Ernst-Abbe-Hochschule Jena
University of Applied Sciences

Max Planck Institute
for Biogeochemistry

Evaluating global patterns of leaf area using Machine Learning approaches

A Thesis work submitted to Ernst Abbe Hochschule Jena

In partial fulfilment of the requirements for the degree of

Master of Science in Scientific Instrumentation

By

Vamsi Krishna Kommineni

Matriculation number: 642296

At

Max Planck Institute for Biogeochemistry

Jena, Germany

Under the guidance of

Prof. Dr. Henning Kempka (Professor, Ernst-Abbe-Hochschule Jena)

Dr. rer. nat. Susanne Tautenhahn (Postdoctoral Researcher, Max Planck Institute for Biogeochemistry)

July 8, 2020

DECLARATION

I, hereby solemnly declare that this thesis work is a presentation of my original research work. Wherever contributions from other sources are involved, every effort has been made to cite and acknowledge this clarity, with due reference to the literature.

Vamsi Krishna Kommineni

ACKNOWLEDGMENT

I express my sincere gratitude to Dr. rer. nat. Susanne Tautenhahn (Postdoctoral Researcher, Department of Functional Biogeography) for providing me this opportunity to do the Master Thesis at Max Planck Institute for Biogeochemistry. You have always been the backbone for me and thank you for giving me continuous ideas and promptly reviewing my work.

I would like to thank Dr. Jens Kattge, Dr. Jitendra Gaikwad, Mr. Pramod Baddam (B.Tech.) and Dr. Martin Jung for the continuous support and for giving insight feedback throughout the thesis work. I would like to acknowledge Mr. Ulrich Weber and Dr. Sujan Koirala for providing the data and giving more ideas about the scientific data.

I must express my very profound gratitude to Prof. Dr. Henning Kempka for your sincere support in every aspect and giving me complete independence in moving forward with my research work and timely technical support without hesitation whenever necessary.

None of this would have been possible without the support of my family. I express my heartfelt gratitude to my family and friends for their love and patience.

With Best Regards
Vamsi Krishna Kommineni

ABSTRACT

In this master thesis, I want to pave the way for investigating the intraspecific variability of leaf areas on a global scale for *Salix bebbiana*, *Alnus incana* and *Viola canina*. Since there are no extensive studies for leaf area variability especially not for longer time scales, I decided to collect the data from the digitized herbar specimens to study pervasively.

To this end, I want to analyze the variability of leaf area for three selected species from digital herbar specimen using image recognition techniques (Trait Ex). I want to investigate environmental drivers of leaf area by taking various measures like features related to spatial variation in environment (soil data, current climate data), features related to climate change over temporal domains (change in climate data sets) and features representing the season (day of year) (figure 1) into the account using machine learning techniques.

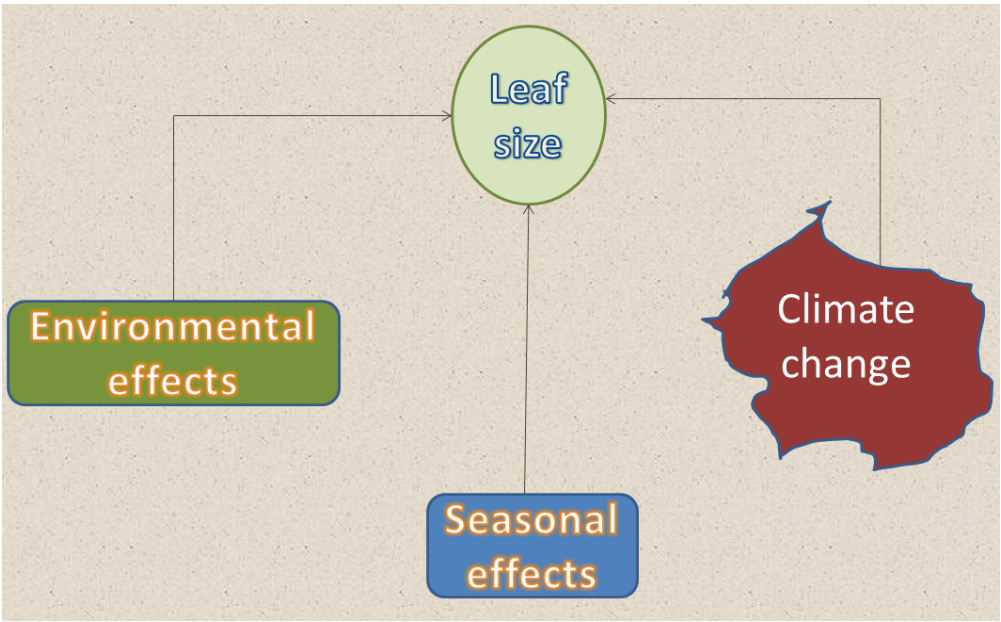


Figure 1: Factors affecting leaf area

The specific objectives of this master thesis are Trait Ex optimization and giving first user potential feedback of Trait Ex for the automation of Trait Ex tool and building machine learning models to investigate the intraspecific variability of leaf area for three species to environmental data, climate change data and seasonal data. This study will serve as a prototype to investigate the interspecific variability of leaf areas for all available species (worldwide) in the future.

Finally, I found the environmental factors that affect the intraspecific variability of leaf sizes for three selected species. In addition to this, I implemented different machine learning methods and technical approaches to get better insights into the data. Overall the results are stupendous and this study serves as the prototype to evaluate the interspecific trait variability of leaf areas.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Idea

One of the greatest challenges facing humanity is to understand the mechanisms by which human activities are altering the environment and biota of our planet [1] [2]. Due to the changes made to the environment by man-made activities (deforestation, pollution), global temperature's are alarmingly raising. According to NASA and NOAA, the past five years (2015 - 2019) are, collectively, the warmest years in the modern record [3]. According to the IPCC report, due to rapid industrialization from the last three decades, the global temperature was unexpectedly raised nearly 1°C compared to earlier decades [4]. The vegetation has to adapt to changing environments and at the same time modifies it according to the changes in climate and other environmental conditions.

Plants influence everything from food chains to climate change. They store carbon, fix nitrogen and produce oxygen. They shape weather patterns, provide flood defence, purify water, provide food, and offer solace and inspiration [5]. One needs to understand how the plant species are adapting themselves to environmental change. One way to understand adaption is reporting about relationships between environmental factors and the intraspecific changes in plant functional traits.

Plant functional traits (e.g. stem, roots, leaves etc.) are the morphological, physiological and phenological characteristics that represent ecological strategies and determine how plants respond to environmental factors, affect other trophic levels and influence ecosystem properties [6]. Leaf traits are the most important functional traits among all available plant traits and using leaf traits one can predict the performance of the whole plant [7]. The major factors affecting the leaf traits are temperature, precipitation, geographical location and humidity etc [8] [9].

Leaf size is coordinated with many other features of plant architecture, canopy display, and plant hydraulics, apparently leading to many equally viable leaf size strategies for a given climate [9]. Nonetheless, it appears that climate provides the dominant control on the global geographic limits to leaf size on the interspecific level [9]. The most interesting features that influence the leaf sizes are mean annual

temperature, mean annual precipitation, growing degree days(GDD), mean annual irradiance, humidity, wind speed, soil types, water availability, nitrogen content, texture of the soil and cation exchange capacity etc [10] [11] [9].

Space for time substitution To understand and predict the adaptation of plants to changing environmental conditions, frequently spatial gradients are used as natural laboratories reflecting the temporal effects [8]. It is very cost and time-intensive to travel all around the world to collect field data on plant functional traits like leaf size of large numbers of species from different eco-regions. Even more, it is impossible to study such relationships on a temporal time scale on which climate change is happening, which means centuries. It is unclear whether relationships over spatial gradients can be used to predict temporal gradients.

Herbar specimen Recently, initiatives extracting trait information [12] from digitized herbar specimens using machine learning became more popular due to the increasing samples of digitized herbar specimen. The Herbar specimen [13] is the collection of preserved plant or plant parts and the associated data used for scientific study. These specimens are usually in dried form mounted [14] on a sheet of paper as shown in figure 1.1 and available in the databases Gbif (refer section 2.1.1) [15] and Idigbio (refer section 2.1.1) [16] as digital images. In today's digital world, millions of herbar specimens are digitized and the number is increasing daily. By combining this new data source with image recognition techniques it became possible to observe adaptations in leaf size over large spatial and especially large temporal domains, which is very new. By combining this data with paleontological and historical climate information from climate simulation outputs and further gridded environmental data (e.g. different models of CMIP5 [17]) it is for the first time possible to study the adaptation of leaf size to changing environmental conditions worldwide on the temporal scale of several centuries.

Figure 1.1: Herbar Specimen

Trait Ex [18] is a semi-automated image processing tool which is used to measure the morphological functional traits from digitized herbar specimen. Trait Ex can measure quantitative traits such as the length, area, width and perimeter of leaves along with the petiole length from digitized herbar specimens [19]. Since Trait Ex is a semi-automated tool, one can measure the morphological functional traits of herbar specimen with freehand only (refer section 2.1.4). It is possible to automate the Trait Ex with some efforts.

Trait Ex has been developed only recently and until now it was not applied to a large number of herbar specimens. So, first user feedback would be needed to optimize the functionality of Trait Ex for future users. The Trait Ex automation depends on the coupling of Trait Ex with the automated image segmentation, which may be available from the Trait Ex team within the next years. This will

open new ways to measure thousands of herbar specimen traits within a short time.

Study species In this thesis, the sample species *Salix bebbiana*, *Alnus incana* and *Viola canina* had been selected based on the following reasons:

1. A high number of digital herbar specimens are available compared to other species.
2. The species should be widely spread across the globe and well distributed over the time of sampling, this gives us a better understanding while analysing the results.
3. The species are selected in such a way that these herbar specimens are compatible and easy to work with Trait Ex as discussed in section 2.1.4.

Random Forests [20] is one of the supervised machine learning algorithms and it is also the most used algorithm compared to other machine learning algorithms because of its robust nature, great accuracy and good interpretative results. Random Forest is an ensemble method that builds multiple decision trees and merges them to get a more accurate and stable prediction.

The reason for selecting Random Forest is that the functional relationships between the environmental data and leaf area should not be defined by us, the Random Forest automatically finds the relationship in the given data. Since a lot of environmental features can potentially influence leaf area, it is important to identify and select the most important ones to build a good machine learning model using feature selection (explained below).

Partial Least Square linear regression model is a very simple machine learning model which will look after only linear relations between the data. It is computationally beneficial (fast run time) compared to nonlinear models like Random Forests. The main idea behind the usage of the linear model in this project is to save the run-time of the model and to see whether there are non-linear relations available in data (whether performance is reduced compared to random forest model).

Feature selection is the process of selecting the features automatically that contribute most to the response variable. In most of the cases, feature selection is required in machine learning because the training time for the model reduces, the model is computationally fast and also over-fitting is omitted and therefore cross-validation correlation usually increases.

Spatial auto-correlation [21] reports the presence of systematic spatial variation in a variable. Some of the examples for the spatial auto-correlation is geological data, house price's at the same location etc. In the case of spatial auto-correlation, components of data set are correlated with itself in terms of spatial distances and therefore the assumption of independence is violated (Random Forest model and linear model).

If spatial auto-correlation is present in the given data, then the machine learning model gives us the overestimated R^2 value. One can overcome this positive or

negative auto-correlation by block cross-validation strategies (figure 2.27) [22], these strategies give the way for the realistic R^2 value.

Chapter 2

Methods used

2.1 Data acquisition

Data acquisition plays an important role in this thesis because if the data is not processed correctly at any stage of data extraction, then the model will be worse than expected. So, great care had been taken while extracting the data from different databases. The images from scanned herbar specimen were processed and downloaded from Gbif (refer section 2.1.1) [15] and Idigbio (refer section 2.1.1) [16]. After that, the leaves of the herbar specimens had been measured using Trait Ex (refer section 1.1).

In addition to this, the huge amount of environmental data had been processed from different databases named CMIP5 (refer section 2.1.6) [17], Worldclim (refer section 2.1.6) [23], Terraclim (refer section 2.1.6) [24], SoilGrids (refer section 2.1.6) [25] and Harmonised world soil data (refer section 2.1.6) [26]. As discussed, the environmental data had been processed from different databases because the leaf area has to be analysed based on both current climate, climate change, soil and seasonal data.

The leaf area could be modulated with space, time, season. The space approach is replaced with the environmental data sets Worldclim (refer section 2.1.6) [23], Terraclim (refer section 2.1.6) [24], SoilGrids (refer section 2.1.6) [25] and Harmonised world soil data (refer section 2.1.6). The time approach is replaced by the change of climate model simulations (CMIP5, refer section 2.1.6) [17] and the season approach is replaced by the day of year.

2.1.1 Extracting data from Idigbio and Gbif

Idigbio and Gbif are the open databases, the data and images of millions of herbar specimens are being conserved and made available in digital format for the scientific research community. From this huge databases, the herbar specimen data is being processed to run the analysis based on leaf area. In addition to the herbar specimen storage URLs, the information of latitude,

longitude, date collected, scientific name, institution code, catalogue-number and other information (not important to this project) is also available.

The data had been analysed by excluding the grass species from the total data available from Gbif and Idigbio, the preprocessing and download routine for Idigbio and Gbif metadata is shown in figure 2.1. All the required data after excluding the unnecessary data was stored in a pickle file and further, this metadata pickle file is used to process and edit the data in the next steps.

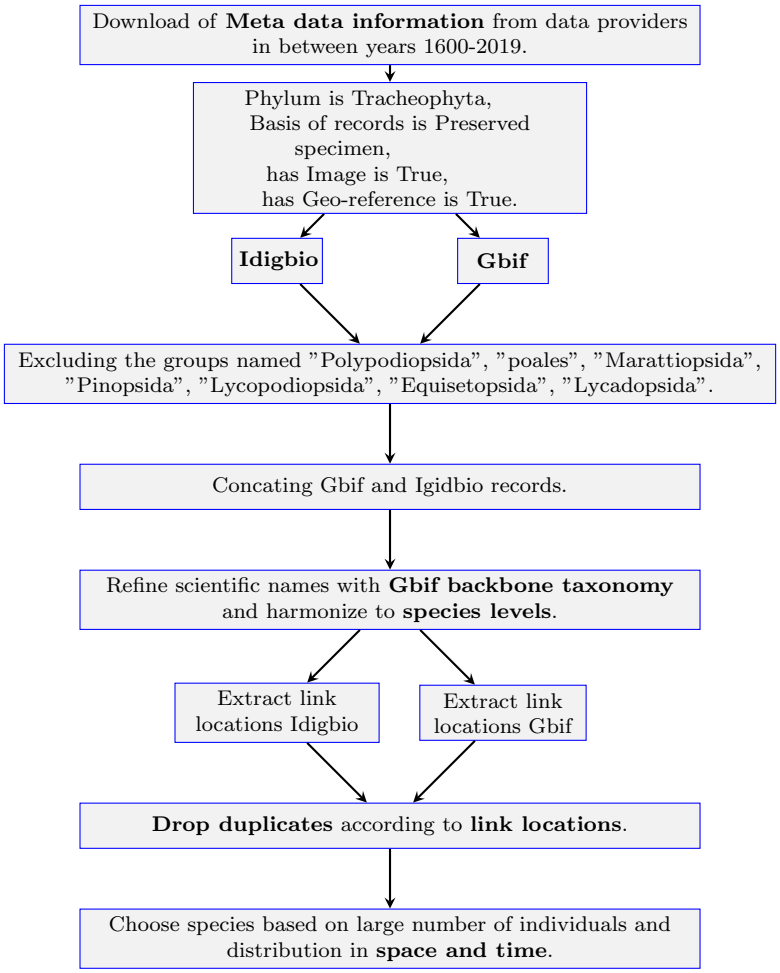


Figure 2.1: Flowchart for processing the Idigbio and Gbif meta data on scanned herbar specimen

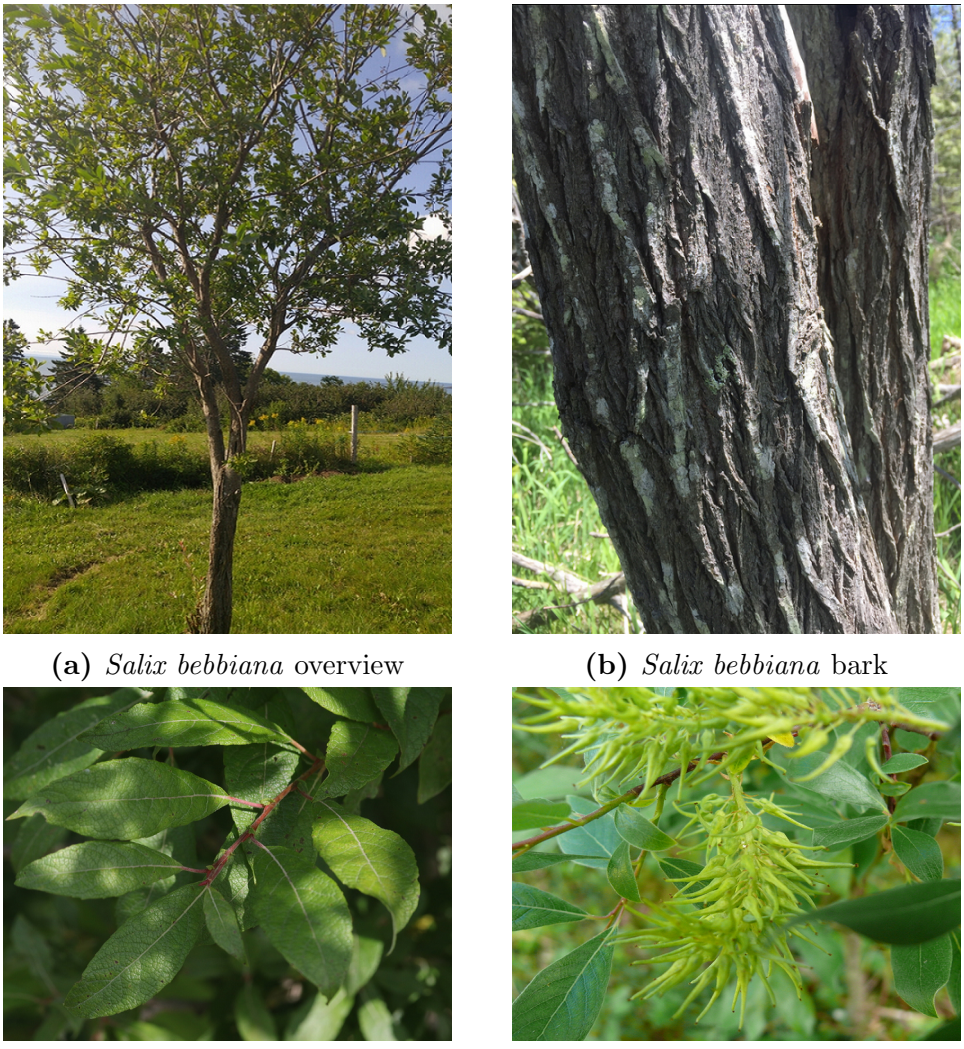
2.1.2 Selected species

After downloading all the available metadata from the databases Idigbio and Gbif, the metadata pickle file is available as discussed in section 2.1.1. The study species *Salix bebbiana*, *Alnus incana* and *Viola canina* were selected based on certain criteria as discussed in section 1.1. By using the columns date collected, latitude and longitude from the metadata pickle file, the following figures (figure 2.3, figure 2.5, figure 2.7) are created for three species *Salix bebbiana*, *Alnus incana* and *Viola canina*. The histograms (figure 2.3a, figure 2.5a, figure 2.7a) shows that the data

is available also from pre-industrial times and it should be possible to observe climate change effects.

Salix bebbiana

Salix bebbiana is commonly known as Bebb willow and it is a large shrub 10 feet (three meter) tall or a small multi-stemmed tree with a bushy top 15 to 25 feet (4.6 to 7.6 m) tall [27]. Normally, the largest mature leaves of *Salix bebbiana* are 2.6 to 6 inches (6.6 to 15 cm) long and the bark is smooth when young but becomes rough and furrowed when it is old enough.



(a) *Salix bebbiana* overview

(b) *Salix bebbiana* bark

(c) *Salix bebbiana* leaves

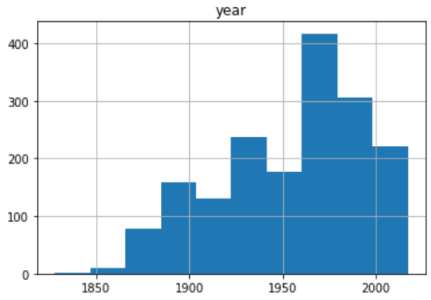
(d) *Salix bebbiana* fruits

Figure 2.2: *Salix bebbiana* detailed overview

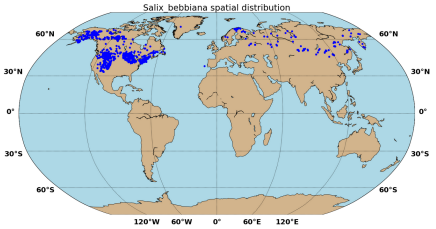
Salix bebbiana occurs from Newfoundland west to Hudson Bay and across Canada to the Yukon Territory and interior Alaska. It extends south to southeastern Alaska, British Columbia, the mountains of Washington, central California, Arizona, New Mexico, and Wyoming, through western Nebraska, Montana, and south and east from North Dakota and South Dakota to the northeastern United States. *Salix bebbiana* is usually found on moist sandy or

gravelly soils but is adapted to a wide variety of soil textures. It will tolerate moderately alkaline soils but does poorly in extremely acidic or alkaline conditions [27].

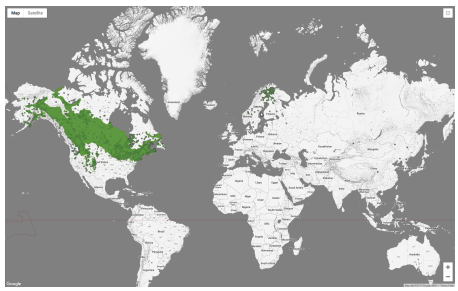
The figure 2.3a is the histogram for the available data (metadata pickle file) based on the year. Maximum herbar specimens are collected around the year 1970 for the species *Salix bebbiana*. *Salix bebbiana* species distribution around the globe based on available data (metadata pickle file) is showed in the figure 2.3b and spatial distribution of the *Salix bebbiana* based on total data is showed in the figure 2.3c. The global maps for the spatial distribution of total data are taken from the Map of Life [28] databases. This shows that the herbar specimens cover the species distribution well.



(a) *Salix bebbiana* histogram from Idigbio and Gbif



(b) *Salix bebbiana* saptial distribution from Idigbio and Gbif



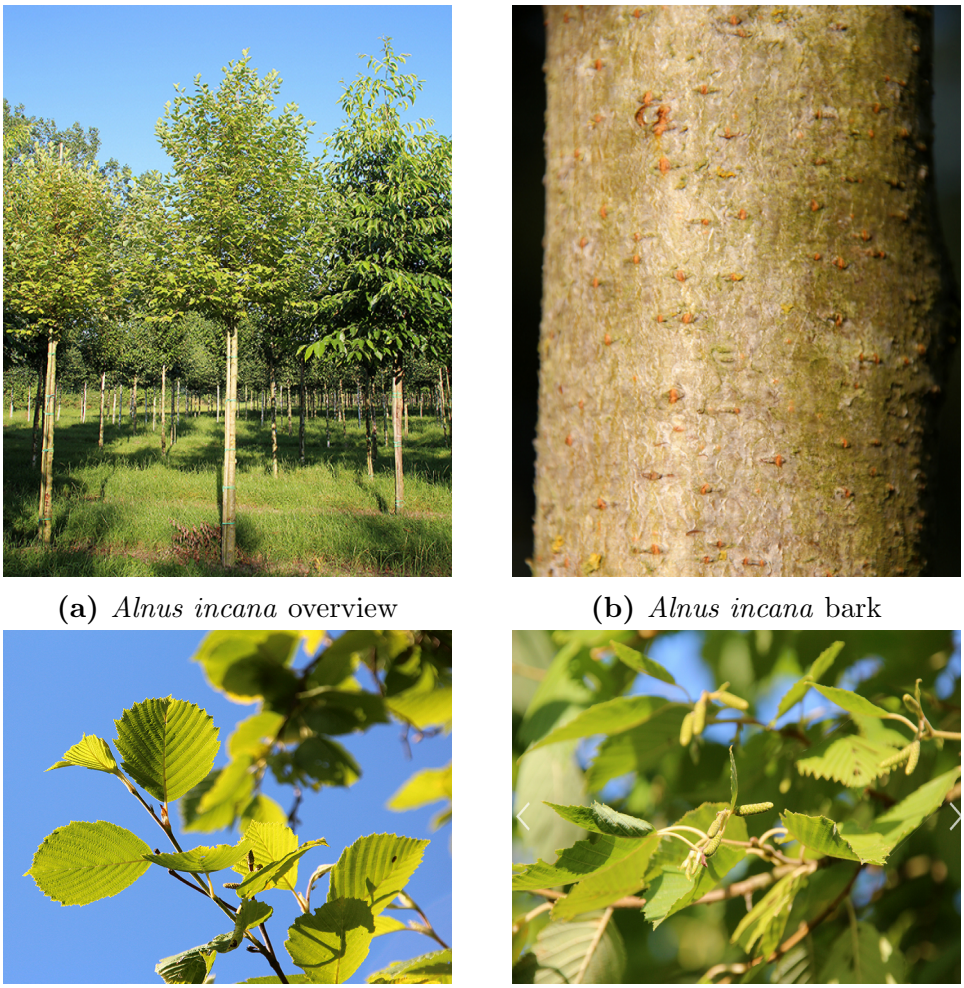
(c) *Salix bebbiana* total spatial distribution from map of life [28]

Figure 2.3: *Salix bebbiana* spatial and temporal distributions

Alnus incana

Alnus incana is also known as the grey alder and speckled alder and it is a species of tree in the birch family which is spread across the cooler parts of the Northern Hemisphere [29]. It usually grows from 15 feet to 82 feet (4.6-25 m) tall and the wood is soft, leaves are oblong and serrated at the margins. In the ecological context, *Alnus incana* also plays an important role to eradicate the afforestation. These can be achieved by planting the *Alnus incana* on non-fertile soils and it enriches the nitrogen-fixing bacteria in its root nodules [29].

As discussed in section 2.1.2 the figures 2.5a, 2.5b and 2.5c are the histogram of the data available, the spatial distribution of *Alnus incana* based on available



(a) *Alnus incana* overview

(b) *Alnus incana* bark

(c) *Alnus incana* leaves

(d) *Alnus incana* fruits

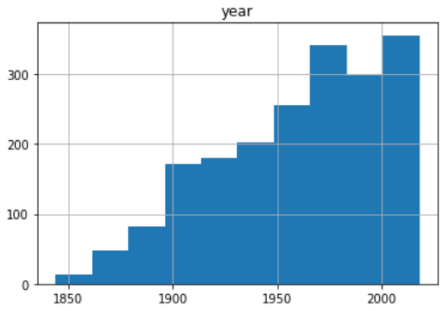
Figure 2.4: *Alnus incana* detailed overview

data (metadata pickle file) and the *Alnus incana* spatial distribution based on total data (data from map of life [28]) respectively.

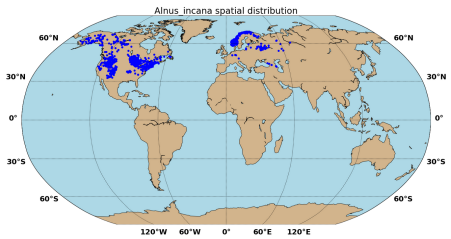
Viola canina

Viola canina is commonly known as heath dog-violet or heath violet is native to Europe and is a species of the genus Viola. it is a herb, usually growing to 5 to 15 cm tall. The leaves of *Viola canina* are much smaller compared to other genus types of Viola. *Viola canina* is a species with a wide range of distribution in the Northern temperate region of the Old World from North-Eastern Siberia westwards to South Greenland and from Lapponia south to the Mediterranean area [30].

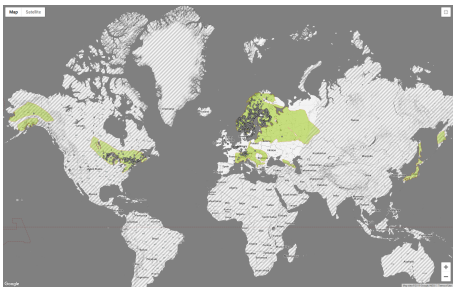
As discussed in the section 2.1.2 the figures 2.7a, 2.7b and 2.7c are the histogram of the data available, the spatial distribution of *Viola canina* based on available data (metadata pickle file) and the *Viola canina* spatial distribution based on total data (data from map of life [28]) respectively.



(a) *Alnus incana* histogram from Idigbio and Gbif



(b) *Alnus incana* spatial distribution from Idigbio and Gbif



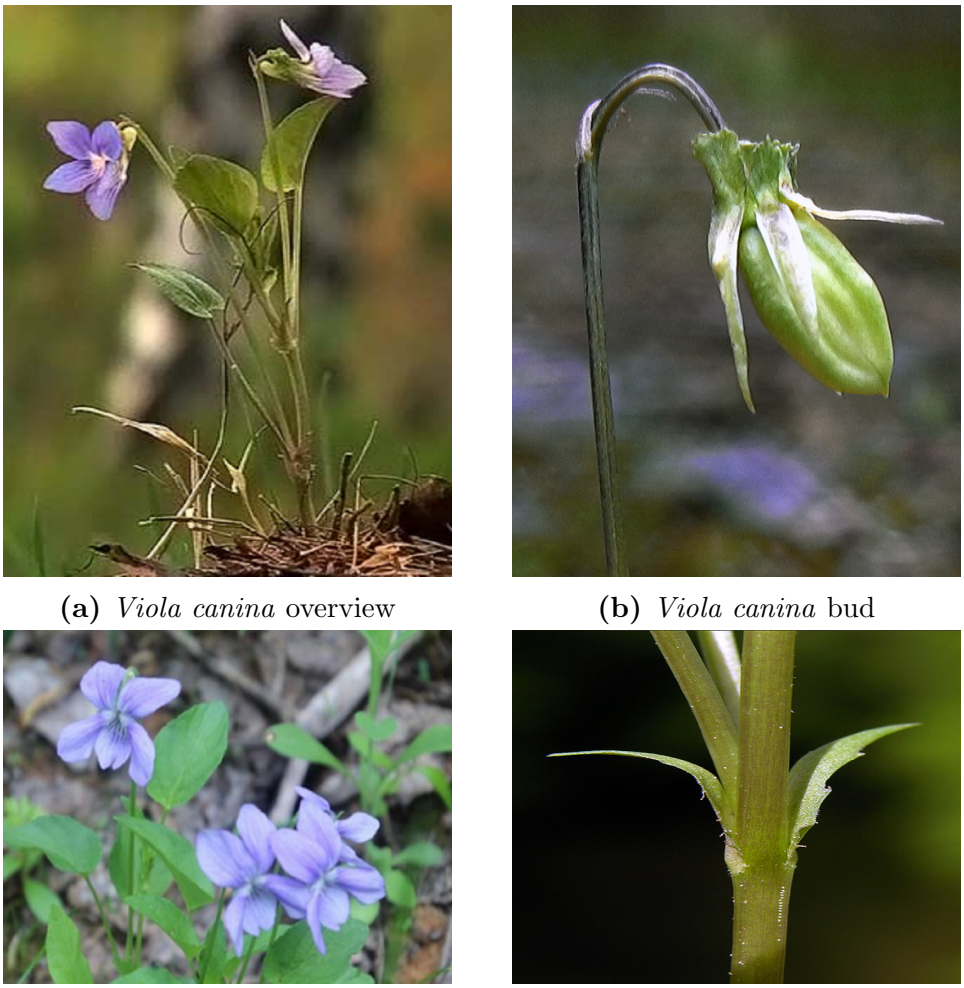
(c) *Alnus incana* total spatial distribution from map of life [28]

Figure 2.5: *Alnus incana* spatial and temporal distributions

2.1.3 Download of images

There are hundred's of herbar specimen URLs available in metadata pickle file and each image has to be named uniquely to track and review the specific herbar specimen later. So, the column named sample_id had been created in metadata pickle file by using the institution code and catalog-number. The sample_id may contain the duplicates because the institution code and catalog-number are same for some samples (herbar specimens) so another column named image_id (Unique column) is needed and it had been created by enumerating the sample_id. Now, every entry of the image_id column had a unique name and these names can serve as the name for the folders and the name for the images.

The images had been downloaded using the URLs provided by the Idigbio and the Gbif databases (metadata pickle file) and stored in a well-organised way (Figure A.1) to access the images easier with Trait Ex. The Python function (refer section A.1) is developed in such a way that if one of the URL is not responding then it will skip the specific URL and continues with the next URL and at the same time the Python function resizes the image that is stored in the specific URL with specified width (1900 pixels) and height (2400 pixels). The objective workflow for downloading images, measuring the herbar specimen leaves with Trait Ex and merging csv files from Trait Ex output is shown in figure 2.8



(a) *Viola canina* overview **(b)** *Viola canina* bud

(c) *Viola canina* flowers **(d)** *Viola canina* stem

Figure 2.6: *Viola canina* detailed overview

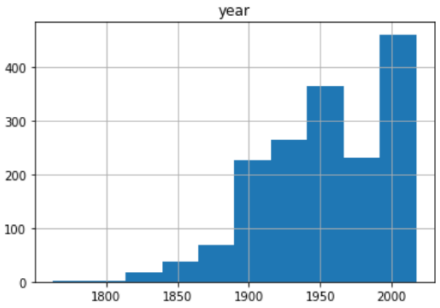
2.1.4 Description of Trait Ex workflow

Graphical interface of Trait Ex

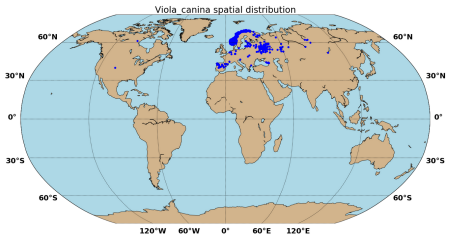
Trait Ex has different functionalities which are spread over three menus arranged across the top of the desktop [19]: Project, Tools, Help. The visualization of a graphical user interface of the Trait Ex is shown in figure 2.9.

In figure 2.9, section one represents a menu bar, section two represents work space panel, section three represents a preview panel and section four represents measurement panel [19]. The purpose of the four sections are elaborated below:

1. Menu Bar: Provide access to all Trait Ex Functions.
2. Work space Panel: Provide access to all measured specimen images.
3. Preview Panel: Display a preview on the current specimen image to be measured.
4. Measurement Panel: Display a table containing the quantitative trait values for each measured leaf.



(a) *Viola canina* histogram from Idigbio and Gbif



(b) *Viola canina* spatial distribution from Idigbio and Gbif

(c) *Viola canina* total spatial distribution from map of life [28]

Figure 2.7: *Viola canina* spatial and temporal distributions

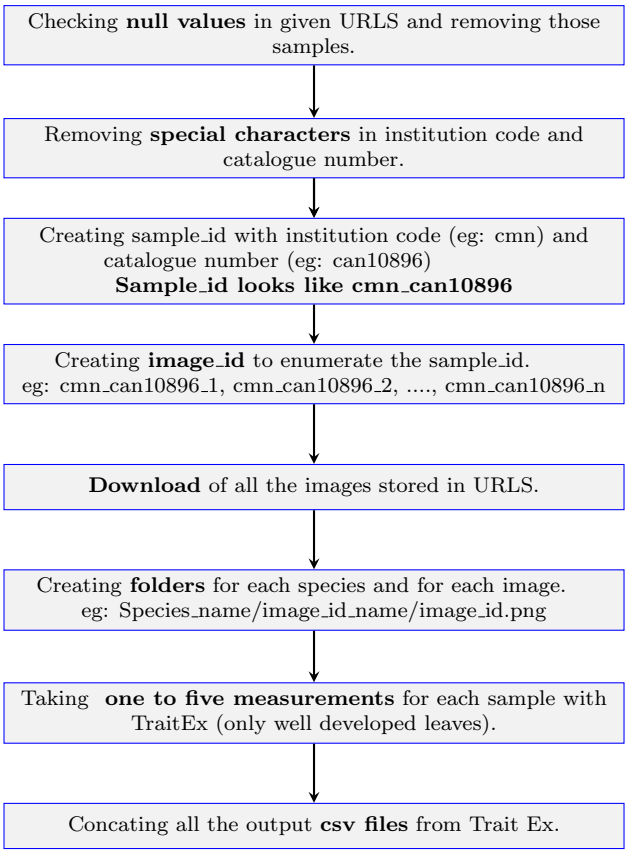


Figure 2.8: Flowchart for downloading the images

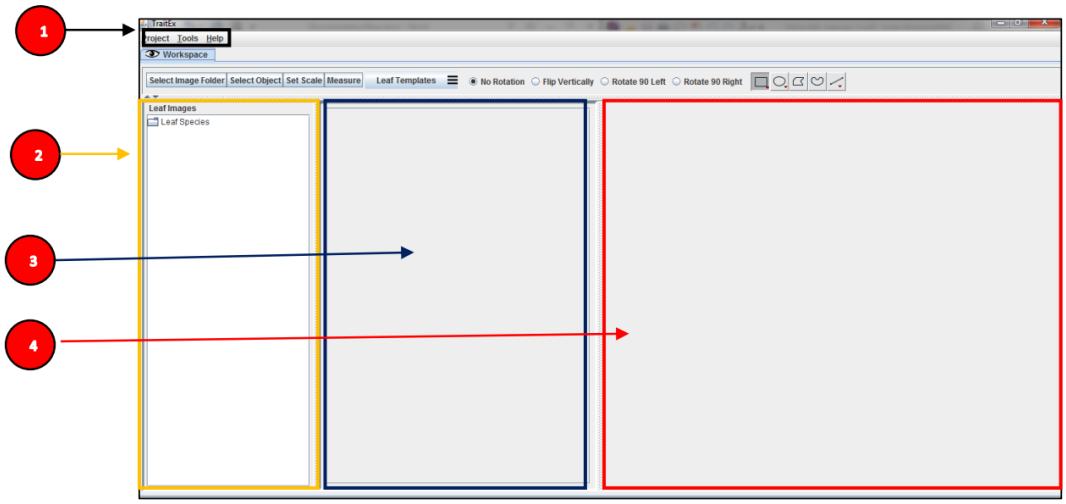


Figure 2.9: Graphical interface of Trait Ex

Basic functionality of Trait Ex

One can measure the leaf area, leaf perimeter, leaf length, leaf width, petiole length with Trait Ex. Now the basic and necessary steps to measure the leaves within the herbar specimen [19] are shown below.

Step 1: Selecting image folder Using "Select Image Folder" button in the left-hand corner of Trait Ex, one can select a folder containing the pre-selected and resized digitized specimen images (Figure 2.10) [19].

Figure 2.10: Button for selecting image folder in Trait Ex

After selecting the input folder, all digitized specimen images are loaded in the work space as shown in the figure below (Figure 2.11).

After loading all digitized specimen images in the work space, one can select each image separately and then, the selected image will be displayed in the preview panel as shown in the following image (Figure 2.12).

Step 2: Set the Scale The second step in the quantitative trait measurements pipeline is setting the scale and units. The process involves calibrating a single image against known values, then applying that calibrated image to the unknown image [19].

Here one has to click on "Select Object" button to open a new instance of the selected specimen image (Figure 2.13).

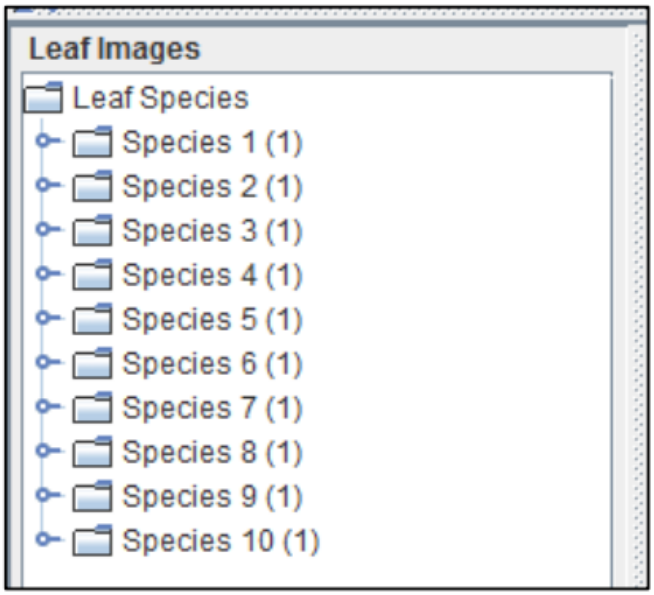


Figure 2.11: Preview of loaded images in Trait Ex

Figure 2.12: Preview of selected image in Trait Ex

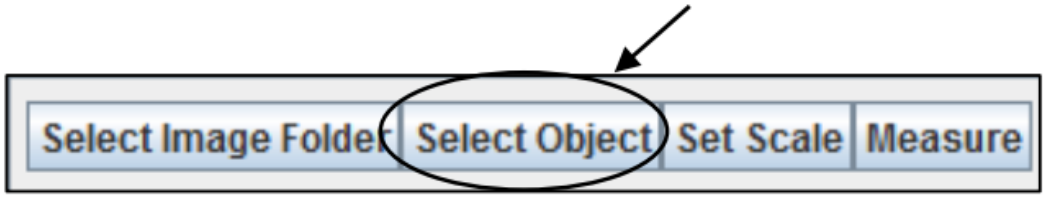


Figure 2.13: Button for opening the current image to set the scale

One can select the straight-line tool in the right-hand corner of Trait Ex to draw a line along the scale bar of known length (10 cm had to be taken). Then, one has to search for the mentioned scale (known length) in the digitized herbar specimen. After that, one has to select "Set Scale" from the menu [19] (Figure 2.14).

Here one can click on "Set Scale" button from the menu. The Plot Scale dialog box will open. Here, one can check if the scaling process was done perfectly or not

Figure 2.14: Selecting a straight line tool for the menu bar of Trait Ex

[19] (Figure 2.15).

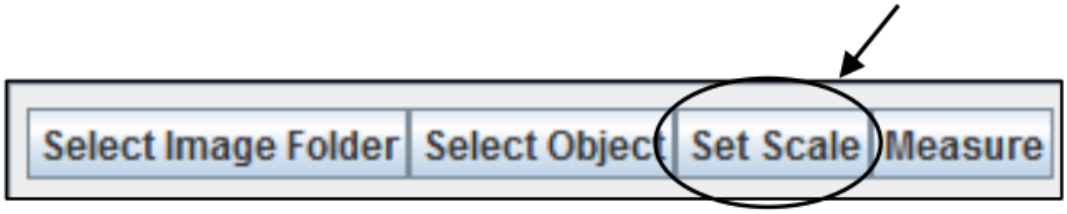


Figure 2.15: Button for setting the scale in Trait Ex

Note that the process of scaling is applied for each selected specimen image loaded in the work space. Once this step is done for a specimen image, it is not required to redo this work again during the measuring process of selected specimen [19] .

Step 3: Select the leaf to be measured After setting the scale, one can measure the quantitative traits.

Select the image using "Select Object" button from the menu to open again a new instance of the selected specimen image. Select the leaf of interest to be measured using the rectangular, polygon, or freehand selection tools [19] (available next to straight line tool) (Figure 2.16).

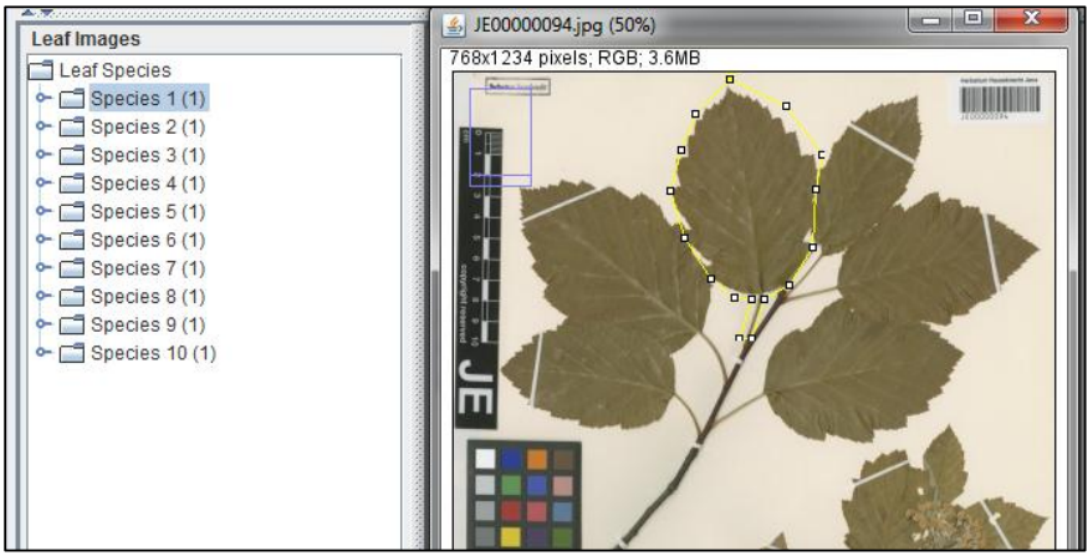


Figure 2.16: Select the leaf of interest after loading the images in Trait Ex

Click on "Measure" button from the menu to measure the quantitative traits of the selected leaf (Figure 2.17). The measured traits will be displayed as a table in the measurement panel. Moreover, a red bounding box will be drawn in the preview panel to make it easy for the user to know which leaf is already measured [19] (Figure 2.18).

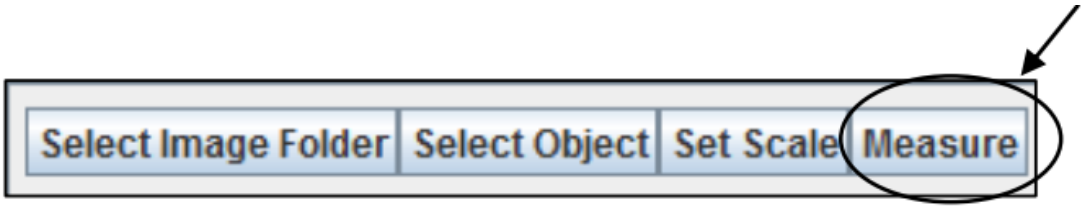


Figure 2.17: Button for measuring the selected leaf in Trait Ex

Figure 2.18: Measurements for the selected leaf in the herbar specimen

Note that the leaf orientation influences the reliability of Trait Ex tool. To obtain good results [19], the leaf should be fixed as follow:

1. The petiole should be located at the bottom
2. Whereas the blade is oriented upwards.

If the selected leaf does not meet these criteria, a set of option (shown below) is setting up in Trait Ex to resolve such issues (Figure 2.19).

- **Flip vertically**: the output image will be rotated 90 degrees.
- **Rotate 90 Left**: rotates the entire image counter-clockwise 90 degrees.
- **Rotate 90 Right**: rotates the entire image clockwise 90 degrees.

Figure 2.19: Options for changing the leaf orientation in Trait Ex

2.1.5 Effective usage and data organization of Trait Ex

I used Trait Ex to extract leaf size values for the species *Salix bebbiana*, *Alnus incana* and *Viola canina* from 8355 digital herbar specimens using Trait Ex workflow (refer section 2.1.4). Different aspects of output files and data

organization (special cases of herbar specimens, data limitations of herbar specimens based on compatibility of Trait Ex and how hundred's of output csv files are merged?) are discussed below.

Overview of output files (Trait Ex)

The images are saved as discussed in section 2.1.3. Now, if the number of measured leaves in a herbar specimen is five, then there are more than 23 output files which will be stored in the specific sub-directory. For each measured leaf, there are some output files as discussed below:

Trait Ex output csv files There are always three csv output files, first one is the main csv file, this file is common for all and it is updated if the leaf is measured within a specific herbar specimen. The second file contains only individual measurements, the third file is for initial raw measurements. If the number of measured leaves is three, then the main csv will look like as shown in figure 2.20. These csv files contains a summary of all measurements associated with the respective image.

	ImageID	LeafID	type	length	width	area	perimeter	circularity	parent
0	/home/vkommin/Downloads/Alnus_incana/trh_17192...	1	leaf	4.312404	3.159438	9.409671	15.967219	0.007016	NaN
1	NaN	2	petiole	0.468084	NaN	NaN	NaN	NaN	1.0
2	/home/vkommin/Downloads/Alnus_incana/trh_17192...	3	leaf	4.291341	3.427615	11.143362	16.360351	0.007908	NaN
3	NaN	4	petiole	0.628346	NaN	NaN	NaN	NaN	3.0
4	/home/vkommin/Downloads/Alnus_incana/trh_17192...	5	leaf	3.053474	2.459995	5.640956	11.279872	0.008422	NaN
5	NaN	6	petiole	0.709135	NaN	NaN	NaN	NaN	5.0

Figure 2.20: Visualization of main csv file from output files of Trait Ex

Trait Ex output image files There are two output image files, first one is the cropped image of the measured leaf as shown in figure 2.21 and the second one is the mask of the cropped image as shown in figure 2.22.

Figure 2.21: Cropped view of measured leaf after the measurement in Trait Ex

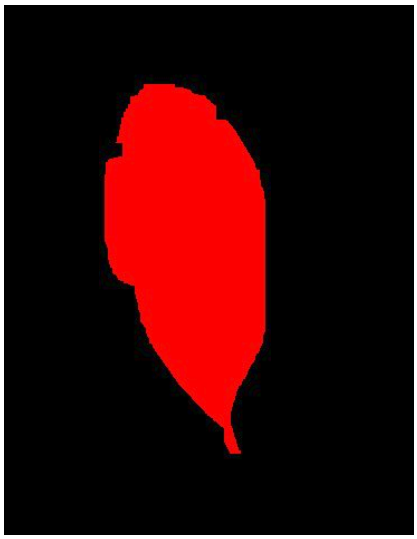


Figure 2.22: Mask of the measured leaf after the measurement in Trait Ex

Trait Ex output text files There are two output text files, first one contains some number regarding the scale and the second file contains the mapping points of the measured leaf.

Available data based on different aspects

Table 2.1 shows the available images after and before the measurement of leaf sizes of herbar specimens with Trait Ex, the total number of leaves measured with the Trait Ex and the total number of features added to the specific species. The reason behind the decrease of 20 to 40 per cent of data points after the measurement of leaf sizes of herbar specimens with Trait Ex are due to some special cases. The special cases are the presence of juvenile leaves (figure 2.23a), saplings (figure 2.23b), no images (figure 2.23c) in herbar specimen, no measurable leaves in herbar specimen (overlapped leaves (figure 2.23d) and no perfect petiole (figure 2.23e)) and no scale (figure 2.23f) available in herbar specimen.

As mentioned above, for the special cases, the excel sheet (contains the information like URL, sample_id and image_id of each species from metadata pickle file) is maintained and updated while measuring the leaf sizes of the herbar specimen with Trait Ex. This excel sheet is useful to eradicate the special cases automatically using some machine learning techniques for future species.

Combining Trait Ex results per species

After measuring all the leaf sizes of available herbar specimens in the specified species with Trait Ex, the main csv's from all the sub-directories (refer figure A.1) has to be appended to obtain the Trait Ex data frame from single csv files. After some data processing (removing every second row (contains petiole length data) in the Trait Ex data frame and adding this information as a separate column) as shown in the python code (refer section A.2), the Trait Ex data frame and the

Species name	Total number of image URLs available before download	Total number of images remained after measured with Trait Ex	Total number of features added	Total number of leaves measured with Trait Ex
Salix bebbiana	3230	1744	78	5934
Alnus incana	2871	1950	78	5104
Viola canina	2308	1675	78	3683

Table 2.1: Available data of the species based on different aspects

metadata pickle file (data frame) is merged based on the Image_id column.

Let us call the merged data frame as the main data frame form now, the main data frame has the data from Idigbio, Gbif and the original leaf trait values from Trait Ex data frame. From now, the data from different data sets are going to be merged to the main data frame and then the machine learning algorithm comes into the play. One can see the number of features added to the metadata frame according to the specific species selected in table 2.1.

2.1.6 Extracting features from different databases

As discussed in section 2.1, the leaf area could be modulated by the environment, climate change and season. To find which feature data set is influencing the leaf area, different features are extracted from various databases as discussed below.

Features representing the spatial variability of the environment

To represent the spatial variability of the environment, I used several databases containing interpolated measurements of current climate and soil. Latitude (-90 to 90 in degrees) and longitude (-180 to 180 in degrees) are the common dimensions for all the environmental features. The current climate data represent mean values of today's environment and refer to measurements of the years 1970-2000. Soil data have assumed constant over time. All the environmental features are appended to the main data frame using latitude and longitude.

WorldClim WorldClim [23] is a data set of spatially interpolated monthly climate data for global land areas at a very high spatial resolution (approximately 1 km^2). Worldclim data set included monthly temperature (minimum, maximum and average), precipitation, solar radiation, vapour

(a) juvenile leaves (in downloaded images)

(b) Saplings (in downloaded images)

(c) No image (in downloaded images)

(d) Overlapped leaves (in downloaded images)

(e) No perfect petiole (in downloaded images)

(f) No scale (in downloaded images)

Figure 2.23: Special cases of digital herbar specimen (downloaded images)

pressure and wind speed, aggregated across a target temporal range of 1970–2000, using data from between 9000 and 60,000 weather stations, another 19 bioclimatic features [31] are also available in worldclim data set and calculated using the available features. The detailed overview of the features available in the data set is shown in table 2.2.

Table 2.2: WorldClim data specifications.

Feature long name	Feature short name	Units	Spatial resolution

To be continued

Table 2.2 – *Continued from previous page*

Feature long name	**Feature short name**	**Units**	**Spatial resolution**
Annual Mean Temperature	BIO1	°C	1km
Mean Diurnal Range (Mean of monthly (max temp - min temp))	BIO2	°C	1km
Isothermality (BIO2/BIO7) (* 100)	BIO3	No units	1km
Temperature Seasonality (standard deviation *100)	BIO4	No units	1km
Max Temperature of Warmest Month	BIO5	°C	1km
Min Temperature of Coldest Month	BIO6	°C	1km
Temperature Annual Range (BIO5-BIO6)	BIO7	°C	1km
Mean Temperature of Wettest Quarter	BIO8	°C	1km
Mean Temperature of Driest Quarter	BIO9	°C	1km
1Mean Temperature of Warmest Quarter	BIO10	°C	1km
Mean Temperature of Coldest Quarter	BIO11	°C	1km
Annual Precipitation	BIO12	mm	1km
Precipitation of Wettest Month	BIO13	mm	1km
Precipitation of Driest Month	BIO14	mm	1km
Precipitation Seasonality (Coefficient of Variation)	BIO15	mm	1km
Precipitation of Wettest Quarter	BIO16	mm	1km
Precipitation of Driest Quarter	BIO17	mm	1km
Precipitation of Warmest Quarter	BIO18	mm	1km
Precipitation of Coldest Quarter	BIO19	mm	1km

To be continued

Table 2.2 – *Continued from previous page*

Feature long name	**Feature short name**	**Units**	**Spatial resolution**
Average temperature	Tavg	°C	1km
Solar radiation	Srad	$kJm^{-2}day^{-1}$	1km

Terraclim TerraClim [24] is a data set of monthly climate and climatic water balance for global terrestrial surfaces from 1958-2015. These data provide important inputs for ecological and hydrological studies at global scales that require high spatial resolution and time-varying data. All data has monthly temporal resolution and a 4-km (1/24th degree) spatial resolution.

The available temporal domain of terraclim is from 1958 to 2015 but the data from 1970 to 2000 (mean) had been processed in this project because available temporal data for worldclim is 1970 to 2000. The common features in the worldclim and the terraclim are omitted from the worldclim data set, this means the remaining features of both climate data sets (worldclim and terraclim) are completely complementary (table 2.3). The features from terraclim and worldclim are processed similar to the features extraction of Soil Grids database (refer section A.3).

Table 2.3: Terraclim data specifications.

Feature long name	**Feature short name**	**Units**	**Spatial resolution**
Maximum temperature	Tmax	°C	4km
Minimum temperature	Tmin	°C	4km
Vapour pressure	vap	kPa	4km
Precipitation accumulation	ppt	mm	4km
Downward surface shortwave radiation	srad	Wm^{-2}	4km
Wind speed	ws	ms^{-1}	4km
Potential evapotranspiration	pet	mm	4km
Runoff	q	mm	4km
Actual evapotranspiration	aet	mm	4km

To be continued

Table 2.3 – *Continued from previous page*

Feature long name	Feature short name	Units	Spatial resolution
Climate water deficit	def	mm	4km
Soil moisture	soil	mm	4km
Snow water equivalent	swe	mm	4km
Palmer drought severity index	pdsi	unitless	4km
Vapor pressure deficit	vpd	kPa	4km

SoilGrids database SoilGrids database is used to extract the soil features, it is the global digital soil mapping that makes use of global soil profile information and covariate data to model the spatial distribution of soil properties across the globe [25]. The SoilGrids predictions are made at six different depths (top depths are $0, 5, 15, 30, 60$ and 100 cm), only the data for top four depths are processed in this project. The weighted mean (based on depths) (refer section A.3) for top four depths (till 30cm) had been calculated per feature (table 2.4).

Table 2.4: Soil Grids data specifications.

Feature long name	Feature short name	Units	Spatial resolution
Grade of a sub-soil being acid e.g. having a pH < 5 and low BS	ACDWRB	Grade	10km
Available soil water capacity (volumetric fraction) with FC = pF 2.0	AWCh1	percentage	10km
Available soil water capacity (volumetric fraction) with FC = pF 2.3	AWCh2	percentage	10km
Available soil water capacity (volumetric fraction) with FC = pF 2.5	AWCh3	percentage	10km
Saturated water content (volumetric fraction) for teta-S	AWCtS	percentage	10km

To be continued

Table 2.4 – *Continued from previous page*

Feature long name	**Feature short name**	**Units**	**Spatial resolution**
Depth to bedrock (R horizon) up to 200 cm	BDRICM	cm	10km
Probability of occurrence of R horizon	BDRLOG	percentage	10km
Absolute depth to bedrock	BDTICM	cm	10km
Bulk density (fine earth)	BLDFIE	kg/m^3	10km
Cation Exchange Capacity of soil	CECSOL	cmolc/kg	10km
Weight percentage of the clay particles (<0.0002 mm)	CLYPPT	percentage	10km
Volumetric percentage of coarse fragments (>2 mm)	CRFVOL	percentage	10km
Histosols probability cumulative	HISTPR	percentage	10km
Soil organic carbon density	OCDENS	kg/m^3	10km
Soil organic carbon stock	OCSTHA	ton/ha	10km
Soil organic carbon content	ORCDRC	permille	10km
pH index measured in water solution	PHIHOX	pH	10km
pH index measured in KCl solution	PHIKCL	pH	10km
Sodic soil grade	SLGWRB	grade	10km
Weight percentage of the silt particles (0.0002–0.05 mm)	SLTPPT	percentage	10km
Weight percentage of the sand particles (0.05–2 mm)	SNDPPT	percentage	10km
Texture class (USDA system)	TEXMHT	factor	10km
Available soil water capacity (volumetric fraction) until wilting point	WWP	percentage	10km

Harmonized world soil database Harmonized world soil data is another soil data set which had similar features like SoilGrids but used different methods in data modelling. The harmonized world soil features (table 2.5) are also processed to compare with the Soil Grids database.

Table 2.5: Harmonized world soil data specifications.

Feature long name	Feature short name	Units	Spatial resolution
Available water capacity	AWC	mm	1km
Depth of obstacles to roots	ROOTS	Coded values 0 through 6	1km
Subsoil bulk density	S_BULK _DENSITY	$kgdm^{-3}$	1km
Cation exchange capacity of the clay fraction in the subsoil	S_CEC _CLAY	cmol per kg	1km
Subsoil CEC soil	S_CEC _SOIL	cmol per kg	1km
Subsoil clay fraction	S_Clay	% weight	1km
Subsoil gravel content	S_Gravel	% volume	1km
Subsoil organic carbon	S_OC	% weight	1km
Subsoil pH (in water)	S_PH _H2O	$-\log(H+)$	1km
Subsoil sand fraction	S_Sand	% weight	1km
Subsoil silt fraction	S_Slit	% weight	1km
Topsoil bulk density	T_BULK _DENSITY	$kgdm^{-3}$	1km
Cation exchange capacity of the clay fraction in the topsoil	T_CEC _CLAY	cmol per kg	1km
Topsoil CEC soil	T_CEC _SOIL	cmol per kg	1km
Topsoil clay fraction	T_Clay	% weight	1km
Topsoil gravel content	T_Gravel	% volume	1km
Topsoil organic carbon	T_OC	% weight	1km

To be continued

Table 2.5 – *Continued from previous page*

Feature long name	**Feature short name**	**Units**	**Spatial resolution**
Topsoil pH (in water)	T_PH _H2O	−log(H+)	1km
Topsoil sand fraction	T_Sand	% weight	1km
Topsoil silt fraction	T_Slit	% weight	1km

Additional features There are some further features which may create a great influence on leaf area. The additional features (table 2.6) are also appended to the main data frame using latitude and longitude columns. In addition to this, the data from the digital elevation model (altitude around the globe) is also appended to the main data frame.

Draught stress index is calculated (refer section A.5) as actual evapotranspiration (Terraclim) divided by potential evapotranspiration (Terraclim). Seasonal amplitude of climate water deficit (Terraclim) and palmer drought severity index (Terraclim) are calculated as annual maximum value minus annual minimum value of a specific feature (refer section A.5). Latent heat, Latent heat mean absolute deviation, Net radiation, Net radiation mean absolute deviation, Day time land surface temperature and Night time land surface temperature are calculated from the satellite data of the specific year 2001 (refer section A.5).

Features representing temporal variability due to climate change

To represent the temporal variability of the climate (climate change), I used the projected data from the climate model (MIROC-ESM) available via the CMIP5 [17]. CMIP5 [17] is Coupled Model Inter-comparison Project phase 5. The CMIP5 model projections are less reliable compared to the measured database like CRU [32], however, the measured data is not available for years before 1880. The CMIP5 data has to be analyzed from 1600 on-wards because the herbar specimen samples are available from the year 1600.

CMIP5 data contains three different time phases: past1000 [33] (before 1850 gridded as yearly), historic (1850-2005 gridded as daily), current (2005-present gridded as hourly) data sets. Irrespective of the available time-frequency, the monthly gridded data is processed. The detailed overview and the data processing steps are shown in the python code (refer section A.4). Since all the data of CMIP5 are downloaded as NetCDF (n-dimensional arrays) files, there are some interesting details like spatial resolution, short name of feature, long name of feature and units are shown in table 2.7.

Feature long name	Feature short name	Units	Spatial resolution
Draught stress index	NA	NA	4km
Seasonal amplitude of climate water deficit	NA	mm	4km
Seasonal amplitude of palmer drought severity index	NA	unit less	4km
Latent heat	LE	$MJm^{-2}d^{-1}$	10km
Latent heat mean absolute deviation	LE_mad	$MJm^{-2}d^{-1}$	10km
Net radiation	Rn	$MJm^{-2}d^{-1}$	10km
Net radiation mean absolute deviation	Rn_mad	$MJm^{-2}d^{-1}$	10km
Day Time Land Surface Temperature	LST_Day	K	10km
Night Time Land Surface Temperature	LST_Night	K	10km

Table 2.6: Additional features data specifications

Feature long name	Feature short name	Units	Spatial resolution
Daily maximum near surface air temperature	tasmax	k	300km
Daily minimum near surface air temperature	tasmin	k	300km
Near surface air temperature	tas	k	300km
Precipitation	pr	$kg/m^2/s$	300km
Surface Down-welling short radiation	rsds	Wm^{-2}	300km
Surface temperature	ts	K	300km
Wind speed	sfcwind	ms^{-1}	300km
Atmospheric water vapour content	prw	kgm^{-2}	300km

Table 2.7: CMIP5 data specifications

In addition to the data specifications specified in table 2.7, there are some common dimensions for all the features like latitude (-90 to 90 in degrees), longitude (0 to 360 in degrees) and the available time (850 to 2100 in year). The climate change data set had been created by calculating the anomalies of climate compared to today's climate as the difference between the CMIP5 data and the mean of CMIP5

data is calculated for the years 1970 to 2000 (because WorldClim had temporal domain only for 1970 to 2000). The feature values from the climate change data set is appended to the main data frame using the latitude, longitude and the date collected columns of all the herbar specimens.

Seasonal features

To represent the seasonal effect, I used day of year which is calculated from the date collected column for all the available data (metadata pickle file).

Finally, the main data frame contains the data from Idigbio metadata, Gbif metadata, Trait Ex data frame data, anomalies of CMIP5 data, Worldclim data, Terraclim data, SoilGrids data, Harmonized world soil data, Digital elevation model and day of year.

2.2 Building a machine learning model to predict the leaf size from features related to environment, climate change and season

Why Random Forests? Random Forests are selected because of its sheer advantages over the other machine learning models. There might be a chance that we may have some non-informative features in our model because of a lot of climatic and soil data features. Most of the times, Random Forests is the best choice to deal with non-informative data because it accommodates less importance to non-informative features. This model deals with both categorical and continuous features but, moreover, it creates hundred's of decision trees to avoid over-fitting and increase model accuracy. This is a robust method and working well with standard hyper parameters.

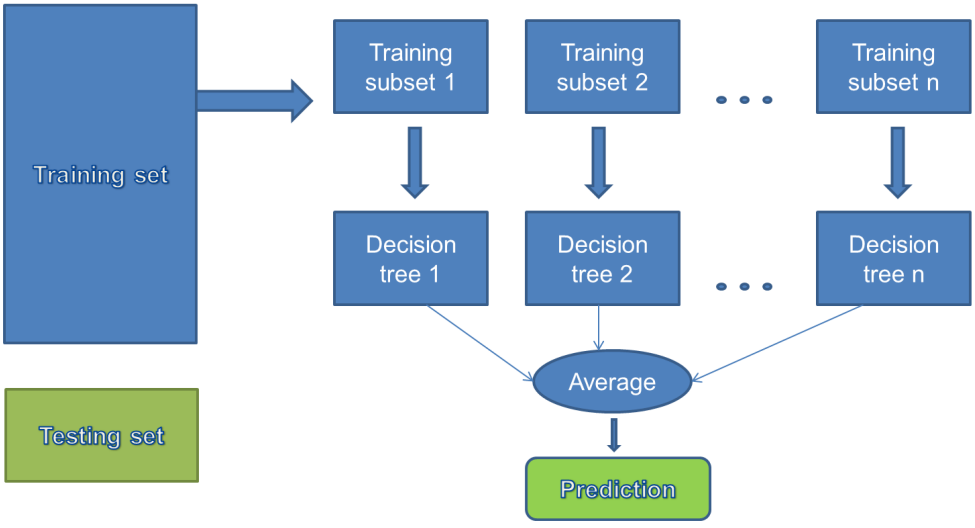


Figure 2.24: Explaining Random Forest

How Random Forest regression works? Random Forests consist of several hundreds of decision trees (figure 2.24), each of them builds on a random subset of observations and features from the data set. Not every node sees all the features or all the observations, and this guarantees that the trees are de-correlated and therefore less prone to over-fitting. Each tree is also a sequence of yes-no questions based on a single or combination of features. At each node (this is at each question), the tree divides the data set into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket [34]. Figure 2.25 is one of the sample tree (only up to three depths) from the Random Forest and it is generated using the real data (main data frame).

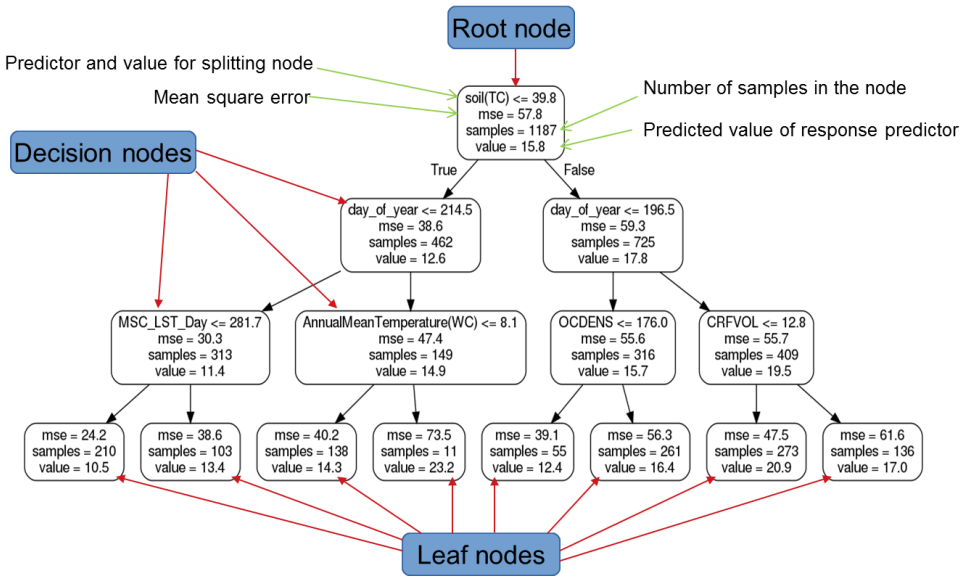


Figure 2.25: Explaining Random Forest tree

Why Partial Least Square regression? Partial Least Square regression is used to see whether the data have non-linear relationships or not. The main advantages of Partial Least Square regression are no particular feature selection is needed and run time is faster compared to the Random Forest model especially when working with larger data sets. The detailed explanation of Partial Least Square regression is given in the URL http://www.eigenvector.com/Docs/Wise_pls_properties.pdf.

How Partial Least Square regression works? Partial Least Square regression reduces the features to a smaller set of uncorrelated features and performs Least Square Regression on these reduced set of features. Partial Least Square regression is a good method to use when the features are highly correlated or when the data have more features than observations. Partial Least Square regression does not assume that the features are fixed, this means that the features can be measured with error, making Partial Least Square regression more robust to measurement uncertainty.

2.2.1 Missing value strategy

Random Forest Imputation (MissForest) [35] had been used to fill the missing values in the main data frame. MissForest imputes the missing values using the Random Forests in an iterative fashion [35] as discussed below.

- This method starts imputing the missing values with the column contains the smallest number of missing values, let us call this as candidate column.
- In the second step, all the missing values of remaining non-candidate columns are filled with column mean for numerical columns and column mode for categorical values.
- After that, the imputer fits a Random Forest model with the candidate column as the response feature and the remaining columns as the features overall rows where the candidate column values are not missing, then the missing values of the candidate column are filled with the prediction of the Random Forest.
- Following this, the imputer moves to the next candidate column with the second smallest number of missing values among the non-candidate columns in the first round.
- This procedure repeats itself to fill all the missing values in the given data set.

2.2.2 Removing outliers

Removing outliers is not required for the data used in this thesis because every herbar specimen leaf area is measured with hand by using the semi-automated tool Trait Ex. Even though, outliers are removed by using the IQR (Inter Quartile Range) (refer figure 2.26) strategy to avoid the human-made mistakes and Trait Ex mistakes (with special cases (refer figure 2.23)). The equation to remove all the possible outliers is shown below.

$$\text{Data without outliers} = (\text{Data} > (Q1 - 1.5{*}IQR))\&(\text{Data} < (Q3 + 1.5{*}IQR))$$

Where,
Q1 = Median of the n smallest values (first 25% of data).
Q3 = Median of the n largest values (last 25% of data).
IQR = Q3-Q1.

99 data points of *Salix bebbiana*, 78 data points of *Alnus incana* and 70 data points of *Viola canina* are removed from the total data points as outliers.

2.2.3 Creating customized block cross validation

Customized block cross-validation is dividing the given data into different sets based on the block strategies. This block cross-validation is required when the

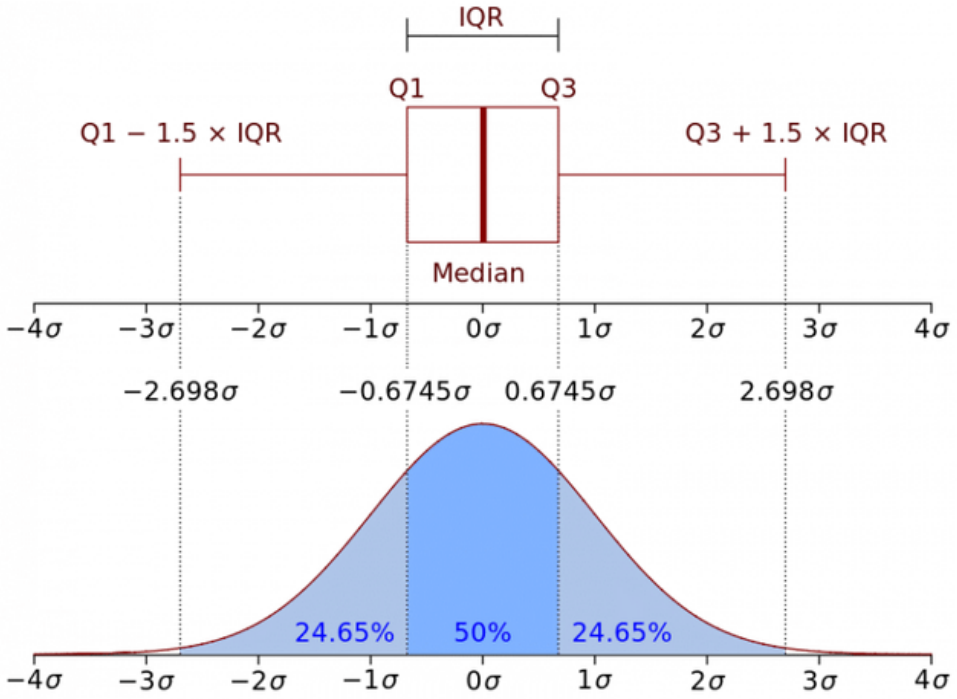


Figure 2.26: Representation of the Interquartile Range

data came from latitude and longitude positions (data points are dependant on each other) or to divide the seasonal effects on sales.

If two data points came from nearby spatial location then there might be a chance that one data point could go into training data set and other might go into testing data set. In this case R^2 is not realistic because same kind of information is already seen in training data set. To avoid all these kind of sampling biases, customized block cross-validation is required. There might be a chance that the herbar specimens used in these analysis are collected from nearby spatial locations because each herbar specimen is collected by different botanists around the globe. In the main data frame, data is divided into blocks as discussed below.

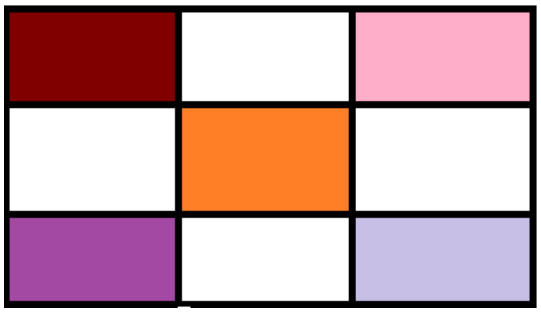


Figure 2.27: 3*3 Spatial blocking

As discussed in the section A.6 every data point is designated with specific block number 0 to 8 based on latitude and longitude position. This block number is used to split the final data frame into 9 folds, so the data from each fold is independent to other fold, this ensures that all the herbar specimens are independent to each other.

For example, if 9 blocks (figure 2.27) are taken from the 360∗180 virtual matrix (as discussed in section A.6), each block in the figure 2.27 must be filled with different number in between 0 to 8.

2.2.4 Working with different combinations of available data sets

To understand which data sets are helping the most for predicting leaf area, different combination of data sets are experimented to get the insights of the combinations using benchmark model. The Normal R^2 value (using 70% of the data set for training and 30% of the data set for testing), customised block cross validation (refer section 2.2.3) R^2 value, Out of bag score ($2/3^{rd}$ data is used as training and $1/3^{rd}$ as testing) etc are shown in tables 2.8 and 2.9.

The terminology of the tables 2.8 and 2.9 are explained below:

- PLS regression = Partial Least Square regression.
- R^2 = R^2 calculated using 70% of data as training set and 30% as testing set.
- R^2_{train} = Train score for the data (70% train and 30% test data).
- Oob_{scor} = Oob score is calculated using total data with Random Forest regressor.
- R^2_{bs} = R^2 calculated using customised block cross validation (refer section 2.2.7).
- HWSD1 = HWSD + TC + WC + Cmip + Cmipmd + Cmipac + doy + DEM,
- HWSD2 = HWSD + WC + TC + Cmip + doy + DEM,
- HWSD3 = HWSD + WC + TC + Cmipmd + doy + DEM,
- HWSD4 = HWSD + WC + TC + Cmipac + doy + DEM,
- HWSD5 = HWSD + WC + TC + Cmip + Cmipmd + doy + DEM,
- HWSD6 = HWSD + WC + TC + Cmip + Cmipac + doy + DEM,
- HWSD7 = HWSD + WC + TC + Cmipmd + Cmipac + doy + DEM,
- SD1 = SD + TC + WC + Cmip + Cmipmd + Cmipac + doy + DEM,
- SD2 = SD + WC + TC + Cmip + doy + DEM,
- SD3 = SD + WC + TC + Cmipmd + doy + DEM,
- SD4 = SD + WC + TC + Cmipac + doy + DEM,
- SD5 = SD + WC + TC + Cmip + Cmipmd + doy + DEM,
- SD6 = SD + WC + TC + Cmip + Cmipac + doy + DEM,
- SD7 = SD + WC + TC + Cmipmd + Cmipac + doy + DEM,

Dataset	Random Forests				PLS regression		
	R^2	R^2_{train}	Oob_{scor}	R^2_{bs}	R^2	R^2_{train}	R^2_{bs}
HWSD1	0.2763	0.6816	0.2727	0.2506	0.0952	0.3197	0.1533
HWSD2	0.2797	0.6707	0.2688	0.2447	-0.0244	0.3125	0.1469
HWSD3	0.2799	0.6707	0.2685	0.2449	-0.0244	0.3125	0.1469
HWSD4	0.2770	0.6807	0.2731	0.2486	0.2055	0.3042	0.1560
HWSD5	0.2798	0.6706	0.2685	0.2442	-0.0132	0.3112	0.1540
HWSD6	0.2763	0.6817	0.2729	0.2504	0.1227	0.3202	0.1487
HWSD7	0.2763	0.6818	0.2733	0.2510	0.1227	0.3202	0.1487
SD1	0.2723	0.6984	0.2732	0.2491	0.1554	0.3395	0.1813
SD2	0.2729	0.6922	0.2723	0.2438	0.1893	0.3285	0.1823
SD3	0.2737	0.6923	0.2730	0.2436	0.1893	0.3285	0.1823
SD4	0.2730	0.6963	0.2695	0.2470	0.1497	0.3256	0.1657
SD5	0.2733	0.6922	0.2725	0.2434	0.1841	0.3282	0.1895
SD6	0.2728	0.6982	0.2726	0.2483	0.1479	0.3411	0.1761
SD7	0.2725	0.6983	0.2731	0.2487	0.1479	0.3411	0.1761
TcWc_out	0.2534	0.6811	0.2691	0.2321	0.1816	0.2967	0.1839
Soil_out	0.2760	0.6801	0.2743	0.2560	0.1863	0.3134	0.1991
SD1doy	0.2774	0.6841	0.2228	0.2096	0.1251	0.3265	0.1550
SD2doy	0.2764	0.6797	0.2306	0.2013	0.1391	0.3149	0.1539
SD3doy	0.2768	0.6796	0.2305	0.2015	0.1391	0.3149	0.1539
SD4doy	0.2707	0.6751	0.2049	0.1822	0.0889	0.2706	0.0887
SD5doy	0.2759	0.6797	0.2307	0.2009	0.1361	0.3139	0.1543
SD6doy	0.2777	0.6841	0.2228	0.2091	0.1230	0.3278	0.1510
SD7doy	0.2777	0.6842	0.2232	0.2090	0.1230	0.3278	0.1510

Table 2.8: Possible R^2 values for *Salix bebbiana*

Dataset	Random Forests				PLS regression		
	R^2	R^2_{train}	Oob_{scor}	R^2_{bs}	R^2	R^2_{train}	R^2_{bs}
HWSD1	0.1605	0.6282	0.2094	0.1466	0.1304	0.2310	0.1384
HWSD2	0.1628	0.5981	0.1979	0.1415	0.1324	0.2204	0.1504
HWSD3	0.1628	0.5981	0.1979	0.1415	0.1324	0.2204	0.1504
HWSD4	0.1559	0.6151	0.2061	0.1470	0.1191	0.2330	0.1360
HWSD5	0.1572	0.6051	0.2030	0.1416	0.1392	0.2191	0.1502
HWSD6	0.1612	0.6209	0.2098	0.1460	0.1289	0.2313	0.1422
HWSD7	0.1612	0.6209	0.2098	0.1460	0.1289	0.2313	0.1422
SD1	0.1715	0.6325	0.2044	0.1549	0.1359	0.2438	0.1520
SD2	0.1688	0.6093	0.1998	0.1527	0.1305	0.2323	0.1499
SD3	0.1688	0.6093	0.1998	0.1527	0.1305	0.2323	0.1499
SD4	0.1682	0.6180	0.2086	0.1543	0.1228	0.2433	0.1464
SD5	0.1776	0.6221	0.2068	0.1529	0.1361	0.2302	0.1499
SD6	0.1747	0.6280	0.2078	0.1546	0.1424	0.2437	0.1500
SD7	0.1747	0.6280	0.2078	0.1546	0.1424	0.2437	0.1500
TcWc_out	0.1768	0.6193	0.1986	0.1680	0.1414	0.2171	0.1499
Soil_out	0.1644	0.6215	0.2032	0.1473	0.1254	0.2229	0.1469
SD1doy	0.1571	0.6215	0.1919	0.1467	0.1384	0.2408	0.1582
SD2doy	0.1421	0.5920	0.1775	0.1513	0.1329	0.2282	0.1529
SD3doy	0.1421	0.5920	0.1775	0.1513	0.1329	0.2282	0.1529
SD4doy	0.1442	0.6020	0.1835	0.1453	0.1315	0.2410	0.1555
SD5doy	0.1581	0.6054	0.1887	0.1514	0.1372	0.2270	0.1509
SD6doy	0.1548	0.6095	0.1953	0.1466	0.1402	0.2404	0.1556
SD7doy	0.1548	0.6095	0.1953	0.1466	0.1402	0.2404	0.1556

Table 2.9: Possible R^2 values for *Alnus incana*

- TCWC_out = SD + Cmip + Cmipmd + Cmipac + doy + DEM,
- Soil_out = TC + WC + Cmip + Cmipmd + Cmipac + doy + DEM,
- SD1doy = SD + TC + WC + Cmip + Cmipmd + Cmipac + DEM,
- SD2doy = SD + WC + TC + Cmip + DEM,
- SD3doy = SD + WC + TC + Cmipmd + DEM,
- SD4doy = SD + WC + TC + Cmipac + DEM,
- SD5doy = SD + WC + TC + Cmip + Cmipmd + DEM,
- SD6doy = SD + WC + TC + Cmip + Cmipac + DEM,
- SD7doy = SD + WC + TC + Cmipmd + Cmipac + DEM.

Where,

HWSD = Harmonized world soil database,

TC = Terra clim model,

Wc = World clim model,

Cmip = Coupled Model Intercomparison Project 5,

Cmipmd = Cmip mean deviation,

Cmipac = Cmip actual values (Data set created by using CMIP5 data and with the models of worldclim [31]),

SD = SoilGrids data,

doy = day of year,

DEM = Digital Elevation Model.

I decided to use Random Forest models for all my analysis because in the tables 2.8 and 2.9, R^2 values of Partial Least Square regression are consistently lower compared to R^2 values of Random Forest models. Within Random forest models, the ${R^2}_{bs}$ values are consistently lower compared to R^2 values. Generally, ${R^2}_{bs}$ values and normal R^2 values should be near to each other, if data points are independent of each other. In the case of spatial autocorrelation (refer section 1.1), ${R^2}_{bs}$ is lower than normal R^2. Therefore, the block selection cross-validation model is selected because it gives the realistic R^2 values and this model is prone to overfitting and bias of the data.

I decided to use data set SD3 = SD + WC + TC + Cmipmd + doy + DEM because Cmipac and Cmip data is not needed since Cmipmd is representing both (Cmipac and Cmip) and data set SD3 had better R^2 values compared to other data sets in tables 2.8 and 2.9.

2.2.5 Feature selection method

To acquire the most contributed features (from total feature set) to the response feature, two feature selection methods are used, they are Recursive feature elimination with cross-validation (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html) and Guided hybrid genetic algorithm for feature selection [36]. After comparing the results of both methods, the feature set selected using the guided hybrid genetic algorithm had more explanatory power (table 2.10) compared to the recursive feature elimination with cross-validation. Moreover, recursive feature elimination with cross-validation method selected double the number of optimum features compared to guided hybrid genetic algorithm. Because of the mentioned reasons, the guided hybrid genetic algorithm for feature selection using block cross-validation had been selected and all the results are compiled using it.

2.2.6 Hyperparameter tuning

Hyperparameter tuning is tuning the parameters of the specified model (here Random Forest regressor) to get the best possible explanatory power for given data. To tune hyperparameters, one has to create the random set of hyperparameters and RandomizedSearchCV or https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html will search for the best parameters from the given random set of hyperparameters. I have used this method to search for the best parameters to gain more explanatory power (block cross-validation is used).

2.2.7 Model Evaluation

In contrast to the basic settings used in the benchmark model, Customised cross validation model gets the model parameters using parameter tuning (refer section 2.2.6) method and the custom block cross-validation [22] (refer section 2.2.3) is used to avoid spatial auto-correlation (refer section 1.1) and to ensure the herbar specimens are independent of each other. The block cross-validated R^2 values for the selected species are shown in table 2.10.

The R^2 (Coefficient of determination) is calculated using the below equation [37].

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Where,
$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$
$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$
$\hat{y}_i$ = Predicted value of the i-th sample.
y_i = corresponding true value for total n samples

Species name	Block cross validated R^2 value using RFECV	Block cross validated R^2 value using GHGAFS
Salix bebbiana	0.2444	0.2592
Alnus incana	0.1804	0.1981
Viola canina	0.1059	0.1405

Table 2.10: Comparison of block cross validated R^2 values for all selected species using different methods. Where, RFECV = Recursive feature elimination with cross-validation, GHGAFS = Guided hybrid genetic algorithm for feature selection.

2.2.8 Presentation of results

Feature importance

Feature importance plot is plotted using the scikit-learn Feature_importances_ package. Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of samples that reach the node, divided by the total number of samples. The higher the value the more important the feature. The feature importance value is calculated using the following equation [38].

$$RFfi_i = \frac{\sum_{j \in all\ \text{trees}} normfi_{ij}}{T}$$

where,
$RFfi_i$ = The importance of feature i calculated from all trees in the Random Forest model.
$normfi_{ij}$ = Normalized feature importance for i in tree j.
T = Total number of trees.

The higher the $RFfi_i$, the more important is the feature.

Partial dependency plots (PDP plots)

When using black-box machine learning algorithms like Random Forest, it is hard to understand the relations between features and model outcome [39]. In Random Forest regression it is possible to calculate feature importance score and it is known which feature is significantly influencing the outcome based on the importance calculation and it is not possible to know in which direction the feature is influencing concerning response feature.

To visualize the relation between the influencing features with respect to response feature, The PDP box (https://github.com/SauceCat/PDPbox) is used and it is

A	B	C	Y
A1	B1	C1	Y1
A2	B2	C2	Y2
A3	B3	C3	Y3

(a) Sample data set

A	B	C	Y
A1	B1	C1	Y11
A1	B2	C2	Y21
A1	B3	C3	Y31
A2	B1	C1	Y12
A2	B2	C2	Y22
A2	B3	C3	Y32
A3	B1	C1	Y13
A3	B2	C2	Y23
A3	B3	C3	Y33

(b) Data set combinations

A	B	C	Y	mean
A1	B1	C1	Y11	Y(A1)
A1	B2	C2	Y21	
A1	B3	C3	Y31	
A2	B1	C1	Y12	Y(A2)
A2	B2	C2	Y22	
A2	B3	C3	Y32	
A3	B1	C1	Y13	Y(A3)
A3	B2	C2	Y23	
A3	B3	C3	Y33	

(c) Mean values of prediction

X	A1	A2	A3
Y	Y(A1)	Y(A2)	Y(A3)

(d) Final data used for plotting

Figure 2.28: Different stages of sample data set while plotting PDP plots

based on the following equation.

$$\hat{f_{x_S}} = \frac{1}{N} \sum_{i=1}^{N} \hat{f}\left(x_S, x_{C_i}\right)$$

Where, S = Response feature,
C = Complete feature set other than response feature,
N = Total number of observations in the training set.

For example, let us assume a data set that only contains three data points and three features (A, B, C) as shown in figure 2.28a [39]. If feature A is influencing the prediction Y, what PDP does is to generate a new data set (figure 2.28b) and do prediction as usual (here the assumption is that feature A has three unique values: A1, A2, A3). In this way, PDP will generate the number of predictions (figure 2.28b) and averaged them for each unique value of feature A (2.28c). Finally, PDP would only plot out the average predictions (2.28d) [39].

Predict observed plot

It is the plot used to see how well the data points are predicted compared to the observed data. With the help of the predict observed plot, one can visualize the prediction errors. The predict observed plot is plotted based on Random Forest model with block cross-validation using cross-validated predictions (https://scikit-learn.org/0.18/auto_examples/plot_cv_predict.html) function from Scikit-learn.

Chapter 3

Results

3.1 Comparison of Random Forests with Partial Least Square regression

The block cross-validated R^2 values for the Random Forest model is quite high compared to the Partial Least Square regression model for all the selected species (refer tables 2.8 and 2.9). This tells us that there are considerable non-linear relationships within the data.

3.2 Goodness of fit for Random Forest model per species

Salix bebbiana peaks the block cross-validated R^2 values with 0.259 compared to *Alnus incana* and *Viola canina* (refer table 3.1). The prediction errors of the model (figures 3.1b, 3.2b and 3.3b) are quite high for all the selected species (*Salix bebbiana*, *Alnus incana* and *Viola canina*).

3.3 Predictors on leaf size per species

The leaf sizes of three species (*Salix bebbiana*, *Alnus incana* and *Viola canina*) are highly influenced by all the selected feature data sets current environment, season and climate change. However, season plays a vital role (figures 3.1a, 3.2a and 3.3a) for all the species and climate change is most influencing for the species *Viola canina*.

Table 3.1: R^2 values for final Random Forest model

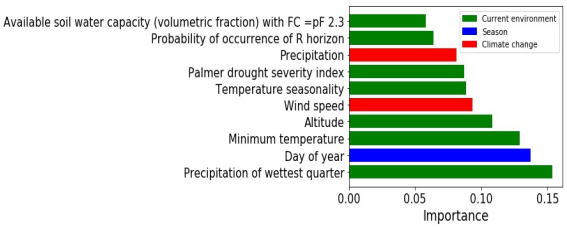
Species name	R^2 **value**
Salix bebbiana	0.2592
Alnus incana	0.1981
Viola canina	0.1405

It is very clear from the results (figures 3.1d, 3.2c and 3.3c) that the leaf areas of all three species are increasing with the day of year.

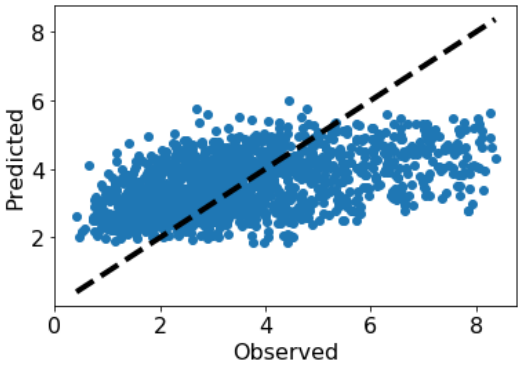
The leaf area is increasing with minimum temperature (figure 3.1e) and night time land surface temperature (figure 3.2f). But surprisingly, at the same time, the leaf area is decreasing with surface temperature (figure 3.3d) and it is increasing with mean temperature of warmest quarter (figure 3.3e) for the species *Viola canina*. The relations for remaining features with leaf area are discussed below.

- Leaf area increases with precipitation of wettest quarter (figure 3.1c) and decreases with altitude (figure 3.1f) for the species *Salix bebbiana*.
- Volumetric percentage of coarse fragments (> 2 mm) (figure 3.2e) had a negative relation with leaf area and downward surface shortwave radiation (figure 3.2d) had hump-shaped relation with leaf area for the species *Alnus incana*.
- Draught stress index (figure 3.3f) had a negative relation with leaf area for the species *Viola canina*.

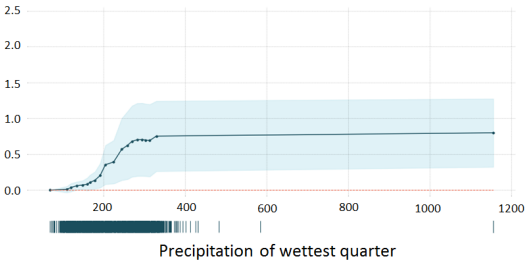
There is a climate change effect for the species *Salix bebbiana* (figure 3.1a) and *Viola canina* (figure 3.3a) but, there is no sign of climate change for the species *Alnus incana* (figure 3.2a).



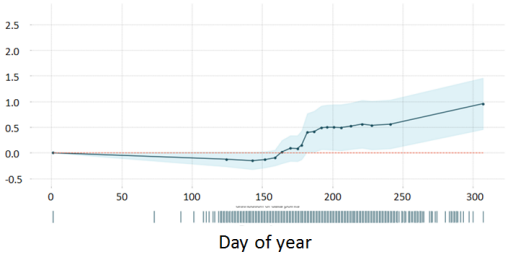
(a) Feature importance plot



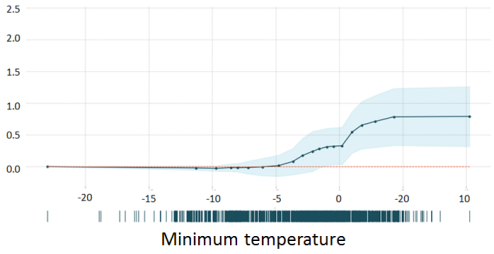
(b) Observed predict plot



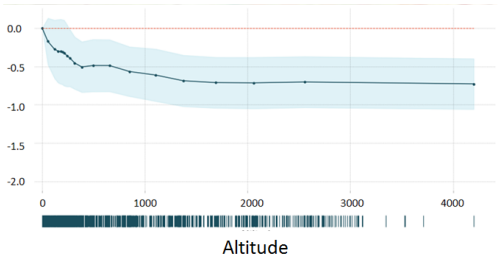
(c) Partial dependency plot (Precipitation of wettest quarter)



(d) Partial dependency plot (Day of year)

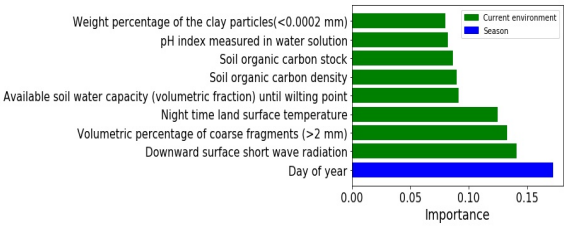


(e) Partial dependency plot (Minimum temperature)

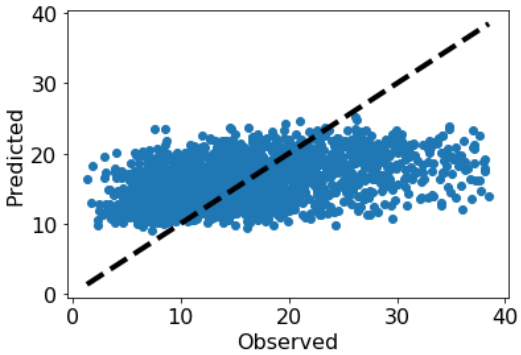


(f) Partial dependency plot (Altitude)

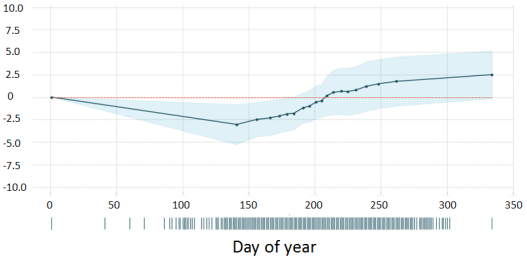
Figure 3.1: *Salix_bebbiana* results



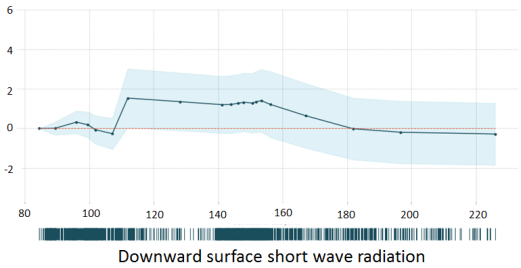
(a) Feature importance plot



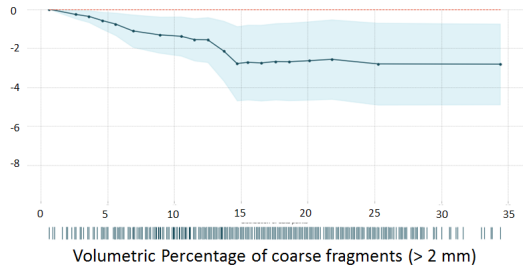
(b) Observed predict plot



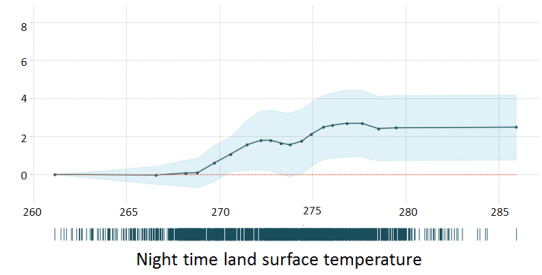
(c) Partial dependency plot (Day of year)



(d) Partial dependency plot (Downward surface shortwave radiation)

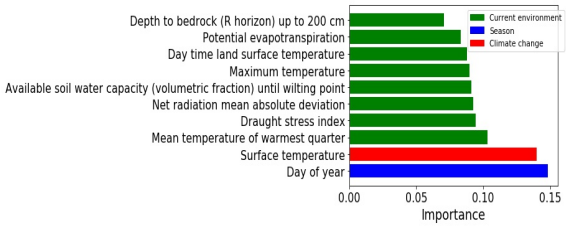


(e) Partial dependency plot (Volumetric percentage of coarse fragments (> 2 mm))

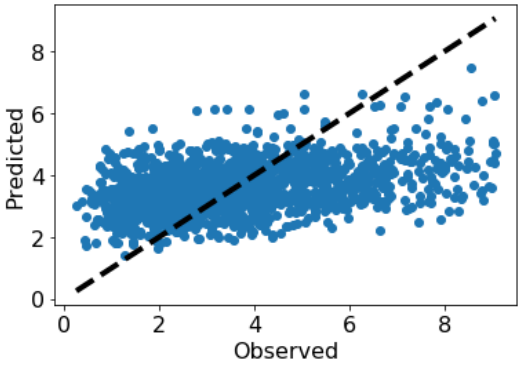


(f) Partial dependency plot (Night time land surface temperature)

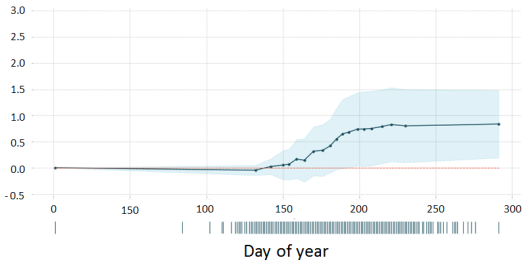
Figure 3.2: *Alnus_incana* results



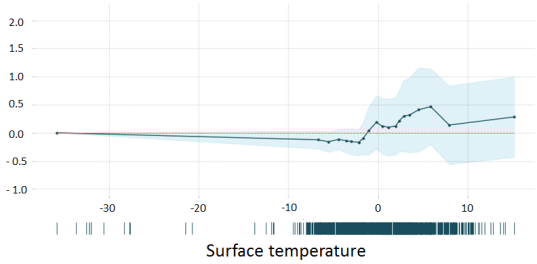
(a) Feature importance plot



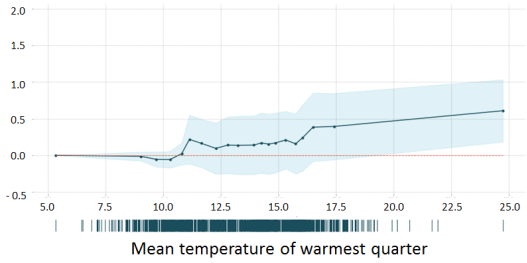
(b) Observed predict plot



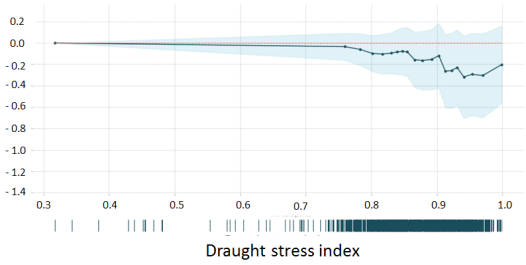
(c) Partial dependency plot (Day of year)



(d) Partial dependency plot (Surface temperature)



(e) Partial dependency plot (Mean temperature of warmest quarter)



(f) Partial dependency plot (Draught stress index)

Figure 3.3: *Viola_canina* results

Chapter 4

Discussion of results

Some of the Trait Ex functions like Trait Ex tab placements (menu bar, workspace panel, preview panel) has been optimized based on my first user experience (refer section 4.0.1). Different combinations of data are created and implemented machine learning techniques (Random Forests and Partial Least Square regression) to get better insights into the data (refer tables 2.8 and 2.9). Secondly, customized block cross validation (refer section 2.2.3) is created to eradicate the sampling bias and spatial auto correlation. In addition to this different feature selection methods (refer section 2.2.5) are used to find the best feature set, hyper parameter tuning (refer section 2.2.6) is used to find best hyper parameters for machine learning techniques and partial dependency plots (refer section 2.2.8) are used to visualize the relationship between the influencing feature and response variable.

4.0.1 Trait Ex first user feedback

Trait Ex is a tremendous tool to extract the leaf sizes from the digital herbar specimens. I found some drawbacks while working with Trait Ex as follows:

- If the digital herbar specimen has a small or no petiole, then Trait Ex is not able to extract correct measurements of leaf size.
- Since we have to load every digital herbar specimen into Trait Ex and have to extract the leaf sizes with hand (refer mapping function in Trait Ex workflow (refer section 2.1.4)), for this reason, Trait Ex is not a recommended tool when working with thousands of images.
- Trait Ex is not compatible to work with the special kind of digital herbar specimens (refer section 2.1.5).
- Trait Ex is mostly useful to extract the leaf sizes for perfect digital herbar specimens (specimen contains accurate visible petiole and accurate visible leaf).
- This tool is only useful to work with some specific species (species with perfect petiole) and not useful for grass type species because there is no perfect visible petiole for them.

Suggestions for further development of Trait Ex Trait Ex has to be optimized to work with all special kinds of herbar specimens (see section 2.1.5) and it has to be automated to work with thousands of digital herbar specimens without human interference. Moreover, the tool needs to be improved to avoid the workflow difficulties (for example loading image folder at once instead of loading each image into the tool) within the tool and to improve it as a user-friendly one.

4.0.2 Features on leaf size

There are so many approaches to analyze the leaf area from the climate [7] [8] and soil data separately and few approaches are available based on both climate and soil data [40]. In contrary to all these approaches the leaf area had been analyzed using spatial, temporal and seasonal data for intraspecific level.

Spatial features

Spatial features played an important role in this analysis because herbar specimens are collected from different spatial conditions (environmental conditions are completely different in each location). We can see that most of the features affecting leaf area are related to temperature and precipitation at the intraspecific level. Leaf area for all the species is affected with temperature related feature and Leaf area increasing with minimum temperature (refer figure 3.1e) for *Salix bebbiana*, increasing with night time land surface temperature (refer figure 3.2f) for *Alnus incana* and increasing with mean temperature of warmest quarter (refer figure 3.3e) for *Viola canina*. Leaf area is positively correlated with the precipitation related features across the selected species (refer figure 3.1c).

Our results have shown similar patterns compared to the previous studies [42] based on herbar specimen for spatial features. The patterns of temperature and precipitation (figure 3.1c) related features are completely inline with Wright et al. [9] results. Remaining spatial features are also supporting the relations within the research papers of Wright et al. [9], S. Lavorel et al. [40]. This shows us that the digital herbar specimens will serve as the real world experimental specimens for further scientific analysis.

Temporal features

I absorbed the effect of temporal features on leaf size analysis for all the species selected, this is very new. Herbs adapt themselves quickly according to the surrounding environment and this could be one of the reasons that surface temperature (climate change feature) is the most influencing feature for the species *Viola canina*((5-15 cm tall)). Leaf area for *Viola canina* is increasing with surface temperature and this trend is very new. But there are some research papers [41] exploring the temporal change of the various specimens based on

flowering times but there are no such studies reveals the temporal trends with leaf area. At the same time, temporal trends are not observed in the species *Alnus incana* because it is not quickly possible to adapt themselves for trees.

Seasonal features

Generally leaves increase their size daily and they reach the large size when they are fully developed and the larger size remains the same until they fall from the tree. According to this theory, the relation between the leaf area and the day of year should remain constant because herbar specimens are collected only when the leaves are fully developed but it has increasing trend as shown in the results (chapter 3). The increasing trend could be possible because ecologists do not know whether the leaf size is fully developed or not, so there is a chance that ecologists may collect the leaves before they developed fully. There are no specific rules when (which stage of leaf area) herbar specimen have to be collected.

4.0.3 Remarks on the extension of this approach to multiple species

We have to optimize methodological limitations (refer section A.7) to extend this approach to multiple species. Trait Ex has to be automated to work extensively with multiple species within a short period. This could be possible coupling the object detection software (helpful to identify the objects in digital herbar specimen) with the Trait Ex leaf extracting process. I have developed the neural networks algorithm that automatically classifies the images based on special kinds of herbar specimens (see section 2.1.5), with the help of this algorithm and some other machine learning techniques it is possible to automate the Trait Ex.

4.1 Future work

We are interested to extend this approach to all the available species from Gbif (refer section 2.1.1) [15] and Idigbio (refer section 2.1.1) [16]. We are interested to extend this topic by adding some potential features like land use data and soil groundwater contamination data. There is a possibility to compare the analysis of leaf area and the analysis of phenology timing from digital herbar specimen. We could show that the species adapt to environmental conditions in space and time, space is more important than time and we can check for space for time analysis also. Finally, We want to analyze interspecific trait variability of leaf areas for all the available species.

4.2 Conclusion

Main objectives of the thesis optimizing Trait Ex and identifying environmental drivers using machine learning techniques for intraspecific trait variability are achieved successfully. The whole methodology works good and Trait Ex becomes very handy to measure the leaf sizes without visiting the actual specimen sites. Environmental databases are helpful to get the data quickly for specific conditions and without these data, it is not possible to combine the trait information with environmental data sets.

Finally, the results are overwhelming and fitting with the research papers [9] [42] [41] [40] based on herbar specimen data and real trait data. This thesis serves as the prototype to analyze interspecific trait variability of leaf areas. Furthermore, one can try to automatize the Trait Ex tool and analyze the trait variability of leaf areas for thousands of species at once.

Bibliography

[1] F. Stuart Chapin, Effects of plant traits on ecosystem and regional processes: A conceptual framework for predicting the consequences of global change, Annals of Botany, 2003, Date accessed: 2019-06-04.

[2] Millennium Ecosystem Assessment, Ecosystems and Human Well-being: Synthesis, 2005, Island Press, Washington, DC. Date accessed: 2019-06-14.

[3] S. Coles, 2018 fourth warmest year in continued warming trend, according to NASA, NOAA – Climate Change: Vital Signs of the Planet, NASA Global Climate Change: Vital Signs of the planet, 2019, Date accessed: 2019-06-04.

[4] V. Masson-Delmotte, P. Zhai, H.O. Pörtner, D. Roberts, J. Skea, P.R. Shukla, A. Pirani, W. Moufouma-Okia, C. Péan, R. Pidcock, S. Connors, J.B.R. Matthews, Y. Chen, X. Zhou, M.I. Gomis, E. Lonnoy, T. Maycock, M. Tignor, and T. Waterfield, Summary for Policymakers. In: Global Warming of 1.5^{o}C. An IPCC Special Report on the impacts of global warming of 1.5^{o}C above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty, IPCC, 2018, Date accessed: 2019-07-24.

[5] T. Baldwin, Interactions of plants with their environment, MaxPlanck Institute for Chemical Ecology, 2010, Date accessed: 2019-08-12.

[6] N. Pérez-Harguindeguy et al, New handbook for standardised measurement of plant functional traits worldwide, Australian Journal of Botany, 1996, Date accessed: 2019-09-18.

[7] L. Poorter, F. Bongers, LEAF TRAITS ARE GOOD PREDICTORS OF PLANT PERFORMANCE ACROSS 53 RAIN FOREST SPECIES, Ecology, 2006, Date accessed: 2019-06-15.

[8] P. De Frenne, B.J. Graae, F. Rodríguez-Sanchez, Latitudinal gradients as natural laboratories to infer species' responses to temperature, Journal of Ecology, 2015, Date accessed: 2019-05-10.

[9] I.J. Wright, N. Dong, V. Maire, I. Colin Prentice, M. Westoby, S. Díaz, R.V. Gallagher, B.F. Jacobs, R. Kooyman, E.A. Law, M.R. Leishman, Ü. Niinemets, P.B. Reich, L. Sack, R. Villar, H. Wang, P. Wilf, Global climatic drivers of leaf size, Science, 2017, Date accessed: 2019-05-03.

[10] J. Kołodziejek, and S. Michlewska, Effect of Soil Moisture on Morpho-Anatomical Leaf Traits of Ranunculus acris (Ranunculaceae), Polish Journal of Ecology, 2015, Date accessed: 2019-06-02.

[11] O. Hamann, On Climatic Conditions, Vegetation Types, and Leaf Size in the Galapagos Islands, Association for Tropical Biology and Conservation, 2009, Date accessed: 2019-06-02.

[12] S. Younis, C. Weiland, R. Hoehndorf et al., Taxon and trait recognition from digitized herbarium specimens using deep convolutional neural networks, Botany Letters, 2018, Date accessed: 2019-06-04.

[13] Royal Botanic Garden Edinburgh, What is herbarium, 1997, www.rbge.org.uk, Date accessed: 2019-06-03.

[14] C.O. Gram, Preparing And Storing Herbarium Specimens, National Park Service, 2009, Date accessed: 2019-06-03.

[15] GBIF: The Global Biodiversity Information Facility, Online database, https://www.gbif.org, Date accessed: 2019-11-18.

[16] iDigBio: Integrated Digitized Biocollections, Online database, https://www.idigbio.org/portal/search, Date accessed: 2019-11-18.

[17] K.E. Taylor, R.J. STouffer, and G.A. Meehl, AN OVERVIEW OF CMIP5 AND ThE EXPERIMENT DESIGN, Bulletin of the American Meteorological Society, 2012, Date accessed: 2019-05-03.

[18] Dr. J. Gaikwad, TraitEx: tool for measuring morphological functional traits from digitized herbarium specimens – Fusion, https://fusion.cs.uni-jena.de/fusion/publications/traitex-tool-for-measuring-morphological-functional-traits-from-digitized-herbarium-specimens/, Date accessed: 2019-06-04.

[19] A. Triki, J. Gaikwad and B. Bouaziz, Trait Ex introduction, 2019, Date accessed: 2019-11-01.

[20] L. Breiman, Random forests(book), Machine learning, 2001, Date accessed: 2019-06-05.

[21] R.P. Haining, Spatial Autocorrelation, International Encyclopedia of the Social & Behavioral Sciences, 2001, Date accessed: 2020-01-02.

[22] D.R. Roberts, V. Bahn, S. Ciuti, M.S. Boyce, J. Elith, G. Guillera-Arroita, S. Hauenstein, J.J. Lahoz-Monfort, B. Schröder, W. Thuiller, D.I. Warton, B.A. Wintle, F. Hartig and C.F. Dormann, Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure, Ecography, 2017, Date accessed: 2019-06-03.

[23] S. Fick, R.J. Hijmans, WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas, International Journal of Climatology, 2017, Date accessed: 2019-11-18.

[24] J.T. Abatzoglou, S.Z. Dobrowski, S.A. Parks, K.C. Hegewisch, TerraClimate, a high-resolution global dataset of monthly climate and climatic water balance from 1958–2015, Scientific Data, 2018, Date accessed: 2019-11-19.

[25] T. Hengl, J. Mendes de Jesus, G. Heuvelink et al., SoilGrids250m: Global gridded soil information based on machine learning, PLOS ONE, 2017, Date accessed: 2019-11-18.

[26] G. Fischer, F.O. Nachtergaele, S. Prieler, E. Teixeira, G. Tóth, H. Velthuizen, L. Verelst, D. Wiberg, Global Agro-ecological Zones (GAEZ v3. 0)-Model Documentation, International Institute for Applied Systems Analysis, 2012, Date accessed: 2019-11-18.

[27] J. Tesky, Salix bebbiana, In: Fire Effects Information System, [Online], 1992, Date accessed: 2019-12-03.

[28] W. Jetz, J. McPherson, R. Guralnick, Integrating biodiversity distribution knowledge: Toward a global map of life, Trends in Ecology and Evolution, 2012, Date accessed: 2019-12-03.

[29] P. Schütt, U. Lang, Enzyklopädie der Holzgewächse: Handbuch und Atlas der Dendrologie, Enzyklopädie der Holzgewächse, 2014, Date accessed: 2019-12-03.

[30] J. Clausen, VIOLA CANINA L., A CYTOLOGICALLY IRREGULAR SPECIES, Hereditas, 1931, Date accessed: 2019-12-03.

[31] M. O'Donnell, D. Ignizio, Bioclimatic Predictors for Supporting Ecological Applications in the Conterminous United States, U.S Geological Survey Data Series 691, 2012, Date accessed: 2019-12-11.

[32] I. Harris, P. Jones, T. Osborn, D. Lister, Updated high-resolution grids of monthly climatic observations - the CRU TS3.10 Dataset, International Journal of Climatology, 2014, Date accessed: 2019-06-05.

[33] Data Access — National Centers for Environmental Information (NCEI) formerly known as National Climatic Data Center (NCDC), Department of Commerce, https://www.ncdc.noaa.gov/data-access/paleoclimatology-data, Date accessed: 2019-06-05.

[34] A. Dubey,, Feature Selection Using Random forest, Towards Data Science, 2018, Date accessed: 2019-06-03.

[35] D. Stekhoven, P. Bühlmann, Missforest-Non-parametric missing value imputation for mixed-type data, Bioinformatics, 2012, Date accessed: 2020-01-27.

[36] M. Jung, J. Zscheischler, A guided hybrid genetic algorithm for feature selection with expensive cost functions, Procedia Computer Science, 2013, Date accessed: 2020-02-04.

[37] Pedregosa et al., Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 2011, Date accessed: 2019-09-07.

[38] S. Ronaghan, The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark, Medium, 2018, Date accessed: 2020-01-28.

[39] SauceCat, Introducing PDPbox - Towards Data Science, Medium, 2017, Date accessed: 2019-11-18.

[40] S. Lavorel, K. Grigulis, P. Lamarque et al., Using plant functional traits to understand the landscape distribution of multiple ecosystem services, Functional Ecology, 2011, Date accessed: 2019-11-28.

[41] K. M. Calinger, S. Queenborough, P. S. Curtis, Herbarium specimens reveal the footprint of climate change on flowering trends across north-central North America, Ecology Letters, 2013, Date accessed: 2020-03-28.

[42] Y. Li, D. Zou, N. Shrestha, X. Xu, Q. Wang, W. Jia, Z. Wang, Spatiotemporal variation in leaf size and shape in response to climate, Journal of Plant Ecology, 2019, Date accessed: 2020-01-14.

Appendix A

Appendices

A.1 Herbar specimen download

Importing required packages

```
import matplotlib.pyplot as plt
%matplotlib inline
import csv
import requests
from PIL import Image
import time
import shutil
from PIL import ImageFile
import pandas as pd
import xlsxwriter
from openpyxl import load_workbook
```

Species name and paths, folder creation

```
#defining species name
Species_name = 'Salix_bebbiana'
#Below code create the required folder according to species name.
creatingfolder = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
                          /pickle_files/{}/'.format(Species_name)
if not os.path.exists(creatingfolder):
    os.mkdir(creatingfolder)
#Path for the raw data, copy the raw data file before starting the
                    #execution of the codes. Raw data file is enough
                        #to create all the files and folders automatically
rawdatapath = "/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
    /pickle_files/{}/{}.pkl".format(Species_name,Species_name)
#Processed data path contains the data which removes all the duplicates
                        #and filters all the data from the raw data provided.
processed_datapath = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
    /pickle_files/{}/{}_processed.pkl'.format(Species_name,Species_name)
```

```
#Excel file path to create the new sheet
Excel_path = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
/Excel_files/Species_updated/Spieces_data_1.xlsx'
#Download path gives the path to download images
downloadpath = '/Net/Groups/BGI/work_1/OBG/DOWNLOAD_Herbar/'
```

Only for *Salix bebbiana* Below block of code belongs to *Salix bebbiana* only, because the data is processed from Gbif and Idigbio separately.

```
df=pd.read_pickle('/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
    /pickle_files/Salix_bebbiana/Salix_bebbiana.pkl')
dfgb=pd.read_pickle('/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
    /pickle_files/Salix_bebbiana/Gbif/Gbif_Salix_bebbiana.pkl')
df.drop_duplicates(subset = ['accessuri'],keep = 'first', inplace = True)
dfgb.drop_duplicates(subset = ['accessuri'],keep = 'first', inplace = True)
AS = df.append(dfgb, sort=False)
AS.drop_duplicates(subset = ['accessuri'],keep = 'first', inplace = True)
AS.shape
```

Combined Gbif and Idigbio If data from Gbif and Idigbio is combined before then the file had to be read directly.

```
#Reading the pickle file
AS =pd.read_pickle(rawdatapath)
print(AS.shape)
#Removing the duplicates by accessuri column
AS.drop_duplicates(subset = ['accessuri'],keep = 'first', inplace = True)
print(AS.shape)
```

```
#Removing all leading and trailing whitespaces from the given string
AS['institutioncode'] = AS['institutioncode'].str.strip()
AS['catalognumber'] = AS['catalognumber'].str.strip()
#Replacing all the white spaces with in the string
AS['institutioncode'] = AS['institutioncode'].str.replace(" ","")
AS['catalognumber'] = AS['catalognumber'].str.replace(" ","")
#Removing all the special characters from the string
AS['institutioncode'] = AS['institutioncode'].str.replace('\W', '')
AS['catalognumber'] = AS['catalognumber'].str.replace('\W', '')
```

Creating unique ids for each sample like Sample_id and Image_Id.

```
AS['Sample_id'] = AS['institutioncode'].astype(str) + '_' +
                  AS['catalognumber'].astype(str)
AS['Image_id'] = AS.Sample_id + '_' + (AS.groupby('Sample_id').cumcount()+1)
                  .astype(str)
```

Setting Index and dropping null values in accessuri to avoid errors while downloading.

```
AS.reset_index(level = 0, inplace = True)
AS.drop(['index'],axis = 1,inplace = True)
print(sum(pd.isnull(AS['accessuri'])))
AS = AS.dropna(axis=0, subset=['accessuri'])
AS.drop(['level_0'], axis='columns',inplace=True)
AS.reset_index(drop=True, inplace=True)
AS['Default_speciesname'] = Species_name
# Saving processed data file
AS.to_pickle(processed_datapath, compression='infer', protocol=4)
```

Looping through pandas data frame to execute perfect download of images, creating folders, sub-folders and images with species name, Image_id and Image_id respectively.

```
for index, row in AS.iterrows():
    Common_name_forspecies= row.Default_speciesname
    parent_folder = os.path.join(downloadpath, Common_name_forspecies)
    # Creating the spieces folder, if it is not there
    if not os.path.exists(parent_folder):
        os.mkdir(parent_folder)

    filename= row.Image_id
    url = row.accessuri
    print(url)
    try:
        result = requests.get(url, stream=True)
        #sleep time between iteration to iteration beacause if we ping the
                #same website repeatedly then it does not respond sometimes
        time.sleep(4)
        if result.status_code == 200:
            #joining the path to create image_id folder with in the species
                                                                    #folder
            child_folder = os.path.join(parent_folder, filename)
            # Creating the image_id folder, if it is not there
            if not os.path.exists(child_folder):
                os.mkdir(child_folder)
            #creating the jpg file with image_id
            full_path = os.path.join(child_folder, '{}.jpg'.format(filename))
            #opening the image file and writing the content
            with open(full_path, 'wb') as f:
                result.raw.decode_content = True
                shutil.copyfileobj(result.raw, f)
            #opening the image file and resizing to the specified
                                                #height and width
            try:
                image =  Image.open(full_path)
                image = image.resize((1900, 2400))
                ImageFile.LOAD_TRUNCATED_IMAGES = True
```

```
                image.save(full_path)
            #If image file is not there then it automatically iterate
                                                    #to the next image
            except:
                pass
    except:
        pass


print('done')
```

Below code opens the existing excel sheet and creates the new sheet with specified columns.

```
book = load_workbook(Excel_path)
writer = pd.ExcelWriter(Excel_path, engine = 'openpyxl')
writer.book = book
AS.to_excel(writer, sheet_name = 'Replace with species name',
        columns=['accessuri','Sample_id','Image_id'], header=True)
writer.save()
writer.close()
```

A.2 Merging Csvs

Importing requires pacakages

```
import matplotlib.pyplot as plt
import csv
import os
import pandas as pd
import pickle
from openpyxl import load_workbook
```

Path names and file names

```
Species_name = 'Viola_canina' #defining species name at once
#Path to read all the csvs present at different folders
All_csvs_path = '/Net/Groups/BGI/work_1/OBG/DOWNLOAD_Herbar/Processed_Images
                /{}'.format(Species_name)
#Below paths required for only option 1
creatingfolder = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
                /csv_files/{}/'.format(Species_name)
if not os.path.exists(creatingfolder):
    os.mkdir(creatingfolder)
mergecsv_path = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download/csv_files
                /{}/merge.csv'.format(Species_name)
#For opening excel file and manipulating the data
```

```
Excel_path = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download/Excel_files
            /Species_updated/Spieces_data_1.xlsx'
#Lat long data file path, i used the processed file of the given species
lldatapath = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download/pickle_files
            /{}/{}_processed.pkl'.format(Species_name,Species_name)
#Finaldata path
Finaldatapath = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download
    /pickle_files/{}/{}_finaldata.pkl'.format(Species_name,Species_name)
```

Options to merge all the csv's available There are two options available to merge all the available csv's, they are:

Option 1 Lets use the piece of code (below), if the csv files in not seperated by the single delimiter. For example, semicolon is the delimiter for some csv files and comma is another delimiter for remaining csv files, then below code is very helpful to merge all the csv's together.

Attention : Since this code simply joins the line with in the csv's so, if some one executed the code second time then it automatically joins all the csv lines again for the existing merge.csv (in the below case) file. While saving the merged csv file, change the path as required.

```
with open(mergecsv_path,"a") as fout:
    for f in os.listdir(All_csvs_path):
        expected_csv_path = os.path.join(All_csvs_path, f, f + '.csv')
        csv_exists = os.path.isfile(expected_csv_path)
        if csv_exists:
            x = open(expected_csv_path)
            x.readline()
            for line in x:
                fout.write(line)
            x.close()
df = pd.read_csv(mergecsv_path, header=None,sep='[;,]',engine='python',
 names=['LeafID','type','length','width','area','perimeter','circularity',
        'parent'])
df.reset_index(inplace=True)
df.rename({'index': 'ImageID'}, axis='columns',inplace=True)
```

Option 2 Irrespective of the option 1, if the csv's had only separated by one delimiter then the below code is useful to merge the csv's. In option 1, the csv file is saved in some directory but in the option 2 saving the csv file is not required.

```
df = pd.DataFrame()
for f in os.listdir(All_csvs_path):
    expected_csv_path = os.path.join(All_csvs_path, f, f + '.csv')
    csv_exists = os.path.isfile(expected_csv_path)
    if csv_exists:
```

```
        x = pd.read_csv(expected_csv_path, header=0,delimiter=';',
            index_col=False)
        df = df.append(x)
df.reset_index(inplace=True,drop=True)
print(df.shape)
```

Here is the one method that created the more data frames to ensure final data frame contains all the data. But in another method, one can simply delete every second row to get rid of petiole length. Comparing both the methods, first method is the best one to do because no data has to be deleted with first method.

```
#Creating a dataframe name pl=petiole_length with the column name Petiole
    #length contains the data of every second row and the lenght column of
#dataframe df
pl=pd.DataFrame({'Petiole_lenght' : df.iloc[1::2,3]})
#resetting the index
pl.reset_index(drop=True,  inplace=True)
#wpl= without_petiole_lenght contains every first row and all the columns
wpl=df.iloc[::2,:]
wpl.reset_index(drop=True,  inplace=True)
#joining wpl and pl gives us the complete initial data of all the specimens
                        #including petiole lenght (td=total_data)
td=wpl.join(pl[['Petiole_lenght']])
```

Since part of the ImageID contains the real Image_id, so the string split command is used and created a new column as Image_id and dropped some columns ImageID, LeafId, type, parent which are not at all required for us. Some of the column names are changed as per the requirements.

```
Columnsplit=td['ImageID'].str.split("/", n = 6, expand = True)
td['Image_id']=Columnsplit[5]
td.drop(['ImageID', 'LeafID','type','parent'], axis=1, inplace=True)
td.rename({'length': 'Leaf_lenght', 'width': 'Leaf_width', 'area': 'Leaf_area'
                ,'perimeter': 'Leaf_perimeter'}, axis='columns',inplace=True)
```

Updating the excel file column (Number of leaves measured)

```
x=td['Image_id']
count=x.groupby(x.tolist()).size()
count=count.to_frame()
count.reset_index(inplace=True)
count.rename({'index': 'Image_id', 'Image_id': 'Number of leaves measured'},
                                axis='columns',inplace=True)
print(count.head())
book = load_workbook(Excel_path)
writer = pd.ExcelWriter(Excel_path, engine='openpyxl')
writer.book = book
writer.sheets = dict((ws.title, ws) for ws in book.worksheets)
Spd=pd.read_excel(Excel_path, sheet_name='{}'.format(Species_name))
```

```
Spd.set_index('Image_id', inplace=True)
Spd.update(count.set_index('Image_id'))
Spd.reset_index(inplace=True)
Spd.head()
Spd.to_excel(writer, '{}').format(Species_name)
writer.save()
```

Now the data frame td was merged with the meta data frame based on Image_id to obtain final data frame.

```
#Doing mean values for the multiple measurements for each sample
td=td.groupby('Image_id').mean().reset_index()
#opening lat long data file
lldata=pd.read_pickle(lldatapath)
#Merging based on td
tdf=pd.merge(td, lldata, on='Image_id')
tdf.to_pickle(Finaldatapath, compression='infer', protocol=4)
```

A.3 Extracting features from Soil Grids data set

Importing all the required libraries

```
import numpy as np
import xarray as xr
import pandas as pd
```

Filepaths and species name

```
Species_name = 'Salix_bebbiana'
#Finaldata path
Finaldatapath = 'path to the file/{}/{}_finaldata.pkl'.
                        format(Species_name,Species_name)
```

Functions for the weighted averages for data arrays and data set There is no predefined functions to do the weighted mean, so the functions to do the weighted average was created by me. The functions below are self explanatory.

```
def average_da(self, dim=None, weights=None):
    if weights is None:
        return self.mean(dim)
    else:
        if not isinstance(weights, xr.DataArray):
            raise ValueError("weights must be a DataArray")

        # if NaNs are present, we need individual weights
        if self.notnull().any():
            total_weights = weights.where(self.notnull()).sum(dim=dim)
```

```
        else:
            total_weights = weights.sum(dim)

        return (self * weights).sum(dim) / total_weights
def average_ds(self, dim=None, weights=None):
    if weights is None:
        return self.mean(dim)
    else:
        return self.apply(average_da, dim=dim, weights=weights)
```

Defining weights

```
# Creating the new data array to define the weights
data = np.array([[0],[0.05],[0.15],[0.3]], np.float32)
weights = xr.DataArray(data, dims=('layer', 'y'))
```

Three dimensions of Soil Grids data Some files had dimensions of latitude: 1800, longitude: 3600, levels: 7 and some files with latitude: 1800, longitude: 3600, levels: 9 and others with latitude: 1800, longitude: 3600. It is not possible to open all the files with different dimensions at the same time, so all the files with level 7 was opened in data set ds1, all the files with level 9 in data set ds2, all the remaining files in data set ds3.

```
ds1 = xr.open_mfdataset('/path to files/*.7.nc')
#loading data in to the memory
ds1 = ds1.load()
#Slicing all the data up to level 4
ds1 = ds1.sel(layer=slice(1.0,4.0))
#Computing weighted average
ds1 = average_ds(ds1, dim='layer',weights=weights)
#Removing the dimension y which was created by us at the creation of data
                                                        #array for weights
ds1 = ds1.isel(y=0)
ds2 = xr.open_dataset('/path to files/OCSTHA.SoilGrid.3600.1800.9.nc')
ds2 = ds2.load()
ds2 = ds2.sel(layer=slice(1.0,4.0))
ds2 = average_ds(ds2, dim='layer',weights=weights)
ds2 = ds2.isel(y = 0)
ds3 = xr.open_mfdataset('/path to files/*.nc')
ds3 = ds3.load()
#Merging all the datasets
ds4 = ds1.merge(ds2)
ds = ds4.merge(ds3)
ds
```

Extracting features from the Soil Grids data base using the below code.

```
#Reading the pickle file
```

```
df = pd.read_pickle(Finaldatapath)
#Creating empty dataframe
edf=pd.DataFrame()
#Iterating to each row in the dataframe till the end
for index, row in df.iterrows():
    latitu=row.lat #reading the lat value
    long=row.lon #reading the long value
    #Selecting the data based on the above lat long value
    x=ds.sel(latitude=latitu,longitude=long, method='nearest')
    #converting dataset to array values
    p=x.to_array().values
    #Creating the data frame and writing all the values from the above array
    some=(pd.DataFrame({'AWCh1':p[0],'AWCh2':p[1],'AWCh3':p[2],'AWCtS':p[3],
                        'BLDFIE':p[4],'CECSOL':p[5],'CLYPPT':p[6],
                    'CRFVOL':p[7],,'OCDENS':p[8],'ORCDRC':p[9],'PHIHOX':p[10],
                       'PHIKCL':p[11],'SLTPPT':p[12],'SNDPPT':p[13],
                     'TEXMHT':p[14],'WWP':p[15],'OCSTHA':p[16],'ACDWRB':p[17],
                       'BDRICM':p[18],'BDRLOG':p[19],'BDTICM':p[20],
                       'HISTPR':p[21],'SLGWRB':p[22]},index=[0]))
    edf=edf.append(some) #appending all the data in to empty dataframe
edf.reset_index(drop=True, inplace=True)
final = df.join(edf)
print(final.shape)
final.to_pickle(Finaldatapath, compression='infer', protocol=4)
```

A.4 Extracting features from climate change data set

Importing required libraries

```
import xarray as xr
import pandas as pd
```

Filepaths and species name

```
Species_name = 'Salix_bebbiana'
#Finaldata path
Finaldatapath = '/path to species/{}/{}_finaldata.pkl'
                    .format(Species_name,Species_name)
```

Function to fetch all the Cmip data files (past1000, historical, rcp 6.0) Every scenario (past1000, historical, rcp 6.0) had different path, so I created a function to combine all the scenarios in to one file.

```
def CMIP_past_histo_RC6(variable,variable_past1000,variable_historical
                        ,variable_rcp6):
```

```
#Path for past1000 files
path_past1000 = 'path to past1000 scenario/past1000/{}/'
                    .format(variable_past1000)
#if the lenght of the files in the directory is more than one then it
        #chooses the line immediately after if condition, in other case
                #it choose else.
if len([name for name in os.listdir(path_past1000) if os.path.isfile
                        (os.path.join(path_past1000, name))])>1:
    ds_past1000= xr.open_mfdataset('/path to past1000 scenario/past1000/{}
    /*.nc'.format(variable_past1000),decode_times=False)
else:
    ds_past1000= xr.open_dataset('path to past1000 scenario/past1000/{}/
            {}_Amon_MIROC-ESM_past1000_r1i1p1_085001-184912.nc'.format
                (variable_past1000,variable_past1000),decode_times=False)
ds_past1000=ds_past1000.load()
# slicing the data from 1678 to 2100 because xarray used another time
                    #frame before 1678 same as pandas
ds_past1000=ds_past1000.sel(time=slice(302431,365230))
#we decoded the time because we kept decode_times=False opening the file
ds_past1000=xr.decode_cf(ds_past1000)

path_historical = 'path to historical scenario/historical/{}/'.
                            format(variable_historical)
if len([name for name in os.listdir(path_historical) if os.path
        .isfile(os.path.join(path_historical, name))])>1:
    ds_historical= xr.open_mfdataset('/path to historical scenario/
                historical/{}/*.nc'.format(variable_historical))
else:
    ds_historical= xr.open_dataset('path to historical scenario/historical
            /{}/{}_Amon_MIROC-ESM_historical_r1i1p1_185001-200512.nc'.
                        format(variable_historical,variable_historical))
ds_historical=ds_historical.load()

path_rcp60 = '/path to rcp6 scenario/rcp60/{}/'.format(variable_rcp6)
if len([name for name in os.listdir(path_rcp60) if os.path.isfile
                                (os.path.join(path_rcp60, name))])>1:
    ds_rcp6= xr.open_mfdataset('/path to rcp6 scenario/rcp60/{}/*.nc'
                    .format(variable_rcp6))
else:
    ds_rcp6= xr.open_dataset('/path to rcp6 scenario /rcp60/{}/
            {}_Amon_MIROC-ESM_rcp60_r1i1p1_200601-210012.nc'.
                        format(variable_rcp6,variable_rcp6))
ds_rcp6 = ds_rcp6.load()

ds=xr.concat([ds_past1000,ds_historical,ds_rcp6],
          dim='time')
#Removing the dimensions which are not required for us
ds=ds.drop('lon_bnds')
```

```
    ds=ds.drop('lat_bnds')
    ds=ds.drop('time_bnds')
    return ds
```

Cmip mean deviation Calculation of climate change data set and extracting climate change features are shown below.

```
var = ['ts','tas','tasmax','tasmin','pr','rsds','sfcWind','prw']
for i in var:
    some = CMIP_past_histo_RC6(i,i,i,i)
    #Slicing the data from 1970-2000
    mean = some.sel(time=slice('1970-01-01','2000-12-31'))
    #Calculating the mean
    mean = mean.mean()
    #Mean deviation is original dataset minus mean value
    mean_deviation = some-mean
    df = pd.read_pickle(Finaldatapath)
    df['datecollected'] = pd.to_datetime(df['datecollected'], utc=True)
                                                        .dt.date
    edf=pd.DataFrame()
    for index, row in df.iterrows():
        latitu=row.lat
        #Here we need to use plus 180 because cmip long values varies
                    #from 0-360, but normally we had -180 to 180 in our data
        long=(row.lon)+180
        tim=row.datecollected
        x=mean_deviation.sel(lat=latitu,lon=long,time=tim, method='nearest')
        p=x.to_array().values
        #print(x)
        ext=(pd.DataFrame({'{}(meandev of cmip)'.format(i):p[0]},index=[0]))
        edf=edf.append(ext)

    edf.reset_index(drop=True,inplace=True)
    final = df.join(edf)
    print(final.shape)
    final.to_pickle(Finaldatapath, compression='infer', protocol=4)
```

A.5 Extracting additional features

```
Species_name = 'Salix_bebbiana'
#Finaldata path
Finaldatapath = '/Net/Groups/BGI/work_1/OBG/Vamsi/Herbarium_Download/
    pickle_files/{}/{}_finaldata.pkl'.format(Species_name,Species_name)
```

Calculating Draught stress index Draught stress index is equals to actual evapotranspiration/potential evapotranspiration. How to calculate the draught stress index by using two data sets in python is shown in the below code.

```
ds_aet = xr.open_dataset('path to the file')
ds_pet = xr.open_dataset('path to the file')
Draught_stress_index = xr.Dataset({'Draughtstressindex' : (('latitude',
                            'longitude'), ds_aet['aet']/ds_pet['pet'])},
                                    coords=ds_pet.coords)
```

Calculating the mean seasonal amplitudes for climate water deficit and palmer drought severity index Mean seasonal amplitudes are calculated by subtracting the maximum monthly mean value and minimum monthly mean value of the given data as shown in the below code (similar for climate water deficit and palmer drought severity index).

```
#Opening all the datafiles for a variable
ds_pdsi = xr.open_mfdataset('path to data/*.nc')
#Slicing the temporal frequency for 1970 to 2000
ds_pdsi = ds_pdsi.sel(time=slice('1970-01-01T00:00:00.000000000'
                        ,'2000-12-01T00:00:00.000000000'))
#Doing monthly mean
ds_pdsi = ds_pdsi.resample(time='M').mean()
#Resampling the data and taking the maximum monthly value
ds_pdsi_max = ds_pdsi.resample(time='A').max('time')
#Resampling the data and taking the minimum monthly value
ds_pdsi_min = ds_pdsi.resample(time='A').min('time')
#Creating new data set
ds_pdsi_amplitude =  xr.Dataset({'pdsi_amplitude' : (('time','latitude',
                'longitude'), ds_pdsi_max['PDSI'] - ds_pdsi_min['PDSI'])},
                        coords=ds_pdsi_max.coords)
#Doing mean value
ds_pdsi_amplitude = ds_pdsi_amplitude.mean(dim='time')
```

After calculating all the required data sets, the features was extracted as shown in the section A.3

A.6 Block cross validation

Importing required libraries

```
import numpy as np
import pandas as pd
```

Generating block Lets say block size = 1 degree, so nrows in the below code becomes to 180, ncols becomes to 360. At the beginning the data (virtual matrix 360∗180) has all zero values. Now all the zero values with range of 0 to 8 has to be filled, because its a 3∗3 blocks. Now one need to visualize the 360∗180 matrix in such a way that the whole matrix is divided in to equal 3∗3 blocks and one need to fill all the 3∗3 blocks with the unique numbers from 0 to 8.

Below code consists of i_start in range(3) and j_start in range(3) and fill_number with some equation. if we run this part of code it automatically gives the unique fill_number value from 0 to 8 for each iteration. we had the fill_number 0 to 8 and we need to fill these numbers in 3∗3 blocks. Coming to data now, we had definition named fill_one_number, lets go to the definition to see whats happening there

```
def generate_block(block_size):
    nrows = int( 180/block_size )
    ncols = int( 360/block_size )
    data = np.zeros((nrows,ncols))

    for i_start in range(3):
        for j_start in range(3):
            fill_number = i_start * 3 + j_start
            data = fill_one_number(data, i_start, j_start, fill_number, nrows,
                                                                        ncols)
    return data
```

Visualize the virtual matrix and filling the each indices with some number Here data is filled with all zeros, i_start (0 to 2), j_start (0 to 2), fill_number (0 to 8), nrows=180, ncols=360. So the code below iterates for each possible combination of i_start, nrows ; j_start, ncols, and fills the number with fill_number in the data. Finally, the data output comes as all the 3∗3 blocks filled with unique number (0 to 8).

```
def fill_one_number(data, i_start, j_start, fill_number, nrows, ncols):
    for i in range(i_start, nrows, 3):
        for j in range(j_start, ncols, 3):
            data[i,j] = fill_number
    return data
```

Final output for the block If the lat long position is specified in the below definition it automatically gives us the fill_number by iterating through the above mentioned definitions. By using all the above definitions, the block numbers will be given to all the latitudes and longitudes of digital herbar specimens. This block numbers is used to split the data in to 9 folds, so the data from each fold is independent to other fold, this ensures that all the herbar specimens are independent to each other.

```
def get_index(block, block_size, lng, lat):
```

```
    irow = int((90 + lat) / block_size)
    icol = int((lng + 180) / block_size)
    return block[irow,icol],irow,icol
```

A.7 Methodological limitations

To solve the general aim of this thesis, there are several methodological challenges.

- Measuring the leaf sizes of herbar specimen is challenging because the Trait Ex tool used for measuring the leaf sizes of herbar specimen is a semi-automated tool. So I need to process every specimen by hand.
- Various data sets from different sources (Trait Ex tool, different soil databases, climate databases and climate simulations) have to be extracted, harmonized and pre-processed.
- Identifying most influencing environmental controls on the intraspecific variability of leaf area using machine learning approaches.
- Development of a block selection cross-validation procedure.
- Feature selection will be needed since the number of potential features is large compared to the amount of data.

What happens if the herbar specimen images are not saved in the well organized way as shown in Figure A.1?

Trait Ex generates seven output files for each measurement measured. So if I store all the digital herbar specimens in the same directory then it is hardly impossible to search for the next correct images (which has to be measured) because of overcrowded output files. It is not a good practice to search the particular image in hundreds of output files. So I designed a workflow (saving images as shown in figure A.1) to overcome the overcrowding output files problem.

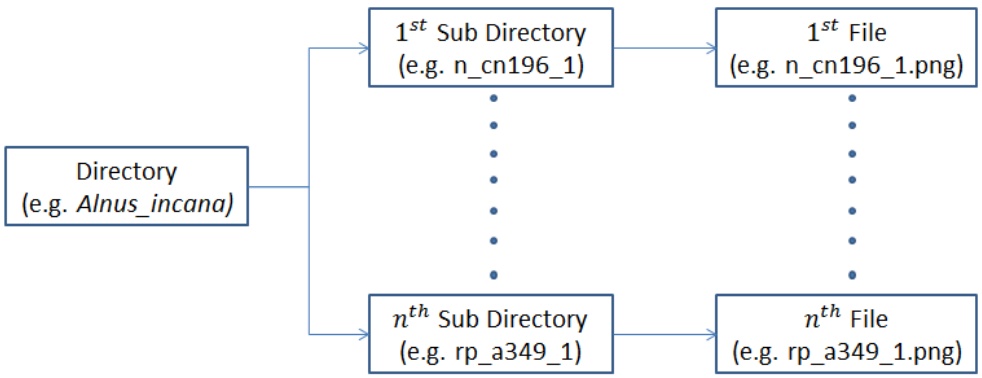


Figure A.1: Outlook of file directory