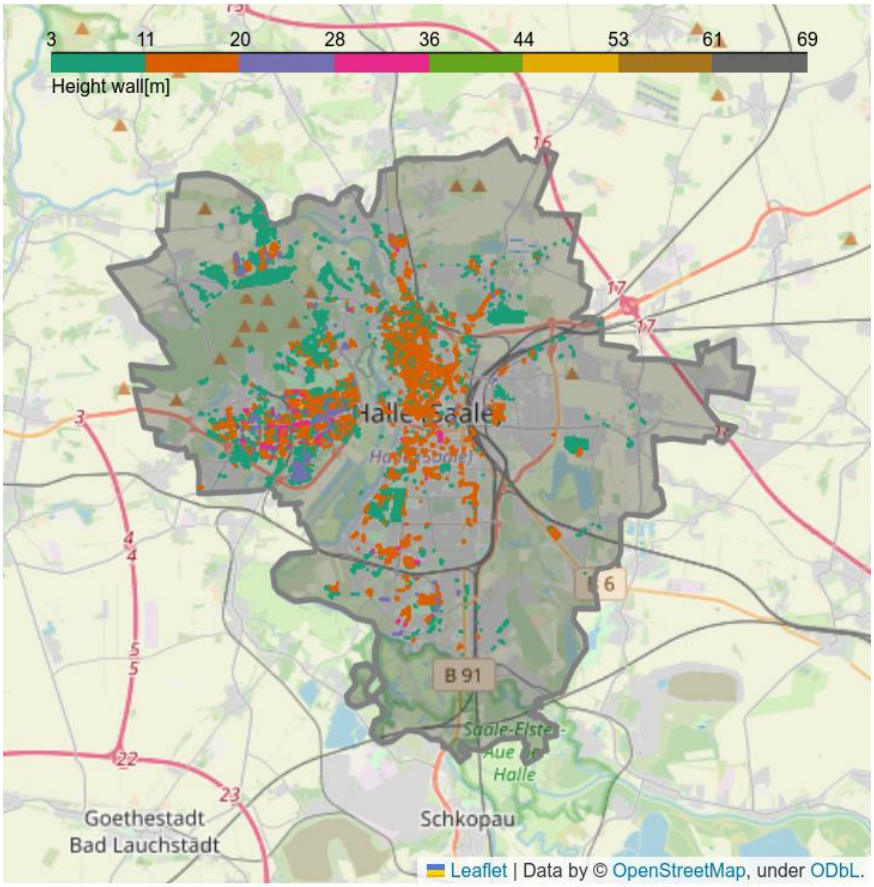

Figure 30 : Building heights [m] estimated by the multiplication of the number of levels by a constant.

Despite its easy implementation, the number of floors times a constant ceiling height method is not always applicable as the number of floors for a whole city is rarely available. Moreover, the method can introduce bias, as the height of the levels is generally heterogeneous and can vary significatively between buildings, city neighbourhoods and cities.

Regarding machine learning (ML), tree-based algorithms are the most used for this kind of application. In this work we have used a random forest ML algorithm, implemented in the ready-to-use Geoclimate[1] software, a geospatial processing toolbox for environmental and climate studies.

A decision tree is a kind of flow-chart with tree structure with nodes, leaves and branches. On each node, a test is carried out on an attribute of a dataset. The outcome of the test represents a new branch on the tree, and it is stored in a terminal node known as leaf. Then recursively introduce new tests on each of the terminal nodes (leaves) hence splitting the tree until a stopping criterion is met. The last output is compared to a known true value and the loop repeats. A metrics is used to select the paths maximizing information gain. What the random forest does is to combine the output of multiple decision trees to reach a single result with lowest bias and variance in the results.

The random forest model for the estimation of building heights implemented in Geoclimate was trained on a dataset of 14 French communes with reference heights provided by BDTopo IGN. For inference (prediction), the software first retrieves the building layer of Open Street Map (OSM) and to compute 62 geographical indicators from a building's closest environment. These features are fed as features (predictors) into the random forest algorithm to predict building heights. Geoclimate is open source. As

[1] Jérémy Bernard et al., Estimated height of the OpenStreetMap buildings of 24 French communes using the GeoClimate Software (version 0.0.1) (2021), , doi:10.5281/zenodo.5746612.

the machine learning model is available through the paper publication, we can directly apply it to other cities, like our city of Halle, Germany. For a wider application, to especially larger cities, a re-write or at least debugging of the existing ML inference is however necessary. Figure 31 shows the prediction of building heights for Halle, Germany.

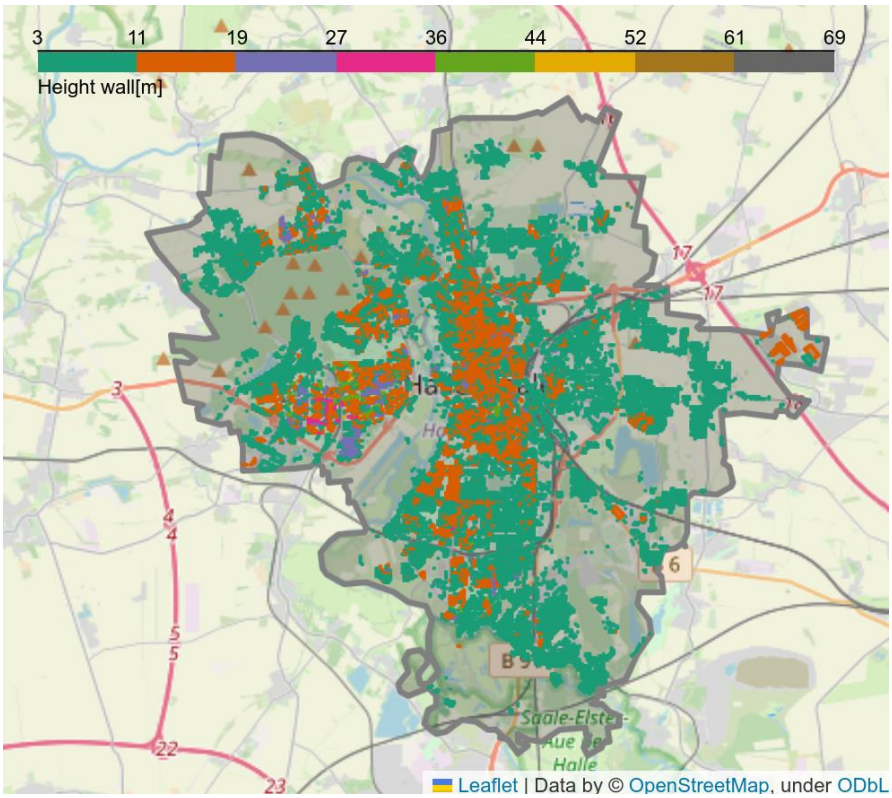

Figure 31 : Building heights estimated by random forest algorithm using the Geoclimate software for the city of Halle, Germany.

Despite the capability of Geoclimate at the current development state, the software still needs fixing some stability issues. Debugging and improving the code can be challenging as it requires knowledge on Apache Groovy, a java syntax-compatible object-oriented programming language for the Java platform not so known in the data science community.

A third approach is using estimating building heights form the difference between a Digital Surface Model (DSM) and Digital Terrain Model (DTM) data as illustrated in Figure 32. Only building heights greater than 3.0m were included in the results shown in Figure 33. Of all the three methods described here, this method is the most stable and least computationally demanding one as it only requires the subtraction of two data layers. However, high resolution DSM data is not widely available in Europe whereas DTM data is available through the digital elevation model (DEM) provided by the Copernicus land monitoring service[1].

[1] https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1

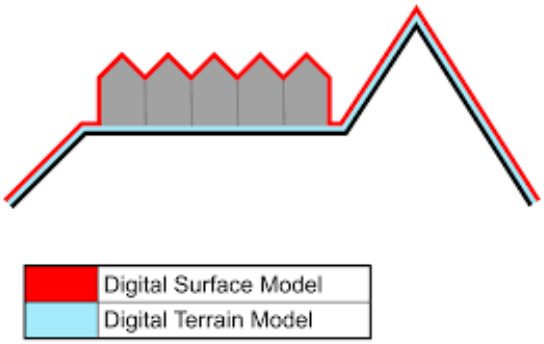

Figure 32 : Illustration of Digital Surface Model (DSM) and Digital Terrain Model (DTM).

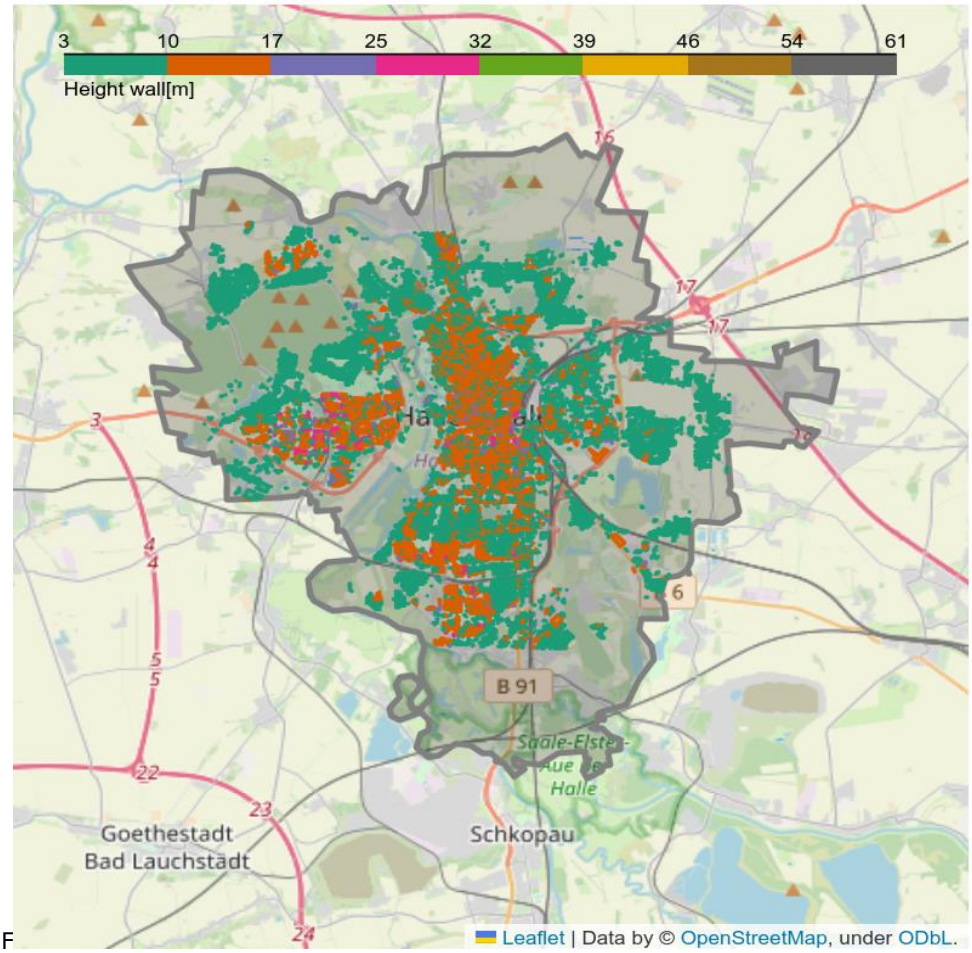

Figure 33 : Building heights estimated by the difference of DSM and DTM for the city of Halle, Germany.

Table 3: Descriptive statistics of building heights in all the GeoTiff layers.

Parameter	Ground truth	Number of levels x factor	Geoclimate	DSM-DTM	Copernicus building height dataset
Max	62.3	45	66.0	30.4	45.0
Mean	8.12	11.9	10.3	9.6	7.7

Min	0.06	6.0	3.4	3.0	3.0

As a final step of the building height estimation task, the output of the three different methods and the Copernicus building height dataset is compared against the ground truth. The ground truth data was provided as 53 separated files in cityGML format which is an extension of the GML standard that handles 3D data such as varying building height information complying to level-of-detail 2 (LoD 2). Note that the ground truth is still not 100% accurate as it is also combines the processed input from several data sources as described in deliverable D3.1. A QGis script was created to convert each GML file into GeoJson and posterior merge into a single GeoJson files. Finally, the merged GeoJson was rasterized and exported as GeoTiff to make it directly comparable to the output of our building height estimation layers.

Binary overlap layers between the GeoTiff layer of the ground truth and the results in all the three estimation methods were generated. The different overlap layers were used as a binary mask to extract data from the different estimation results and the ground truth.

Table 4 shows a descriptive statistic of the layers overlapping with the ground truth.

Table 4: Descriptive statistics of building heights in all the GeoTiff layers.

Parameter	**Ground truth**	**Number of levels x factor**	**Geoclimate**	**DSM-DTM**	**Copernicus building height**
Max	62.3	45	66.0	30.4	45.0
Mean	8.12	11.9	10.3	9.6	7.7
Min	0.06	6.0	3.4	3.0	3.0

To evaluate the accuracy of all estimation methods, the root mean square error (RMSE) with respect to the ground truth data was calculated. The result is presented in Table 5.

Table 5: RMSE in the estimation of building heights by different methods with respect to the ground truth.

Method	**RMSE [m]**
Number of levels x constant	2.41
Geoclimate-random forest	3.18
DSM - DTM	2.46
Copernicus building height dataset	3.38

Originally, the city of Halle, Germany, was a testcase to perform the building height estimation using several methods. The data availability was good and the provided ML algorithm for the random forest regression was running stable. However, depending on the data availability and completeness for other cities we are not able to recommend a method that will likely succeed in all cases.

Naturally, the preferred and recommended approach is to obtain local data from municipalities, similar to the Halle ground truth data. This kind of data seems to be very fragmented within European cities and is not available as a European data layer. The Copernicus urban atlas building height layer[1] is a good starting point but is as well an advanced version of the *DST-DTM* method. If local building height data is not available, we can refer to the methods tested in this work. According to Table 4 the lowest RMSE is given by the method *number of levels x constant*. However, given the sparsity of the OSM data *number of building levels*, building heights cannot be estimated for a majority of buildings. A more complete estimation can be provided via subtracting *DSM – DTM* data which however might not widely

[1] https://land.copernicus.eu/local/urban-atlas/building-height-2012

be available for European cities at our target spatial resolution of at least 10 m. The European data layer digital elevation model (DEM)[1] is for example provided at a spatial resolution of only 25m.

From the viewpoint of the data basis, the most stable method is actually the Geoclimate ML method using OSM input data. OSM data may however vary in accuracy and quality due to its crowd source approach. If we can overcome the numerical issues of executing the Geoclimate ML model on large cities, we can obtain a good base line estimation of building heights or number of stories per building.

We now have derived a building stock model according to the LoD 1, e.g., each building is approximated as a building block with volume given by ground area and height. As a next step in our task, we will assign the building volumes with construction year and classify according to the building types as defined by the Episcope data base. Combining all the input layers finally allows us to estimate energy performance, building material composition and additional measures.

2.4.1 Processing workflow

The overall progress for UC4, as of December 2023, is presented in Figure 34. Same logic as in UC2 holds here, meaning that the traffic sign icons show the general state. However, processes that haven't been marked by any sign icons has not started yet. Explanation of the data sources and processes are provided below based on the provided tags in Figure 34.

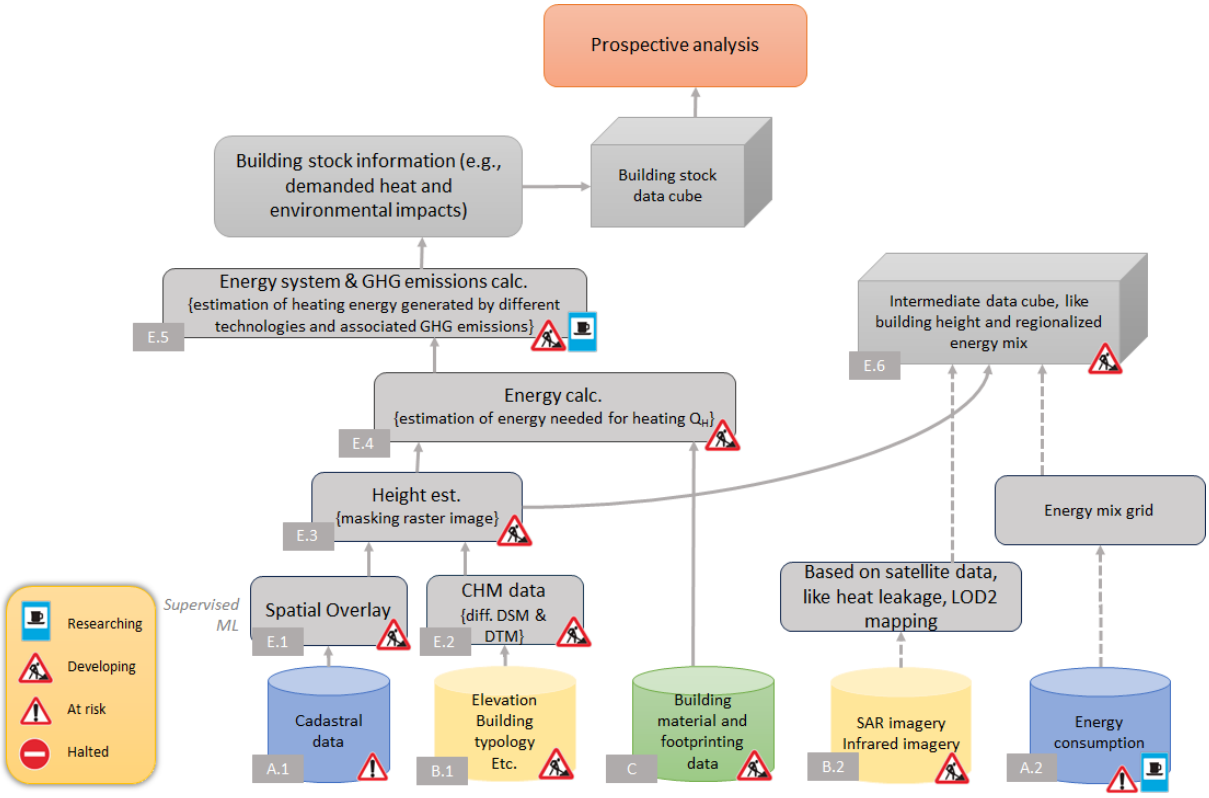

Figure 34 : Progress within UC4 data analysis and processing workflow

[1] https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1

Data sources

Ground-based data
A.1. Cadastral data: Here a combination of community-based data, and local data are considered. Open Street Map (OSM) is used to extract building geometries, while local data containing building information like age and functionality is collected from municipalities. From the OSM data only polygons are extracted, meaning that, point and multi-polygon geometries have been excluded. In addition, A set of point data from the municipality of Oslo is collected. The spatial data present a static information about the building cadastre in Oslo, like construction year, building type, number of floors, etc.

A.2. Energy consumption: So far, UC4 has been using average energy mixes for different building archetypes to estimate the heating demands. This approach has been taken since UC4 did not have an energy mix map projecting the final energy mixes consumed at residential level. It has been identified a couple of potential sources to either make such a map or use an existing one.

Remote sensing data
B.1. Elevation and building typology: Data representing Digital Surface Model (DSM) and Digital Terrain Model (DTM) are used here to estimate building height and subsequently their geometries.

B.2. SAR and infrared imagery: It is of interest to test the feasibility of using Synthetic Aperture Radar (SAR) imagery to estimate building heights. In addition, it is of interest to test the possibility of using infrared data to estimate heat leakage (relative thermal inefficiencies of buildings). European Space Agency and Sentinel hub are the two sources that are considered for SAR and infrared data, respectively.

Tabular data
C. Building archetype information (like environmental footprint per mass of building and heat transfer coefficient of different building elements) is collected from different sources. TABULA/EPISCOPE datasheet is the main source of information here. It contains information about architype building information in certain countries in Europe with their corresponding climate information. In addition, other sources of information are used to estimate in-use mass of materials to estimate availability of secondary construction materials from buildings and estimation of embodied environmental impacts associated with them.

Processing work
E.1. Spatial overlay: Cadastral data are spatially merged here. It might happen a building has a several purpose in its use (like, a building might have grocery store underneath, or coffee shop etc.). In the overlaying process, UC4 exclude building that might have multipurpose/functionality to ensure that the energy balance calculation do not underestimate energy needed for heating.

E.2. Canopy Height Model (CHM): Canopy Height Model contains the height difference between DSM and DTM. However, before this, mosaic of DTM and DSM is created, as the attained raster images are not in one peace.

E.3. Height estimation: Each building geometry is masked over CHM data, and the average height attained from each masked image is stored. The height value is later used to estimate the volume of each residential building and calculate the shaded wall between residential buildings.

E.4. Energy demand calculation: This block estimates the energy needed for each residential building based on the energy calculation formula provided by TABULA/EPISCOPE[1]. The estimations are based on three archetypical scenarios for each residential building: 1) the as-built design, 2) intermediate renovation, and 3) advance renovation. For newly constructed buildings, the as-built design is compatible with intermediate renovated or advance renovated building. Figure 35 presents the as-built energy demand for space heating for residential buildings in Oslo.

[1] Diefenbach et al. (2013). TABULA Calculation Method – Energy Use for Heating and Domestic Hot Water. URL: www.building-typology.eu

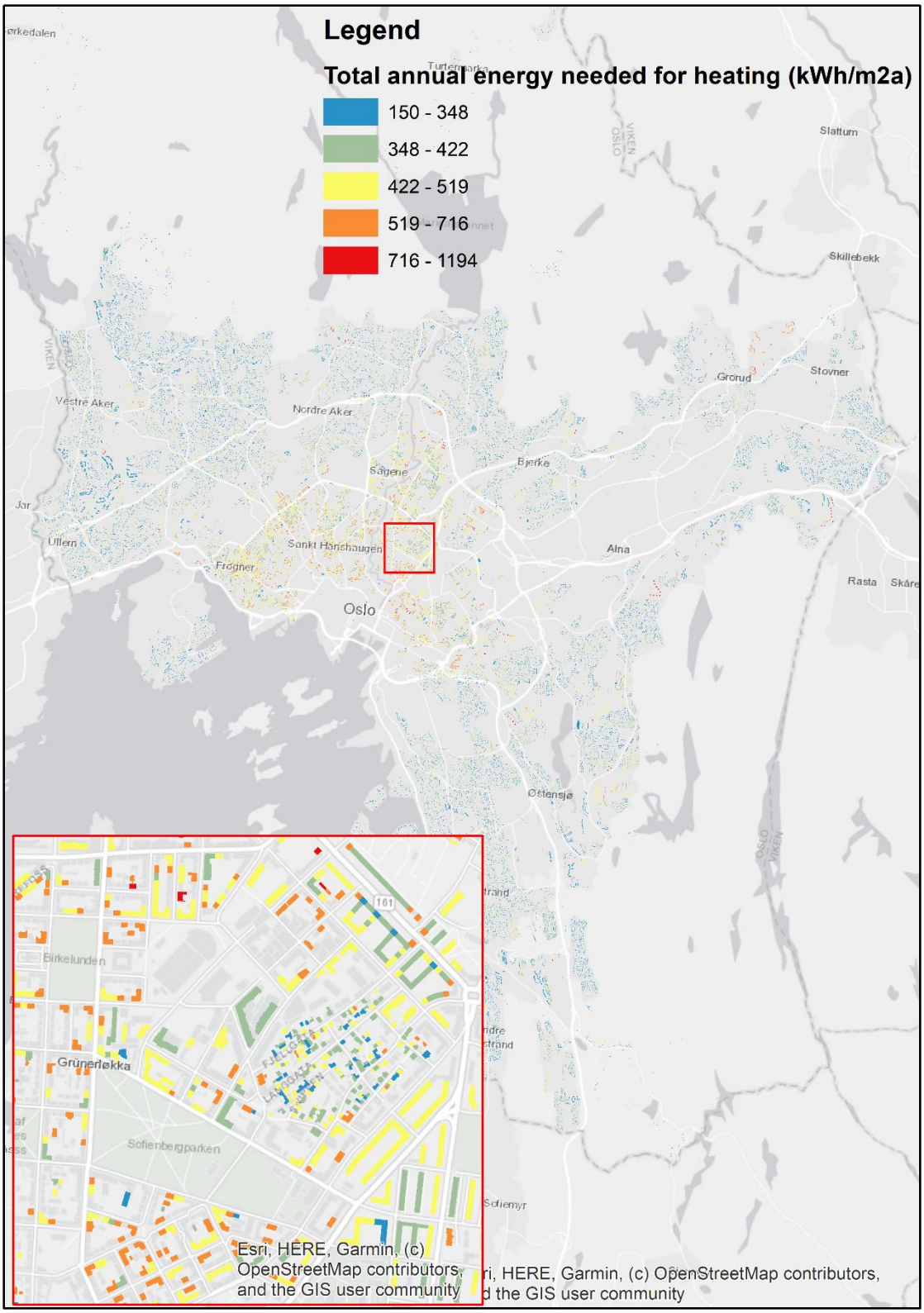

Figure 35 : As-built annual energy needs for space heating (kWh/m2) for residential buildings in Oslo.

E.5. Energy system and GHG emissions calculations: Based on the attained energy demand from Process E.4, a series of energy systems are modelled to satisfy the energy demand. TABULA/EPISCOPE are used to extract the list of potential energy systems for the designated countries. In addition, GHG emissions associated with each energy system are calculated to present embodied environmental impacts.

E.6. Intermediate data cube: some of the attained intermediate results in UC4 might have potential use for other purposes. Hence, such results are stored in grided format. Figure 36presents an example of such results. It shows the grided building height for the city of Oslo. Such an image might be of interest for identification of rooftops with high potential for photovoltaic panel or green roof applications.

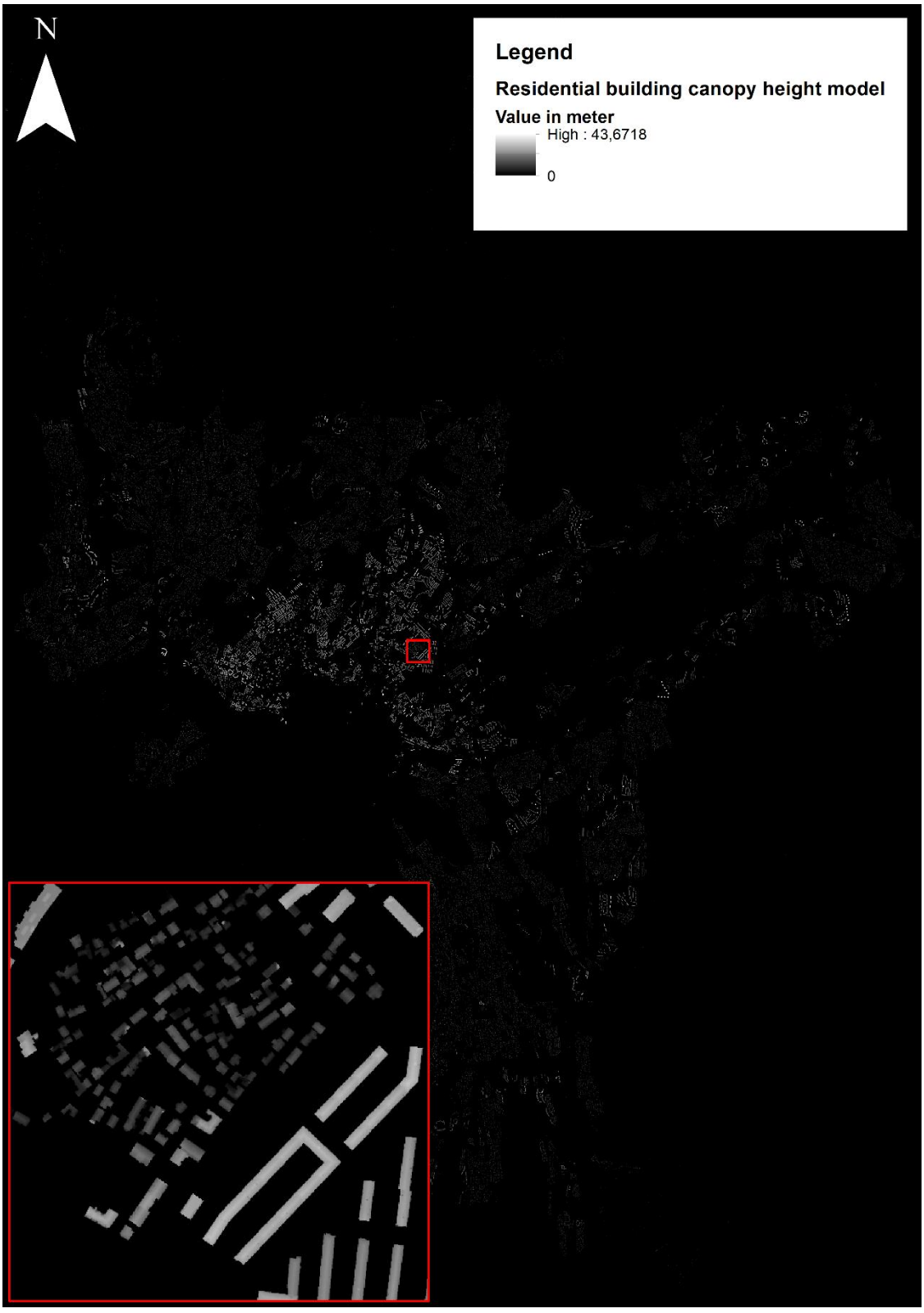

Figure 36 : Canopy height results for residential buildings in Oslo.

2.4.2 Next step

Based on the progress of UC4, the following activities are expected to take place: (1) extension of building energy calculations and data gap filling, (2) testing validation of the attained energy calculations, (3) identification of the deriving forces in energy retrofitting, (4) stochastic modelling of identified drivers on energy retrofitting, and (5) performing multi-objective optimization programming.

Points 1 and 2 in the overseen activities are linked to each other. This implies that the energy performance calculating will first be extended to cover the energy demand and potential energy mix delivered to a combination of building types for the four cities of choice in UC4. Here, the energy demand and delivery estimation and estimate in-use building materials and GHG emissions associated with each scenario. Secondly, the estimate results (concerning the energy performance and in-use materials) will quality controlled with reference values. The second step is a crucial step and requires local values. At this stage, we are attained local data from Norway to carry out a validation test for the city of Oslo. The similar approach is expected to be taken for the remaining cities. Besides, the work will test potential data gap filling approaches and document the findings.

Based on the validation results from each city under investigation, statistical modelling will be carried out to identify key factors influencing renovation rate of residential buildings like income, energy price, age, and education of registered residents in a neighbourhood. The importance of such factors will be assessed based on findings from other scientific works. After the identification of important factors and their causalities, a stochastic model will be developed to test the effect of different actions in the future. This will be useful as it assists in understanding the dynamics of different factors and how they influence the energy retrofitting.

The last step will deal with optimization modelling to calculate optimal solutions to meet the climate and energy goals that each city is aiming to achieve by 2030.

2.5 UC5 Validation of Phytosociological Methods through Occurrence Cubes

The use case on the *validation of phytosociological methods through occurrence cubes* (UC5) started with a significant delay compared to the other use cases due to staffing issues and exploitation of synergies with a sister project funding under the same call. UC5 aims to validate the traditional methods applied in phytosociology to characterize and classify plant communities and to develop a new phytosociological approach to characterize and predict the presence of plant communities for yet unknown localities. This will be approached by linking distribution data of plant taxa and vegetation communities based on habitat types with EO environmental data.

Currently, UC5 is in the data exploration phase which is prerequisite for entering the machine learning phase. No machine learning strategy can therefore be reported and documented yet. A future scheduled update of this deliverable will cover the progress from UC5 as well.

3 Summary and conclusion

Given the status of the data ingestion, the UC owners progress to identify and describe scientific research questions in terms of data relationships that are to be discovered and exploited, and the results from the exploratory data analysis (deliverable D3.2) a first draft of a machine learning strategy was created for each use case.

UC1 performed several unsupervised clustering exercises using different ML algorithms to find similarities of European cities according to the Urban Atlas land classification. Based on only the level 1 coverage ratios, several clusters have been identified and described. A k-means clustering was established as a baseline model.

UC2 invested valuable resources to implement, develop and showcase a ML model inference close to a high performing data base engine which – besides the compute aspect – is usually a bottleneck in ML applications. Following the outlined machine learning strategy, the focus is currently how to create occurrence cubes from sparse biodiversity observation data.

UC3 performed a gap filling exercise of allele frequency data of DNA-sequenced populations to improve an empirical base line model. First an unsupervised k-means clustering (ML base line mode) was used to identified similarities in genetic information which can be exploited to infer the missing data. In a next step, more advanced ML methods using deep neural networks could further improve the accuracy of the gap filling. Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN) provided the best results given modest increase in computational efforts.

UC4 tested several published methods to estimate the building height which can as well seen as a gap-filling exercise as some data sources provide sparse building heights or at least the number of building stories. One of the methods used a provided ML model, others are based on simpler numerical relations. Subtracting a digital elevation model from the digital surface model finally provided the lowest error metrics for the selected test city.

Finally, UC5 started with a significant delay, and the machine learning phase has not been explored yet.