Ernst-Abbe-Hochschule Jena
University of Applied Sciences

Master Thesis

on

Disentangling Spatial and Temporal Effects of Climate on Leaf Size by using Image Recognition and Machine Learning

by

Pramod Baddam
Matriculation number: 642298

submitted to obtain the degree of

Master of Science (M.Sc.)

at

University of Applied Sciences Jena

Course of Studies

Scientific Instrumentation

University supervisor:	Prof. Dr. Barbara Wieczorek Ernst-Abbe Hochschule Jena, Jena
Company supervisor:	Dr. rer. nat. Susannen Tautenhahn Max Plank Institute for Biogeochemistry, Jena

Jena, September 2020

DECLARATION

I hereby declare that this master thesis work entitled **"DISENTANGLING SPATIAL AND TEMPORAL EFFECTS OF CLIMATE ON LEAF SIZE BY USING IMAGE RECOGNITION AND MACHINE LEARNING"** is a presentation of my work. All content and ideas drawn directly or indirectly from external sources are indicated as such and every effort has been made to cite and acknowledged with due reference to the literature in this thesis work.

Place, date	Signed: PRAMOD BADDAM

Abstract

Climate change has caused severe impacts on the ecosystem, and it is also affecting the biodiversity and balance of the ecosystem functioning. Plant functional traits show good response and adaptation to environmental changes, and it also indirectly indicates climate change. However studies on trait-environment relationship have two limitations such as 1) Usually they rely on species mean trait values and intraspecific differences are ignored, Intraspecific variability is only rarely studied to represent the adaptive capacity to climate change in leaf size as well, 2) Usually spatial gradient is studied as natural laboratories to infer effects of climate change since long term data is not available.

In this master thesis, the digital herbar specimen data are downloaded from Gbif and Idigbio, and leaf values are extracted by using image recognition technique with a semi-automated tool called TraitEx and in order to generate models for the understanding of how plant functional traits (Leaf size) adjust according to changing environmental conditions (mainly temperature), in future. This space for time approach will, for the first time offer the unique opportunity to study the adaptation of leaf size over centuries. We specifically want to overcome space for time analysis by focusing on the separation of the effects of space and time while controlling for the season, and for analyzing the intraspecific trait variability machine learning algorithms such as random forest regressor, extreme gradient boosting regressor and mixed effect random forest are used along with other techniques such as cross validation, visualization, and variation partitioning.

The results show the proportion of variability explained by each of the three predictor groups spatial, temporal, seasonal, and their respective interactions and overlaps also gave an insight on how the climate change impacts the intraspecific variability of leaf size and from the results it shows that variations in the leaf size due to climate change are mostly due to spatial features then followed by seasonal features. Moreover, the overall accuracy and learning capabilities of random forest model are compared with the xtreme gradient boosting model with monotonic constraint are similar in both models.

Overall, this study and analysis of plant functional traits according to changing environmental conditions with the functional approach of space for time hypothesis can be applied for predicting the biodiversity and ecosystems functioning in the future.

ACKNOWLEDGEMENT

I Would like to express my sincere gratitude to Dr. rer. nat. SUSANNEN TAUTENHAHN, POSTDOCTORAL RESEARCHER, FROM DEPARTMENT OF FUNCTIONAL BIOGEOGRAPHY AT MAX PLANK INSTITUTE FOR BIOGEOCHEMISTRY, JENA, GERMANY for providing me with this opportunity to do Master Thesis and to her endless support and kindness throughout my entire time of master thesis at institute.

I would like to express my sincere appreciation and indebtedness to following individuals Dr. Jens Kattage, Dr. Jitendra Gaikwad and Mr. Vamsi Krishna Kommineni for their continuous support and for giving insight feedback throughout the thesis work. Also, I would like to acknowledge all my colleagues who have directly or indirectly helped me in my work.

I sincerely like to express my profound gratitude to Prof. Dr. BARBARA WIECZOREK for her support and understanding in every aspect and for giving me complete freedom of moving forward with my research work and technical support without any hesitation whenever necessary.

None of this would have been possible without an active support group first, my committee members, each of whom provided patient advice and guidance through the research process. Secondly, my friends and family, who supported me with love and understanding. Many thanks to all for your unwavering support.

With Best Regards,
Pramod Baddam

Contents

DECLARATION **i**

Abstract **ii**

ACKNOWLEDGEMENT **iii**

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives of this Thesis 4
1.3 Outline of the report 5

2 Theoretical Background **6**
2.1 Herbar Specimen 6
2.2 TraitEx 8
2.2.1 How TraitEx work and the automated part 8
2.3 Machine Learning 8
2.3.1 Decision Tree 9
2.3.2 Ensemble methods 11
2.3.3 Random Forest 13
2.3.4 XGBoost (eXtreme gradient boosting) 14
2.3.5 Mixed effect Random Forest 16
2.3.6 Spatial autocorrelation 17
2.3.7 Cross validation 17
2.3.8 Hyper Parameter Tuning 18
2.3.9 Variation Partitioning 19
2.3.10 Data visualization techniques 21

3 Methodology **22**
3.1 Downloading Herbar specimen 23

3.2 TraitEx 25
3.3 Data Pre-Processing 30
3.3.1 Merging different TraitEx output files 30
3.3.2 Data Filtering 31
3.3.3 Performing Groupby based on Image_id 32
3.3.4 Merging orginal pkl with pkl file with leaf values 32
3.4 Testing for spatial autocorrelation 32
3.5 Machine Learning Methods and algorithms 33
3.5.1 Random forest regressor 33
3.5.2 XGBoost Regressor 36
3.6 Block Selection Cross Validation 36
3.7 Hyper Parameter Tuning 38
3.7.1 Random forest Regressor 38
3.7.2 XGBoost Regressor 39
3.8 Variation Partitioning 39
3.9 Mixed effects Random Forest 40
3.10 Species and their data distribution 40

4 Results **45**
4.1 Results on Random Forest 45
4.2 Results on XGBoost models 52
4.3 Results on Variation Partitioning 54
4.4 Results on Mixed effect Random forest models 56

5 Discussion **59**

6 Conclusion **63**

A Appendix A **64**
A.1 Downloading herbar specimen images 64
A.1.1 Saving images into specific folders 65
A.2 Merging all CSV files generated by TraitEx per specimen which contains Leaf values into one main CSV file 66
A.3 Filtering the main CSV file and saving as pkl file 67
A.4 Merging metadata file from Gbif and Idigbio with the pkl files which contain trait values 68
A.5 Creating Blocks for block selection cross validataion 68
A.5.1 Generating 360x720 Matrix with block size of 0.5 degree and assigning values to those blocks 68
A.5.2 Assigining values to 360x720 matrix 69

A.5.3 Creating Indices for each row and column 70
A.6 Hyperparameter Tuning using Randomized Search CV method 70
A.7 Data Pre-Processing 71
A.7.1 Implementing blocks and DEM (altitude) 71
A.7.2 Encoding Cyclic features and removing outliers 72
A.8 Implementing Blocks along with Machine learning algorithm 73
A.8.1 Organizing datasets, model fitting and getting R2 score 73
A.9 Variation Partitioning 75
A.10 Fitting model to the data with XGBoost Regressor 77
A.10.1 Hyper Parameter Tuning for XGBoost 77
A.10.2 XGBoost 78
A.11 Feature importance plots 80
A.12 Partial dependency plots 81
A.13 Saptial Distribution of data on Cartopy map with Robinson projection . . 81
A.14 Mixed effect Random Forest 82
A.14.1 Spatial joining of species with ecoregions 82
A.14.2 Transforming Day of year to Extraterristial radiation 84
A.14.3 Fitting MERF model to data 86
A.14.4 Plotting using MERF model 87

Bibliography **89**

List of Figures

1.1 Global trends in leaf size (LS) in relation to latitude and Seasonal climate [50] . . . 4

2.1 Herbar Specimen . . . 7
2.2 Simple decision tree [19] . . . 9
2.3 Bagging [29] . . . 12
2.4 Boosting [55] . . . 13
2.5 Sample Random Forest Decision Tree Model . . . 14
2.6 Monotonic Constraint [52] . . . 15
2.7 Conceptual illustration of partitioning the variation in a response matrix . 20

3.1 Sample dataframe . . . 24
3.2 Folder and subfolders created based on Image_id . . . 24
3.3 Folder and subfolders created based on Image_id . . . 24
3.4 folders containing digitized specimen images . . . 25
3.5 Steps for setting the scale . . . 26
3.6 Plot scale dialog box . . . 26
3.7 selecting the tools menu . . . 27
3.8 selecting the tools menu . . . 27
3.9 Change leaf orientation . . . 28
3.10 Estimating the leaf shape using a leaf template . . . 28
3.11 Sub folders named based on Image_id . . . 29
3.12 Folders contains Image, text files, and CSV files . . . 29
3.13 Optional caption for list of figures 5–8 . . . 30
3.14 Final CSV File . . . 31
3.15 Example dataframe . . . 31
3.16 Final dataframe . . . 32
3.17 Optional caption for list of figures 5–8 . . . 35
3.18 example for 9 x 9 block . . . 37
3.19 Example matrix . . . 37
3.20 Optional caption for list of figures 5–8 . . . 41
3.21 Optional caption for list of figures 5–8 . . . 41

3.22 Orange Jewelweed with leaves and flowers growing on a creek side [12] . 42
3.23 Optional caption for list of figures 5–8 42
3.24 *Salix glauca* with grey leaf willow and seed on grey leaf willow [28] 43
3.25 Optional caption for list of figures 5–8 43
3.26 *Solanum dulcamara* [10, 11] 44
3.27 Optional caption for list of figures 5–8 44

4.1 Feature importance plots for *Salix glauca* using R-square value of Random forest on left and XGBoost on right 46
4.2 PDP plots for *Salix glauca* using Random forest 46
4.3 Feature importance plots for *Chenopodium album* using R-square value of Random forest on left and XGBoost on right 47
4.4 PDP plots for *Chenopodium album* using Random forest 48
4.5 Feature importance plots for *Impatiens capensis* using R-square value of Random forestIon left and XGBoost on right 49
4.6 PDP plots for *Impatiens capensis* using Random forest 50
4.7 Feature importance plots for *Solanum dulcamara* using R-square value of Random forest on left and XGBoost on right 51
4.8 PDP plots for *Solanum dulcamara* using Random forest 52
4.9 Optional caption for list of figures 5–8 55
4.10 Partial dependency plots of *Salix glauca* species 56
4.11 Partial dependency plots of *Chenopodium album* species 57
4.12 Partial dependency plots of *Impatiens capensis* species 58
4.13 Partial dependency plots of *Solanum dulcamara* species 58

List of Tables

4.1 R-square values for *Salix glauca* species (shrub and tree). 45
4.2 R-square values for *Chenopodium album* species (it can be herb, shrub, or small tree). 47
4.3 R-square values for *Impatiens capensis* species (herb). 48
4.4 R-square values for *Solanum dulcamara* species (shrub). 50
4.5 R-square values of *Salix glauca* species (shrub and tree). 53
4.6 R-square values of *Chenopodium album* species (it can be herb, shrub, or small tree). 53
4.7 R-square values of *Impatiens capensis* species (herb). 53
4.8 R-square values of *Solanum dulcamara* species (shrub). 54
4.9 R-square values of all species using MERF models. 56

List of Abbreviations

Gbif	**G**lobal **B**iodiversity **I**nformation **F**acility
Idigbio	**I**ntegrated **D**igitized **B**iocollections
IPCC	**I**ntergovernmental **P**anel on **C**limate **C**hange
LSAT	**L**and **S**urface on **A**ir **T**emperature
XGB	**X**treme **G**radient **B**oosting
CNN	**C**onventional **N**eural **N**etwork
MERF	**M**ixed **E**ffect **R**andom **F**orest
TEOW	**T**errestrial **E**coregions **o**f the **W**orld

Chapter 1

Introduction

1.1 Motivation

One of the major challenges and the biggest concern in this era is climate change. Climate change caused by human activities has lead to damage ecosystem severely in turn, which expected to have profound consequences on biodiversity and ecosystem functioning [20]. According to the report from IPCC 2019 (Intergovernmental Panel on Climate Change), the average global land surface to air temperature (LSAT) is increased by 1.44 °C from pre-industrialization (1750-1850) period to present days (1999-2018). So, in order to understand how ecosystem composition, structure, and function respond to climate change and its impact on the ecosystem, new methods and approaches are needed [34].

Plant functional traits is a broadly used concept in community and ecosystem ecology to address the responses of species to changes in the environment and the contribution of species to ecosystem functions [21]. Plant functional traits are defined as the characteristics of a plant (morphological, physiological, phenological, structural) that determine how they react to and interact with the surrounding ecosystem (e.g. plant responses to environmental change like climate, soil resources, temperature, water, and nutrient availability, and disturbance like fire, plowing) [21].

One of the most relevant plant traits in this context is leaf size, which is measured as the one-sided projected area of the leaf [50] and is known to be a key control on plant metabolic rates [50]. In the plant kingdom, leaves vary between species from less than $1mm^2$ to greater than $1m^2$ in leaf area [50]. But also, within one species, the high variability of leaf size (so-called intraspecific variability) can be found [48].

On an Interspecific level temperature, rainfall and solar radiation are the key drivers of leaf size around the world, from the giant leaves of tropical plants to the tiny ones of desert dwellers [50]. Larger leaves have a thicker boundary layer that slows sensible heat exchange with the surrounding air. All else equal, under high irradiance, they therefore, develop larger leaf-to-air temperature differences than smaller leaves [50]. All leaves are cooled by transpirational water loss, but this is particularly critical for large leaves, which face a greater risk of potentially serious heat damage at high air temperatures and high irradiance, especially when soil water is limiting transpiration [50]. These principles are central to well-known theories for optimal leaf size based on daytime leaf energy budgets [50], which predict disadvantages to being large-leaved at hotter, drier, and high-irradiance sites. In support of these predictions, many studies have shown smaller mean leaf sizes at sites with lower mean annual precipitation (MAP) and higher irradiance [50]. This pattern instead accords with control by the nighttime energy balance, an under-appreciated influence, which indicates substantial disadvantage for large leaves in cold regions, as they are more prone to frost damage because a thicker boundary layer slows sensible heat exchange with the soil, air, and surrounding vegetation, which is required to offset the decrease of temperature due to long-wave radiation losses to the nighttime sky [50]. However, it is unclear whether these relationships also hold on an intraspecific level.

Studying the effect of climate change on plant functional traits is sophisticated since direct measurements on trait data only exist in recent times. Thus, very often, spatial gradients are taken as natural laboratories to mimic changing environmental conditions [20]. This so-called space for time approach is used, by analyzing over time the response of plant species traits to spatial variation (latitude, longitude, altitude) to understand and predict the variation across time as environmental factors change due to climate change. In trait research, there are further sources of trait variation as a seasonal or ontogenetic variation for which it is necessary to correct within the statistical analysis or by keeping this part of trait variation constant within the measurement process. Despite the simplicity and applicability of the space for time substitution, it also comes with specific advantages and drawbacks as follows.

The Space-for-time substitution is widely used in biodiversity modelling to infer past or future trajectories of ecological systems from contemporary spatial patterns. However, the foundational assumption that drivers of spatial gradients of species composition also drive temporal changes in diversity is rarely tested. Studies comparing populations across sites with contrasting temperatures, soil characteristics precipitation, photoperiod, light quality, biotic interactions, human influence such as longitudinal, altitudinal and latitudinal gradients, can provide invaluable information about the responses to climate changes [20]. This is because adaption of traits to environmental conditions to spatial gradients may involve different processes compared to the adaptation to temporal changes in the environment. For example, over spatial gradients, genetic differences and therefore genotypic adaptation (i.e. changes in the genotype under different environmental conditions) may play a higher role whereas over temporal gradients phenotypic adaptation (i.e. the ability of a genotype to express a different phenotype under different environmental conditions) may play a higher role because genotypic differences are likely to increase with spatial distance [22].

Further over spatial gradients plant species may have adapted over long time periods. Over temporal gradients, there is a chance that adaptation is slower than the rate of environmental change, and observed trait values do not represent the perfect match with the current environment.

To study the long-term trends of trait variations over spatial and temporal gradients simultaneously, we need to establish new data sources that provide information on trait values for larger temporal gradients reaching several hundred years to the past in order to compare the present-day trait values to pre-industrial ones.

In this context, herbaria is an interesting data source since they have a long tradition of collecting plants. Recently there have been large digitization efforts in herbaria worldwide which in combination with image recognition techniques opens new windows for trait-based research over long temporal gradients. The herbar specimens may be whole plants or plant parts, typically a piece of a stem with attached leaves, flowers and/or fruit. The sample has been pressed and thoroughly dried on a sheet of paper (called "exsiccatae") and will last for hundreds of years [49]. Recently, large collections and digitization of herbaria specimen led to the high availability of scanned specimen, which can be downloaded from the Global Biodiversity Information Facility (Gbif) and the Integrated Digitized Biocollections (Idigbio). This offers a solution by providing millions of scanned herbar specimens that have been collected over broad geographic ranges and across centuries. The specimens are documented and used for future reference and scientific study [49].

With the increasing digitization of herbar specimen simultaneously at the same time, the developments in computer-based pattern recognition from images are also rapidly improving which uses deep learning algorithms such as conventional neural networks (CNN) for recognizing the taxa and traits from herbarium specimen images to measure functional traits such as leaf area, etc [23]. By combining these techniques with the digitized herbar specimen the analysis of leaf size adaptions with respect to changing climates are possible which is also, a first of its kind.

In the literature analysis of Wright et al (on Global climatic drivers on leaf size)[50] on interspecific differences in leaf size relationships space (e.g. latitude) and season (day of the year) can be found as shown in figure 1.1. But, until now there are only a few studies on analysis variations in leaf size due to climate change as analyzed over large temporal gradients (> 100 years) like previous studies by Yaoqi Li et al [31], large geographic extent and on the intraspecifc level in order to understand spatial and temporal adaptions in leaf size to a changing environment. Further, it is not clear whether trait environment relationships on an intraspecific scale follow the ones on the interspecific level. Here I plan to fill this gap by analyzing the intraspecific variation in leaf size as a function of space, time (over a gradient of greater than 100 of years), and season simultaneously at a time.

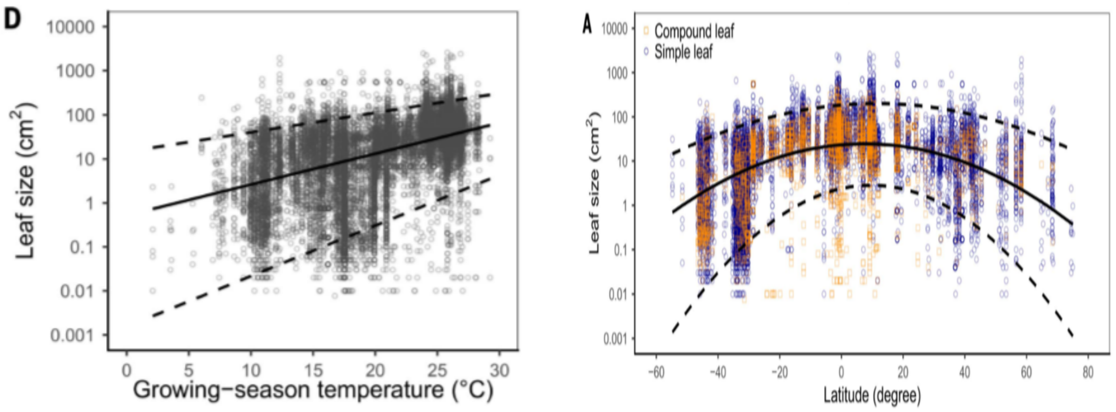


FIGURE 1.1: Global trends in leaf size (LS) in relation to latitude and Seasonal climate [50]

1.2 Objectives of this Thesis

We plan to do this analysis for within the species (intraspecific) for which I have many available images in Gbif and Idigbio and which are easy to digitize in TraitEx. Therefore, we choose the species *Impatiens Capensis*, *Salix Glauca*, *Chenopodium album*, and *Solanum dulcamara* for our analysis by comparing individuals of the same species. Specifically, I do want to answer the following questions:

- Do spatial, temporal, seasonal predictors or overlap of them contribute to understanding the intraspecific variability of leaf size?
- Do different species show similar or different results according to the above-mentioned point?

- What kind of trend does leaf size show with respect to time independent of spatial variation?

This thesis work also focuses on using TraitEx for extracting the measurements of traits from digitized herbar specimens using an image recognition technique. There are two different machine learning algorithms used in this thesis one is Random Forest and the other one is Xtreme Gradient Boosting. Further, an approach with a specific focus on the separation of the effects of space and time while controlling for the season is implemented for analyzing the intraspecific trait variability.

I hypothesize that besides geographic differences also at a given location leaf size changed over time, due to the rising temperatures because the temperature is a key control on plant metabolic rates as well as Photosynthetic carboxylation increases strongly with temperature. The very high or low temperatures can able to impair enzyme function, disrupt membranes and cellular processes, and if sufficiently extreme, cause irreparable tissue damage [50]. Plants show a variety of adaptations depend on temperature ranges thus why temperature is one of the major determinants of the physiology and distribution of organisms [20]. Also, It would be interesting for me to see the effects of Carbon dioxide (Co2) because very high Co2 in the air effects the leaves by shrinking the leaf pores which is used for evapotranspiration and it also, thick's their leaves so it will face a serious heat damage by limiting the amount of evaporation from the land surface.

1.3 Outline of the report

The following chapter 2 will address the theoretical background and contains a literature review, where relevant literature and theoretical concepts will be explained and elaborated. After that in chapter 3, the methodology, empirical data, and in chapter 4 the respective results of the experiments will be shown. Chapter 5 will discuss the results, critically review the taken approaches and methods as well as examine the validity and reliability of the presented results. Finally, chapter 6 will summarize the work and give an outlook for possible future research and expansion on this topic. In the appendix, I provide documented code on all analyses presented in the thesis.

Chapter 2

Theoretical Background

In this chapter, I provide an overview of the relevant theoretical concepts necessary to understand this thesis work. Additionally, the literature review gives an overview of the tools and methods that are already proposed by other research. Furthermore, it describes which tools, methods, and techniques are used in this thesis work.

2.1 Herbar Specimen

Plant functional traits are key for plant success and ecosystem functioning [49], but it is very difficult to collect the necessary data to determine specific environmental factors that drive traits transitions for a given species. Normally, the collection of these data has required long-term field observations or manipulative experiments that are difficult to capture changes in traits for longer time scales on which climate changing is happening as well as it is a time taking process [49].

So, in order to avoid this, researchers have recently turned to the vast collections of plant specimens from all over the world and made them available as a digitized herbarium specimen [49].

For hundreds of years, botanists and naturalists have collected and preserved plants as herbarium specimens for taxonomic research, to record the flora of a region, to document their economic uses, and as a social hobby [49]. As plant collection became more and more among professional botanists in the 18th and 19th centuries, however, the documentation of additional information with each specimen became more detailed and standardized. Most specimen labels created during the past 150 years provide information on locality, date of collection, and habitat as in figure 2.1. In addition to label data, it also contains bar codes, on the top and bottom of the sheet, and sometimes reference color bar on the sides [53].

Physical specimens are rich with information regarding plant health, morphology, and phenological status which provides informations on species reproductive season (e.g., flowers in May–June) and application in agriculture such kind of data can be used for more detailed studies of ecological and evolutionary processes, such as traits sensitivity to temperature.

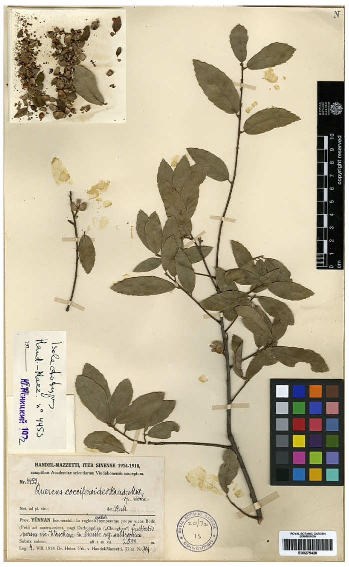


FIGURE 2.1: Herbar Specimen

With the recent growth in the herbarium-based plant functional traits, research the interest in climate change and plant traits around the current 21st century are also growing [49]. This, in turn, led to the use of specimens to estimate adaptions of plant traits to various environmental factors, including temperature, day length, and precipitation. To date, specimens have been used to estimate the onset of several phenophases, including first flowering, peak flowering, leaf-out, and fruit set, as well as the duration of entire growth phases [49]. These estimates have been used to study long-term shifts in plant traits due to interannual or spatial variation in climate [49].

Herbarium-based data, like all sources of data, are likely to have certain biases and limitations [49]. Such limitations are present from the specimen collection phase, to the digitization and processing of specimens, to the analysis and interpretation of specimen data.

It also contains additional biases due to the reason botanists often collect samples depending on their interests, schedule, and location (e.g., near roadsides, populated areas, or universities) and not to capture a representative trait value status of the plant. Collection biases relating to plant habit, morphology, and nativity may also occur in herbarium datasets. Additionally, broader taxonomic, spatial, and temporal biases have been identified with the Global Biodiversity Information Facility (GBIF) occurrence records, which include herbarium records [49].

The Global Biodiversity Information Facility (GBIF) and Integrated Digitized Biocollections (iDigBio) that aims to make scientific data and images for millions of biological specimens available in electronic format for the research community, government agencies, students, educators, and the general public for free of charge and long-term, in digital form on the Internet.

GBIF mainly supplies scientific data on biodiversity from over 1.7 billion documents in natural history research collections around the world. This data is being carefully supplemented by a growing volume of observed data.

2.2 TraitEx

TraitEx is a semi-automated tool, which is used to measure the plant functional traits such as leaf length, area, width, and petiole length from digitized herbarium specimens [23, 46]. Since, the image segmentation (which is used to represent the image in more meaningful and easier to analyze by locating objects and boundaries such as curves, lines, etc) of the TraitEx is not working properly due to that we have to work semi-automated, such as loading images, setting the scale, mapping each leaf and orientating the leaf in a right way is done manually by the user (for more in details refer to section 3.2) and the rest of the tasks done automatically by TraitEx itself. Further, to measure the traits, the presence of a measuring scale of 10 cms in digitized herbarium specimens is required without the scale, the tool cannot measure the trait values.

2.2.1 How TraitEx work and the automated part

Deep learning approaches particularly conventional neural networks (CNN) using image processing techniques is used to recognize the pattern of traits which also, focus mainly on features like leaf count and leaf tip/base [53]. This is achieved by stacking multiple convolution layers, which apply a convolution operation to the input with a kernel, producing a feature map, and passing it as input to the next layer.

As the network learns, the kernels in each layer are updated to improve the feature maps for the classification task. The initial layers in the network compute primitive features on the image such as corners and edges. The deeper layers use these features to compute more complex features consisting of curves and basic shapes and the deepest layers combine these shapes and curves to create recognizable shapes of objects in the image [53].

2.3 Machine Learning

Commonly used machine learning algorithms are such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. From all of this, the techniques which accurately suit this specific project is Supervised learning.

These machine learning algorithms consist of an output variable (or dependent variable) which is to be predicted from a given set of input variables (or independent variables/predictors). Then using these sets of variables, the model is fitted on training data, which can be used to predict the unseen test data. Also, it is possible to learn from the model, which predictors are important (feature importance) and how the functional relationship between the dependent variable and a certain feature is (Partial dependency plot). The training process continues until the model achieves a desired level of accuracy on the training data and we can optimize the algorithm to correctly recognize and map function for new unseen examples outside from the training data set.

Some of the examples of Supervised Learning are linear regression, decision tree, random forest, KNN, logistic regression, xtreme gradient boosting, etc.

2.3.1 Decision Tree

Decision tree is a non-parametric supervised learning method which is used for both classification and regression tasks and it creates a model that predicts the values or outcomes of a target variable by splitting a data set based on different conditions by learning simple decision rules from the data features or predictors [19].

Also, it is a tree-like graph or a flow chart like structure with nodes where each internal or decision nodes represents a test on a feature, each branch represents the outcome of the test and each terminal or leaf nodes represents a class label and the topmost node in a decision tree is known as the root node like as shown in figure 2.2 & 2.5.

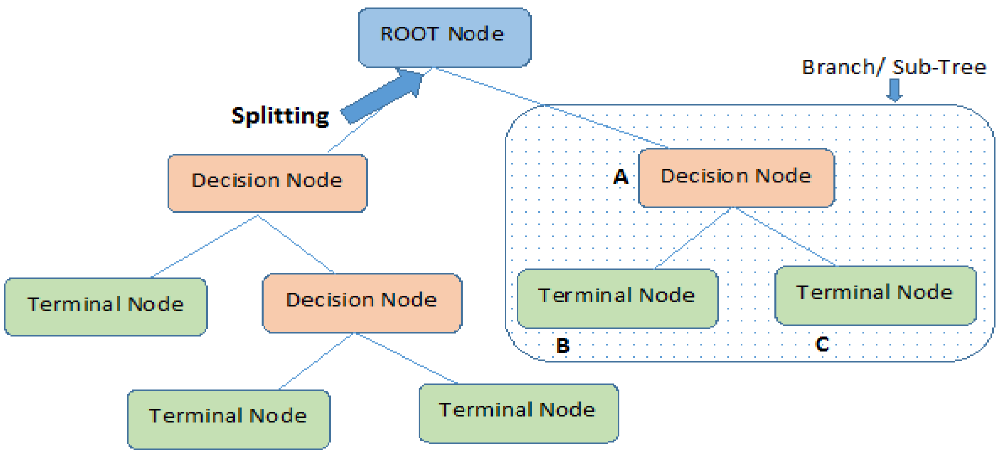


FIGURE 2.2: Simple decision tree [19]

Advantages of Decision tree:

i. Simple to understand, interpret and visualize.

ii. Able to handle numerical very well and requires less time and less effort for data preparation during pre_processing.

iii. Nonlinear relationships between parameters do not affect tree performance.

Disadvantages of Decision tree:

i. Decision-tree learners can create over-complex trees that do not generalize the data well.

ii. Decision trees can be unstable because small variations in the data might result in a completely different tree results.

iii. Does not handle non-numerical data well.

Types of Decision tree:

i. **Categorical Variable Decision Tree:** If the decision tree has a categorical target variable. For example, the yes or no and classification kind of tasks very much comes under the categorical variable decision tree.

ii. **Continuous Variable Decision Tree:** If the decision trees have a continuous target variable. For example, let say the regression type of tasks falls under a continuous variable decision tree.

Also, in this project predicting the target variable (i.e., leaf size) from the set or group of individual features or predictors (lat, lon, altitude, year, day of year). So, in this case it falls under continuous variable decision tree and regression trees are used when the target variable is continuous and it uses the reduction of variance criteria for splitting the node.

In decision tree regression the splitting of the nodes done differently compared to classification tasks and it uses the reduction of variance because it is an impurity criteria suitable for continuous variables and based on this score it further splits the nodes into two or more sub-branches or nodes.

The decision tree regressor will stop split or creating the nodes when it met the conditions as defined in hyperparameters for example such as max_depth and min_samples_split etc.

2.3.2 Ensemble methods

It is a machine learning technique that combines multiple base models into one ensemble model in order to improve the overall performance of the model. These methods are designed to improve the stability and the accuracy of machine learning algorithms by reducing the noise, bias, and variance in the dataset [29, 40].

Advantages of ensemble methods:

i. More accurate prediction results.

ii. Stable and more robust model.

iii. These are able to capture both linear and non-linear relationships in the data.

iv. It can reproduce curve-linear functional relationships.

Disadvantages of ensemble methods:

i. Reduction in model interprets ability due to increasing complexity or it is very difficult to understand or interpret as the complexity of the model increases.

ii. Computational time is very high.

Types of ensemble methods:
The most common and the techniques used in this project are:

- Bagging or Bootstrap Aggregating.
- Boosting

2.3.2.1 Bagging or Bootstrap Aggregating

Bootstrap Aggregating is an ensemble method were random subsets of data are created from the training data. Then, the model is built for each subset sample. Finally, the results of these multiple models are combined using average as shown in figure 2.3. Combinations of multiple models decrease variance, especially in the case of unstable models, and may produce a more reliable prediction than a single model.

In case of Block cross validation first the blocks are created (see section 3.6, appendix A.5, appendix A.7.1) and then every row or data point in datasets is assigned with numbers from 0 to 8 based on latitude and longitude position by creating a new column ′Block′ in the dataset. Now you can see from the (appendix A.8.1 (line 19-25)) first an empty lists with variable ′folds′ is created and every row with value ′0′ under ′Block′ column is appended to one fold let say that is ′fold 0′ and in the next case every row with value ′1′

under 'Block' column is appended to 'fold 1'like that it repeats to the remaining values and their are appended into their respective folds. At the end in that empty lists 'folds' which is created earlier, contains 9 different folds (from fold 0 to fold 8).

Now, when the model fit is performed, one of the fold from list "folds" let say 'fold 0' is used as test data and the remaining folds 1 to 8 are used as training sets and the random subsets are created from this training sets by drawing the samples with replacement from the training sets and it builds the multiple decision trees on them and at the end their predictions are averaged also, at the same time this test values from 'fold 0' are stored in empty list 'ytest' and predicated values to 'ypredic'. In the similar way, in the next iteration let say 'fold 1' is used as a test data and the remaining fold 0 and folds 2 to 8 are used as training sets and it repeats the same thing by creating the random subsets from this training sets and their predictions are averaged at the end and at the same time this test values from 'fold 1' are also stored in list 'ytest' as a extension to the existing test values from the 'fold 0' and predicated values to 'ypredic'.

At the end the list 'ytest' has test values from all folds 0 to 8 and also the list 'ypredic' have predicted values of all folds and the R-square value is calculated using this ytest and ypredic by using the equation eq 3.1.

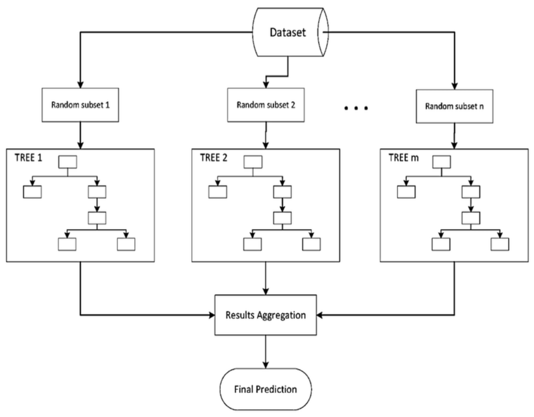


FIGURE 2.3: Bagging [29]

2.3.2.2 Boosting

Boosting is an ensemble method and follows an iterative technique which improves the performance of the model by assigning the higher weightage to the previous incorrect model and in boosting a high number of models is fit with each having a low learning rate.

Boosting is a sequential technique in which, the first algorithm is trained on the entire data set and the subsequent models are built by fitting the residuals of the first algorithm, thus giving higher weight to those observations that were poorly predicted by the previous model see figure 2.4.

Boosting in general decreases the bias error and builds strong predictive models. Boosting has better predictive accuracy than bagging, but it also tends to overfit the training data as well. Thus, parameter tuning becomes a crucial part of boosting algorithms to make them avoid overfitting.

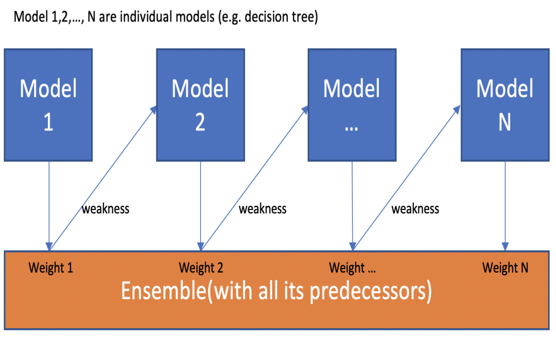

Model 1,2,..., N are individual models (e.g. decision tree)
Model 1
Model 2
Model ...
Model N
weakness
weakness
weakness
Weight 1
Weight 2
Weight ...
Weight N
Ensemble(with all its predecessors)

FIGURE 2.4: Boosting [55]

2.3.3 Random Forest

It is an ensemble model made up of multiple decision trees capable of performing both classification and regression tasks. It builds the each of decision trees on a random subset of observations which are drawn with replacement from original training dataset also known as bootstrap aggregation or bagging and at the end, it average all the decision trees predictions together by combining them to get a more accurate and stable prediction as in figure 2.3. By tuning the parameters (see section 3.7.1), we can improve the accuracy of the model [54]. Figure 2.5 is just an example of one decision tree of many decision trees which built on one of the subset of the observations.

The main principle behind this is that a group of weak learners can come together to form a strong learner. Each model individually, is a weak learner, while all the models together are a strong learner.

The reason for selecting the random forest model [24]:

i. It can handle binary features, categorical features, and numerical features.

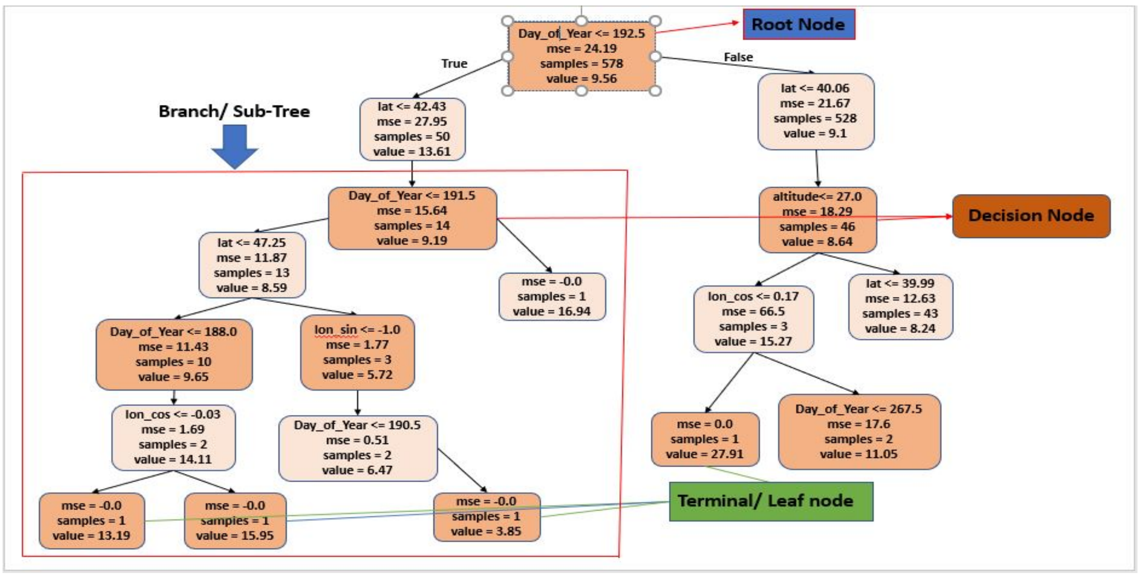

Root Node
Day_of_Year <= 192.5 mse = 24.19 samples = 578 value = 9.56
True
False
Branch/ Sub-Tree
lat <= 42.43 mse = 27.95 samples = 50 value = 13.61
lat <= 40.06 mse = 21.67 samples = 528 value = 9.1
Day_of_Year <= 191.5 mse = 15.64 samples = 14 value = 9.19
altitude<= 27.0 mse = 18.29 samples = 46 value = 8.64
Decision Node
lat <= 47.25 mse = 11.87 samples = 13 value = 8.59
mse = -0.0 samples = 1 value = 16.94
lon_cos <= 0.17 mse = 66.5 samples = 3 value = 15.27
lat <= 39.99 mse = 12.63 samples = 43 value = 8.24
Day_of_Year <= 188.0 mse = 11.43 samples = 10 value = 9.65
lon_sin <= -1.0 mse = 1.77 samples = 3 value = 5.72
lon_cos <= -0.03 mse = 1.69 samples = 2 value = 14.11
Day_of_Year <= 190.5 mse = 0.51 samples = 2 value = 6.47
mse = 0.0 samples = 1 value = 27.91
Day_of_Year <= 267.5 mse = 17.6 samples = 2 value = 11.05
Terminal/ Leaf node
mse = -0.0 samples = 1 value = 13.19
mse = -0.0 samples = 1 value = 15.95
mse = -0.0 samples = 1 value = 3.85

FIGURE 2.5: Sample Random Forest Decision Tree Model

ii. There is very little pre-processing that needs to be done.

iii. These are able to capture both linear and non-linear relationships in the data.

iv. The data does not need to be rescaled or transformed.

Drawbacks:

i. For very large data sets, the size of the trees can take up a lot of memory as well as it takes a lot of computational time.

ii. It can tend to overfit, so you should tune the hyperparameters.

In the case of the random forest regressor, it uses the same criteria to split the nodes as in decision tree but the only difference is by default the Random forest regressor uses the 'MSE'or mean squared error criteria (equation 2.1, where n is no of observations) to split the nodes and which is equal to variance reduction and here the root node is split based on the mse score with the feature which has low variance.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{actual} - \widehat{y_{prediction}})^2 \tag{2.1}$$

2.3.4 XGBoost (eXtreme gradient boosting)

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework [35].

Unlike the Random Forest, it uses a boosting technique where the decision trees are built in sequential wise by doing that each subsequent tree aims to reduce and correct the errors of the previous model until corrections of errors are no longer improving. It also helps to reduce the bias and variance in the dataset [35].

The reason for selecting the XGB model:

i. It can handle missing values in data by using surrogate splits it is one of the inbuilt function XGBoost have [17, 1].

ii. XGBoost on the other hand makes splits up to the max depth specified and then starts pruning the tree backward and remove splits beyond which there is no positive gain [44].

iii. XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization which prevents the model from overfitting [15].

iv. It is also perform at a very high speed with consistency.

v. **Monotonic Constraint:** The XGBoost also have another nice parameter to enforce a monotonic relationship between a feature and the target which makes the limited data models more stable against overfitting issues. Also, it will result in a better model performance on the test data and generalize better [14].

   The weight assigned to the right child must be always higher than the left child In case of positive monotonic and vice versa. If the above condition is not met it abandon the split and goes for the next best split as shown in figure 2.6 [52].

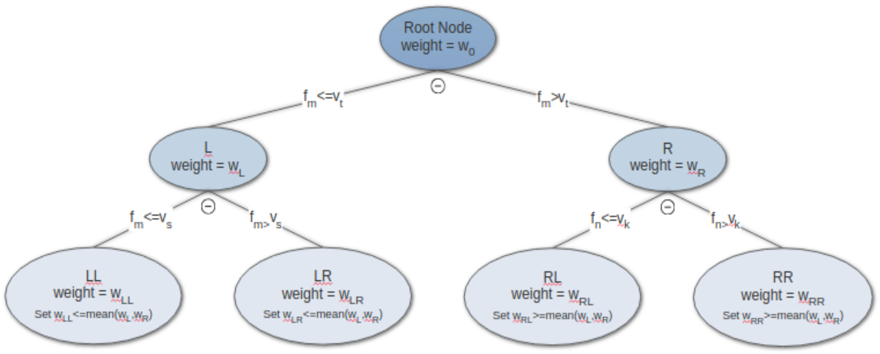


FIGURE 2.6: Monotonic Constraint [52]

Disadvantage:

i. It needs a very careful parameter tuning or else over tuning may cause the overfitting problems in data (see section 3.7.2).

In case of XGBoost regressor, it builds the decision tree in a sequential manner by fitting the residuals from the first algorithm like that it keeps fitting the previous algorithm and it will stop doing that when there are no improvements in the further iterations or results (for more refer 2.3.2.2). Also, while splitting the nodes in the decision tree it uses the RMSE (Root mean square error) criteria and it split the root node based on the least RMSE feature by using equation 2.2, where n is no of observations.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{actual} - y_{\widehat{prediction}})^2} \tag{2.2}$$

2.3.5 Mixed effect Random Forest

The lots of data in wild are clustered and here in this case just to see the variations in Leaf size with respect to changing temporal variables (such as Year, Day of year and annual co2 concentration) for that a mixed effect model is used to separate the variations of leaf size due to temporal effects from the spatial effects and since we are only interested in the temporal effects the eco-regions are used to account or as a substitute for spatial variation as a random process using equation 2.3. Eco-regions are used to represent the large units of water and land containing a distinct assemblage such as endemic species, natural communities [37] and in this case they are used for analysis the impact of variations in leaf sizes of different species due to climate change over the time with respect to different eco-regions as a result of human activities.

The mixed effect random forest (MERF) is the combination of linear mixed effect (LME) and the random forest.

$$y = f(X) + b_i Z + e \tag{2.3}$$

Where as
y - is a target variable.
f(.). - is a non-linear function and this is learned using a random forest. More generally, it can be any non-linear regression model like random forest, gradient boosting trees, or a deep neural network.
X - is the fixed effect features.
b_i - is the random effect coefficient. They are different per cluster i but are assumed to be drawn from the same N(0, sigma_e²) distribution . It is also act as a random slope

and it is achieved by setting the 'Z' matrix or column values of 'Z' to 1 in order to have a random intercept model.
i - is the cluster index.
Z - is the random effect features.
e - is independent, identically distributed (iid) noise. It is distributed as N(0, sigma_e^2).

2.3.6 Spatial autocorrelation

Spatial autocorrelation calculates to what degree the similarity in values between observations in a dataset is related to the similarity in locations of such observations [13].

One of the reasons why spatial auto-correlation is important is because statistics rely on observations being independent of one another. If an autocorrelation exists in a dataset, then this violates the assumption that observations are independent from one another.

As a measure for spatial autocorrelation one can calculate the Moran's I Index using equation 2.4. Moran's I with a positive value up to 1 is treated as positive autocorrelation (it means clustering of similar values), negative values considered as negative correlation (it means it is perfect clustering of dissimilar values) and value which equals to zero then there is no relation exists between the observations [38, 30, 13].

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2} \tag{2.4}$$

N	number of spatial units indexed by i and j
W	sum of all w_ij
w_ij	matrix of spatial weights with zero's on the diagonal (i.e w_ii=0)
x	variable of interest
$\bar{x}$	mean of x

2.3.7 Cross validation

Cross-Validation is a part of the machine learning procedure which is used to split data as training and validation or testing sets. The training set is used to train the machine learning model and the test set is used to test on a new unseen data and this is done by splitting data like 70% - 30% or 80% - 20% or 90% - 10% for example in case of 90%-10% randomly 90% of the data are used for training and remaining 10% for testing. This process will be repeated until each and every data point appears in a test set [41]. By doing this we can also avoid the overfitting and underfitting problems and it is one of the best approaches when we have limited input data as well.

2.3.7.1 Block Selection Cross Validation:

The reasons for choosing block cross validation over all other cross-validation techniques are because, that ecological data often shows spatial, temporal and seasonal and dependence structures in the data and in regular cross-validation techniques where the data is split randomly, which very often tends to perform poorly due to the presence of structure dependencies and that continues as a dependence structure in model residuals, by violating the assumption of independence which can cause overfitting. So, to deal with this issue block selection cross-validation is used where the data is split strategically rather than randomly [39].

In block selection cross-validation in order to increase or maintain the independency, of the data is split into **"blocks"** at some central points of dependence structure such as in time or space by doing that it increases independence, avoid overfitting and provides with more error estimates [39].

However, the blocking strategy must be carefully considered. Blocking in space, time, etc, without knowing it may lead to restricting the ranges or combinations of predictor variables available for model training, thus leads to overestimating interpolation errors.

2.3.8 Hyper Parameter Tuning

The reason for doing parameter tuning is to optimize the performance of the model. Scikit-Learn implements a set of default hyperparameters for all models [2], but these are not optimal for all models and the best way to optimize the settings is to try wide ranges of values with different combinations. Moreover, evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning overfitting where a model performs highly on the training set but poorly on the test set. Therefore, the standard procedure for hyperparameter optimization accounts for overfitting through cross-validation [51].

So in order to overcome this, the **'Randomized search CV'** method is used where the grid of hyperparameter ranges are defined along with randomly sample from the grid, performing K-fold cross-validation which splits the data into k number of subsets called folds. That way, each and every fold get be in test set once and training set as K-1 times and this process continues until each and every fold appears in a test set once [41].

2.3.9 Variation Partitioning

It is a method for partitioning the variation of response data with respect to two or more matrices of explanatory variables [38], this method is used to find out the proportion of variability in leaf size that can be explained by the variation or responses in leaf size due to 1) spatial, 2) temporal and 3) seasonal effects with respect to environmental conditions like for example as shown in figure 2.7 [30].

It allows researchers to estimate or analyze the variations associated with different groups of explanatory variables such as environmental, spatial, temporal, seasonal, and examine their capacity to explain patterns of variation in plant traits. The results of variation partitioning show how much of the variation in the target variable is explained by each group as well as the variations explained by different groups jointly (overlap section or shared part) [30]. So, to do that the following equations are used where:

Now for example, (see appendix A.8.1 also see lines 4 to 49) where the different datasets are organized and an empty dictionary with variable 'd' is created and when the model fit performed first it calculate the R-square value for the 'Common_data' dataset and it save it in the empty dictionary 'd' which created earlier with the same dataset name 'Common_data' and in the next iteration it calculates the R-square for the 'Space' dataset and save it in same dictionary 'd' with same name similarly it repeats for all the datasets.

Now, as you can see from the figure 2.7 (left side) let assume that entire 'A' circle explains the variation in leaf size mostly due to space features but at the same time the variation in leaf size are also partly explained by other groups such as temporal and seasonal. So, in order to get the percentage of explanation only with respect to spatial variables (a (only_space) subtract the R-square value of 'Common_data' dataset with the r-square value of 'climate_change & seasonal' dataset by doing that I get the value which explain the variations in leaf size only due to spatial features. So, for (b (only_climate_change)) subtract the R-square value of 'Common_data' dataset with the r-square value of 'space & seasonal' dataset. In the similar way it was repeated for all other variables to get their respective values.
A is R-square Value of space (latitude, longitude, and altitude)
B is R-square Value of climate_change (year)
C is R-square value of seasonal (day of year)
AB or $(A \cup B)$ is the R-square value of space & climate_change
BC or $(B \cup C)$ is the R-square value of climate_change & seasonal
AC or $(A \cup C)$ is the R-square value of space & seasonal
Totalexplanation or $(A \cup B \cup C)$ is the R-square value of A, B and C together

$a(only\_space) = (A \cup B \cup C) - (B \cup C)$
$b(only\_climate\_change) = (A \cup B \cup C) - (A \cup C)$
$c(only\_seasonal) = (A \cup B \cup C) - (A \cup B)$
$d(only\_space\_\&\_(climate\_change)) = (A \cup B \cup C) - a - b - C$
$e(only\_climate\_change\_\&\_seasonal) = (A \cup B \cup C) - b - c - A$
$f(only\_space\_\&\_seasonal) = (A \cup B \cup C) - a - c - B$
$g(space, climate\_change\_\&\_Seasonal) = (A \cup B \cup C) - a - b - c - d - e - f$
Residuals = 1- $(A \cup B \cup C)$ or 1 - Totalexplanation

For example, the variation partitioning results from fig 2.7 indicates that 8% of the variations in Leaf size was occurred due to seasonal effects and nearly 72% of that was explained by only day of year and the 9% of that was explained jointly by seasonal variable and the 3 spatial variables in the similar way the other 9% was jointly explained by seasonal variable and temporal variable. The negative value or fraction in figure 2.7 indicates that all three effects (spatial, temporal and seasonal and their variables) explains variations in Leaf size (response) as an individual effects better than together also it is not a rightful measure of variations in response and the left out 89% of the (residuals) variations in Leaf size is unexplained by this three variables are related to other environmental variables such as soil, temperatures, etc [30].

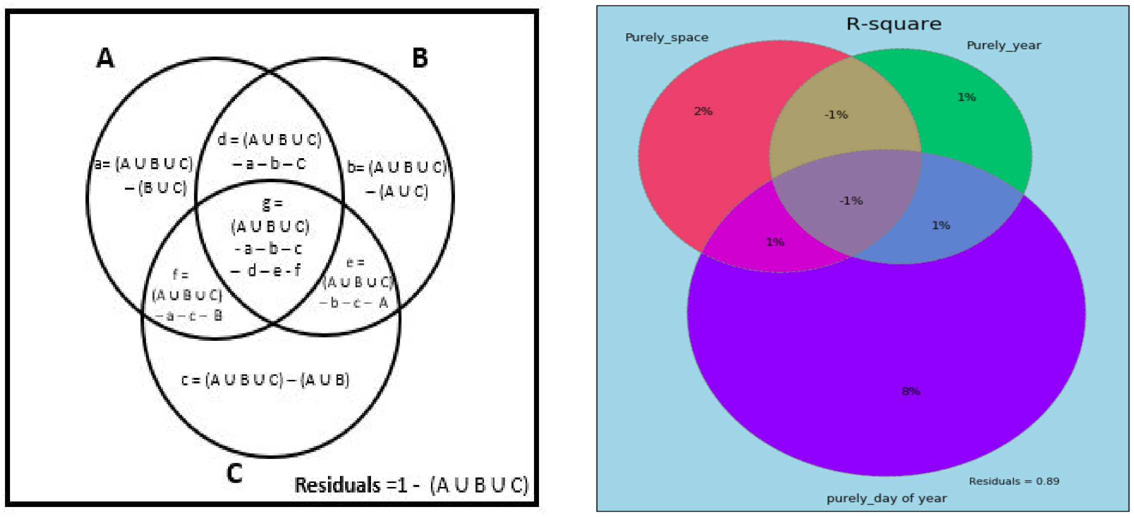

A
B
a= (A U B U C) – (B U C)
d = (A U B U C) – a – b – C
b= (A U B U C) – (A U C)
g = (A U B U C) - a – b – c – d – e - f
f = (A U B U C) – a – c – B
e = (A U B U C) – b – c – A
c = (A U B U C) – (A U B)
C
Residuals =1 - (A U B U C)
R-square
Purely_space
Purely_year
2%
-1%
1%
-1%
1%
1%
8%
Residuals = 0.89
purely_day of year

FIGURE 2.7: Conceptual illustration of partitioning the variation in a response matrix

2.3.10 Data visualization techniques

For the vitalization purpose, I used partial dependency plots, Geo pandas, Feature importance plots as well as other data visualization methods such as histograms, scatter box, and correlation matrix are used.

1. Visualization of machine learning models.

   1.1. **Partial dependency plots (PDP):** Partial dependence plots (PDP) shows the marginal effect that one or set of the target (features or predictors) have on the target response. A partial dependence plot can also show the relationship between the target and a feature such as linear, monotonic, etc., and there are different types of PDP plots like one-way, two-way interaction plots, and PDP box plots [3, 18].

   1.2. **Feature importance plots:** Feature importance is a measure of the effect of the features on the outputs. For each feature, the values go from 0 to 1 where a higher the value means that the feature will have a higher effect on the outputs [32].

2. Visualization of variation partitioning.

   2.1. The matplotlib a package of python is used for implementing the venn diagrams to see up to what proportion the variations in leaf size due to climate change according to three groups such as spatial, temporal and seasonal can be explained as a individual and together.

3. Visualization of spatial data distribution on a world map.

   3.1. **Geopandas:** It works with geospatial data. It implements a geoseries and geodataframes which are the subclasses of pandas series and pandas dataframes. These are mainly used to deal with shapely objects such as points, polygons, and perform geometric operations [4].

Chapter 3

Methodology

In this chapter, we address how we downloaded images, the work with the TraitEx tool, and finally, the steps are taken to create the machine learning model and the different visualization, cross-validation techniques used as well.

Summary of Methods:
The methodology consisted out of different steps, which can be summarized as follows:

- The first part of this project is downloading the herbaria specimen samples and saving them in an organized way based on Image and sample Ids for each image.
- In the second part leaf data is extracted using image recognition technique with the help of semi-automated tool TraitEx then all the sub-data from all sub directories which were created based on Image_id from TraitEx are merged into one main data (into one main file which is saved on species name).
- After that in the next section the data pre-processing of the main data (which contains leaf values from TraitEx) is converted into a dataframe (table-like data structure) it is a more convenient and human-readable way and then the data filtering is done by removing all the unnecessary data.
- In the following sections, we focused on Machine learning algorithms, spatial autocorrelation, cross-validating techniques, Variation Partitioning, data visualization, model accuracy, results, and discussions.
    i. spatial autocorrelation is tested in order to find any correlation exists between the nearest neighbours using latitude and longitude in the data.
    ii. The training and testing data using Block selection cross-validation along with Random Forest and XGBoost.
        (a) The model is fitted using algorithms (Random forest and XGBoost Regressor).

(b) Then the block selection cross-validation is used to obtain realistic R-square values.

iii. The variation partitioning technique is used to see the percentage of explanations given by spatial, temporal, and seasonal features.

iv. Then the data visualization techniques such as partial dependency plots (PDP) and Feature importance plots are used for a better understanding of variations in leaf size.

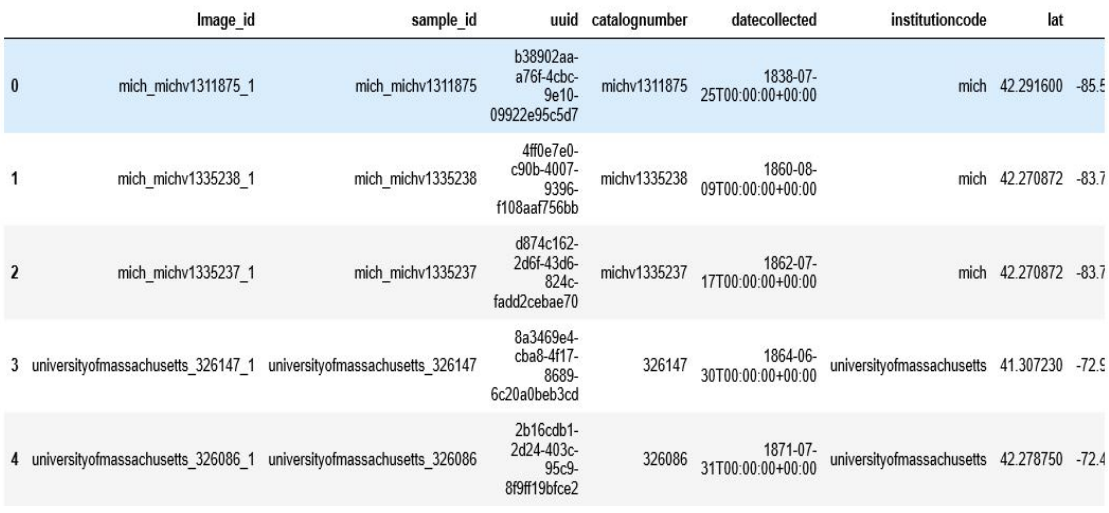
v. Mixed effect random forest model is used for analyzing the variation in leaf size with regarding only temporal features according to climate change.

3.1 Downloading Herbar specimen

The herbar specimen images come from data providers such as Gbif and IDdigbio. It was provided with a metadata information table and corresponding URL's of a combination of Idigbio and Gbif data. This metadata contains information about specimen like latitude, longitude, date of collection, standard institution code, and URLs (for downloading images). Each herbarium is assigned with an official acronym (code) that is used as a standard for referring to the institution and its specimens such as **'Institutioncode'** & **'catalognumber'**.

In order to download the herbar specimen images, a separate **'Image_id'** and **'Sample_id'** columns are created for each row in the dataset based on columns **'Institutioncode'** & **'catalognumber'** (see figure 3.1). The **'Sample_id'** for example it looks like (**cmn_can10896**) (see figure 3.1) is created by removing all the unwanted special characters from columns **'Institutioncode'** & **'catalognumber'** then the **'Image_id'** is created by enumerating **'Sample_id'** and it looks like (**cmn_can10896_1, cmn_can10896_2, ..., cmn_can10896_n**) (see figure 3.1) then by using Urls from column 'accessuri' the images of the herbar specimen are downloaded into specific folders that are created for each image based on Image-id for example (**Species_name/image/image_id.jpg**) (see figure 3.2) for code (see appendix A.1.1).

In some case there are several images exists per **'Image_id'** then in that case the images are downloaded and stored by creating a folder in a sequence of numbers (see fig 3.3 (right image)) and there is one more exceptional case where the exact same images with sequential **'Image_id'** exists (see fig 3.3 (left image)) in that case instead of storing them in folders which created on sequential **'Image_id'** the first image is considered as the original image and it stored and the remaining images are considered as duplicates and dropped (see appendix A.1 line 10).



	Image_id	sample_id	uuid	catalognumber	datecollected	institutioncode	lat	
0	mich_michv1311875_1	mich_michv1311875	b38902aa-a76f-4cbc-9e10-09922e95c5d7	michv1311875	1838-07-25T00:00:00+00:00	mich	42.291600	-85.5
1	mich_michv1335238_1	mich_michv1335238	4ff0e7e0-c90b-4007-9396-f108aaf756bb	michv1335238	1860-08-09T00:00:00+00:00	mich	42.270872	-83.7
2	mich_michv1335237_1	mich_michv1335237	d874c162-2d6f-43d6-824c-fadd2cebae70	michv1335237	1862-07-17T00:00:00+00:00	mich	42.270872	-83.7
3	universityofmassachusetts_326147_1	universityofmassachusetts_326147	8a3469e4-cba8-4f17-8689-6c20a0beb3cd	326147	1864-06-30T00:00:00+00:00	universityofmassachusetts	41.307230	-72.9
4	universityofmassachusetts_326086_1	universityofmassachusetts_326086	2b16cdb1-2d24-403c-95c9-8f9ff19bfce2	326086	1871-07-31T00:00:00+00:00	universityofmassachusetts	42.278750	-72.4

FIGURE 3.1: Sample dataframe

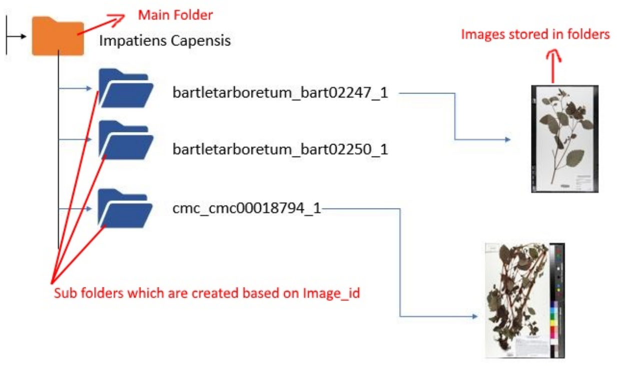

Main Folder
Impatiens Capensis
Images stored in folders
bartletarboretum_bart02247_1
bartletarboretum_bart02250_1
cmc_cmc00018794_1
Sub folders which are created based on Image_id

FIGURE 3.2: Folder and subfolders created based on Image_id

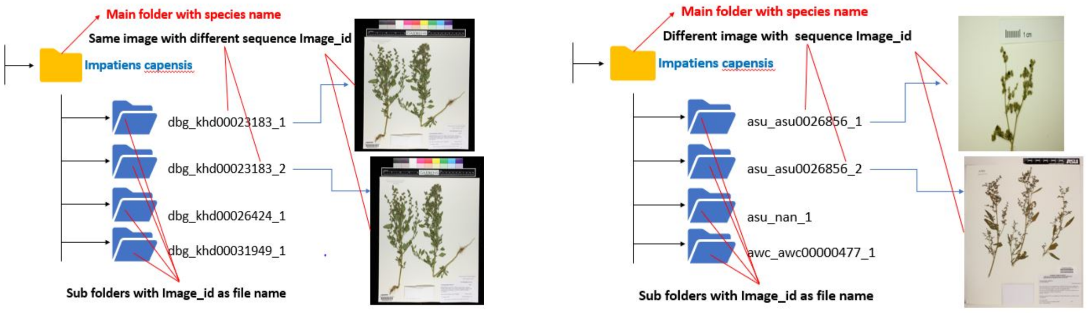

Main folder with species name
Same image with different sequence Image_id
Impatiens capensis
dbg_khd00023183_1
dbg_khd00023183_2
dbg_khd00026424_1
dbg_khd00031949_1
Sub folders with Image_id as file name
Main folder with species name
Different image with sequence Image_id
Impatiens capensis
asu_asu0026856_1
asu_asu0026856_2
asu_nan_1
awc_awc00000477_1
Sub folders with Image_id as file name

FIGURE 3.3: Folder and subfolders created based on Image_id

From here on the process of extracting the leaf measurements or trait data using image recognition is done using the TraitEx tool.

3.2 TraitEx

TraitEx is a semi-automated tool to extract leaf size from herbar specimen which uses deep learning approaches for recognizing the image-based patterns from herbar specimen images and it is used to extract the trait parameters such as leaf length, area, width and petiole length [23, 46]. The complete following things are manually done by the user and the rest of the things are automated (see section 2.2.1).

The first thing to do is selecting the folder which contains the original images using the option **"Select Image"**. After selecting the input folder, all digitized specimen images are loaded in the workspace then the individual image is selected to work with for an example as in figure 3.4.

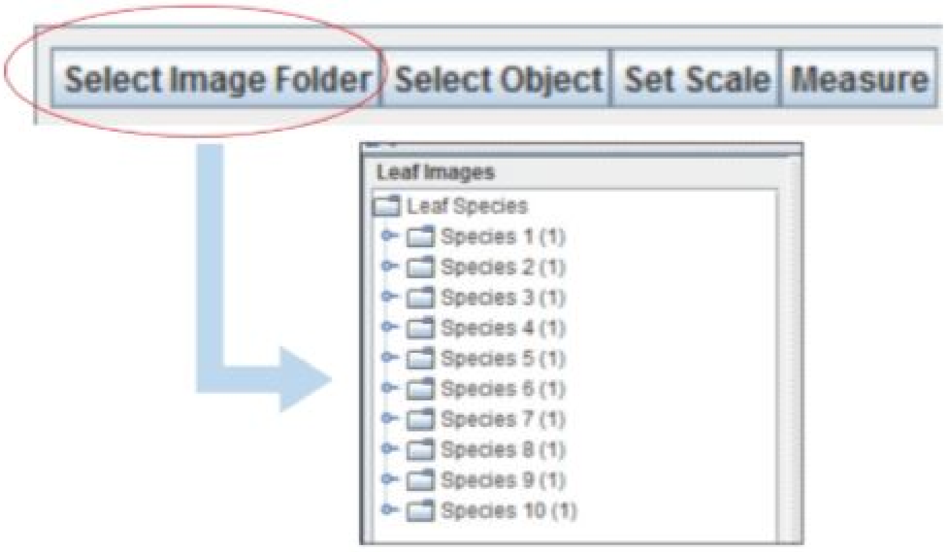


FIGURE 3.4: folders containing digitized specimen images

In the second step setting, the scale and units are required. To measure the traits, the presence of a measuring scale of 10 centimeters in digitized herbarium specimens is needed. Without the scale, the tool cannot measure the trait values. By default, the measurement unit is centimeters then with the help of the **"Set Scale"** option a straight-line is drawn (which is Yellow in colour) on the scale bar of the herbar specimen as shown in figure 3.5. Then the Plot Scale dialog box will open. Here, you can check if the scaling process was done successfully or not as shown in figure 3.6 and this process is repeated for each and every image.

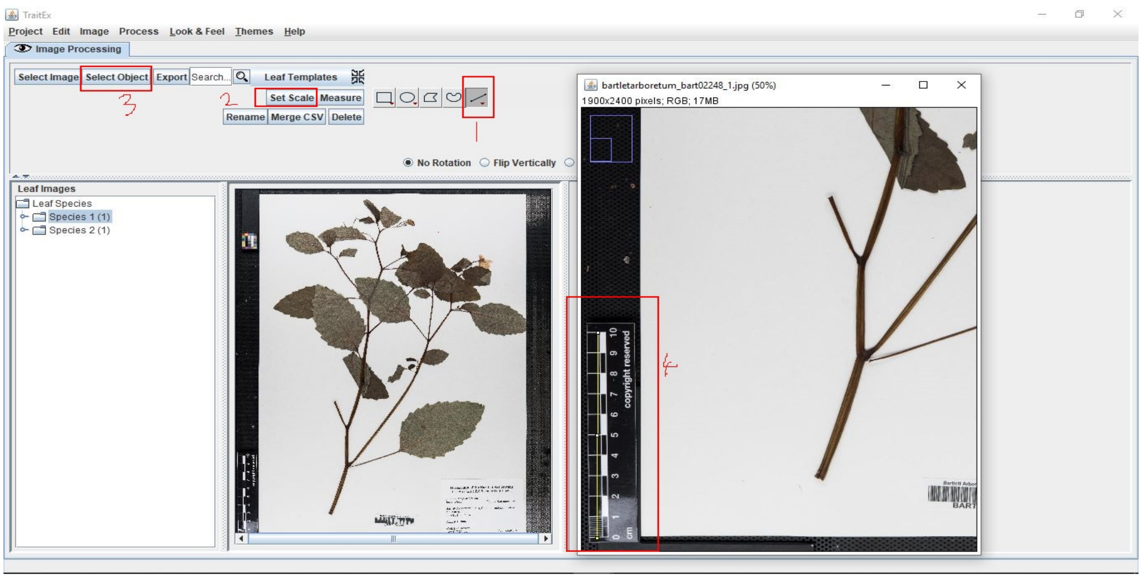
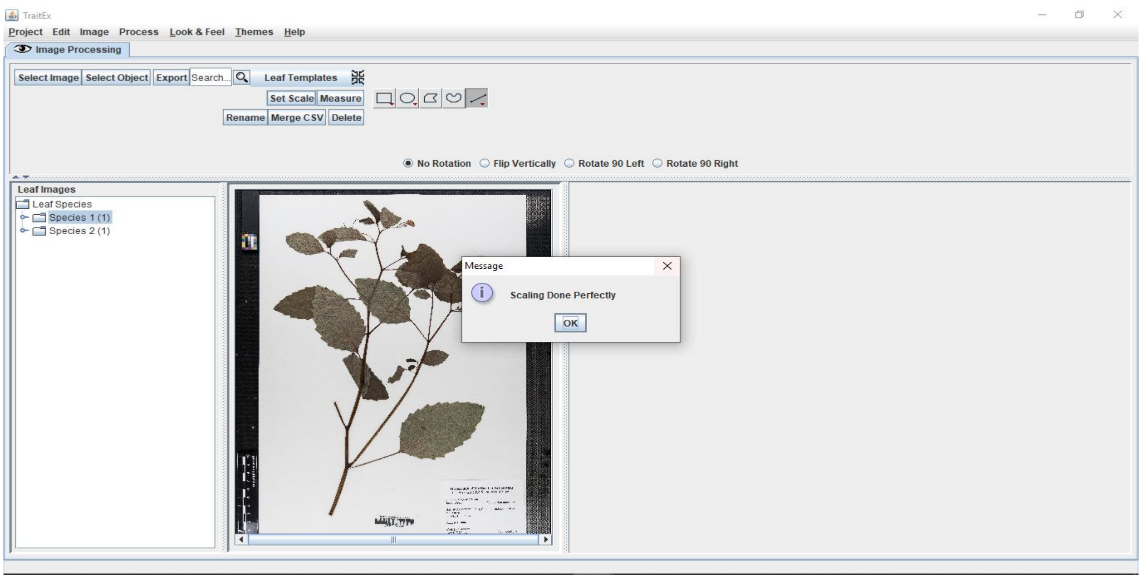


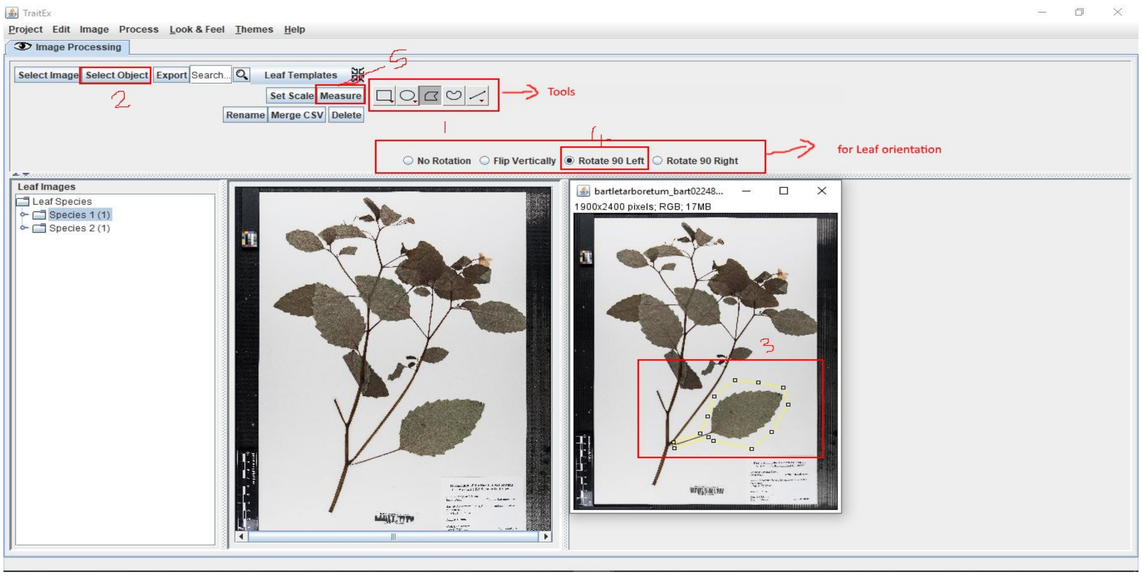
FIGURE 3.5: Steps for setting the scale



Scaling Done Perfectly

FIGURE 3.6: Plot scale dialog box

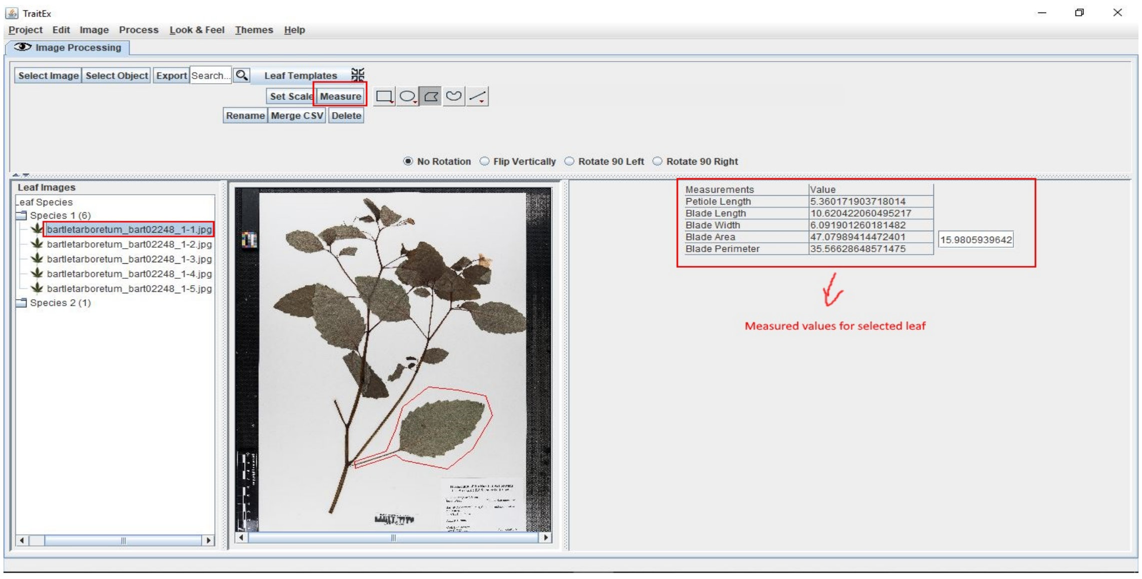
After setting the scale, in order to measure the Leaf open the original image using the **"Select Object"** option. Here, the rectangular, polygon, or freehand tools can be selected based on the interest as shown in figure 3.7 (Shown in YELLOW color).



Chosen setting: No Rotation Flip Vertically [✓ Rotate 90 Left] Rotate 90 Right

FIGURE 3.7: selecting the tools menu

After selecting the leaf that is to be measured and by clicking on the **"Measure"** option, the measurements of the selected leaf will be displayed as a table in the measurement panel and a red bounding box will be drawn in the preview panel as shown in figure 3.8.


TraitEx
Project Edit Image Process Look & Feel Themes Help
Image Processing
Select Image
Select Object
Export
Search...
Leaf Templates
Set Scale
Measure
Rename
Merge CSV
Delete
Leaf Images
Leaf Species
Species 1 (6)
bartletarboretum_bart02248_1-1.jpg
bartletarboretum_bart02248_1-2.jpg
bartletarboretum_bart02248_1-3.jpg
bartletarboretum_bart02248_1-4.jpg
bartletarboretum_bart02248_1-5.jpg
Species 2 (1)
Measured values for selected leaf

Chosen setting: [✓ No Rotation] Flip Vertically Rotate 90 Left Rotate 90 Right

Measurements	Value
Petiole Length	5.360171903718014
Blade Length	10.620422060495217
Blade Width	6.091901260181482
Blade Area	47.07989414472401
Blade Perimeter	35.56628648571475

15.9805939642

FIGURE 3.8: selecting the tools menu

This procedure is repeated for all the images and in each image, I took 5 samples if possible in order to avoid the gap between big and small leaf sizes.

Note: The leaf orientation influences the reliability of the measured traits. To obtain good results, the leaf should be fixed as follow:

i. The petiole should be located at the bottom.

ii. Where-as the blade is oriented upwards. If the selected leaf does not meet these criteria, a set of options are integrated within TraitEx to change the orientation of the leaf (see figures 3.7, 3.8 & 3.9).

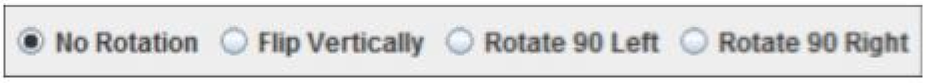

[✓ No Rotation] Flip Vertically Rotate 90 Left Rotate 90 Right

FIGURE 3.9: Change leaf orientation

In some cases, herbarium specimen images contain damaged or mostly overlapping leaves. TraitEx tool is able to deal with damaged leaves and figure 3.10 represents the leaf shape reconstruction option where a menu of leaf templates can be used for outlining the leaf. This option provides a bunch of predefined leaf shape templates used for the task of reconstructing the damaged leaf shape. There are additional commands that may appear when you click on the **'Add shape'** from the menu, which allows the user to add a new leaf to the list.

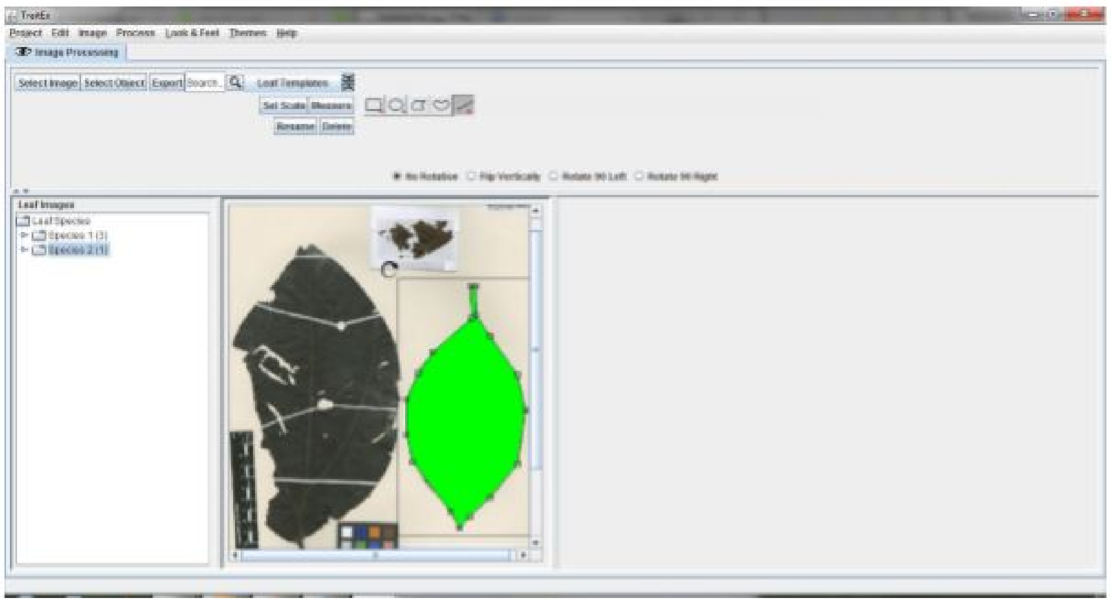

TraitEx
Project Edit Image Process Look & Feel Themes Help
Image Processing
Select Image Select Object Export Search
Leaf Templates
Set Scale Measure
Rename Delete
No Rotation Flip Vertically Rotate 90 Left Rotate 90 Right
Leaf Images
Leaf Species
Species 1 (3)
Species 2 (1)

FIGURE 3.10: Estimating the leaf shape using a leaf template

After extracting the measurements from the digitized images, a sub-folder (named after Image_id like in figure 3.11) associated with the image will be created by TraitEx and contains the separate CSV files, new.CSV, text (.txt), and image of that particular leaf in .jpg format (this files contains the trait values related to that specific leaf measurement) as shown in figures 3.11 & 3.12 and like that for each and every leaf measurement this four files are created and along with that sub-folder also contains a one main CSV which contains appended measured values from all the measurements and one (.txt) text file related to scale of that image which is created automatically by TraitEX tool. Like that, the procedure is repeated for all the images.

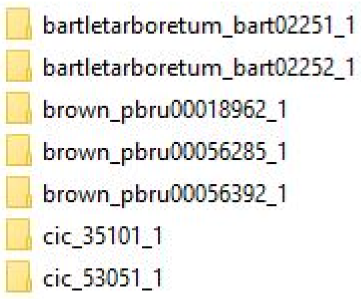

bartletarboretum_bart02251_1
bartletarboretum_bart02252_1
brown_pbru00018962_1
brown_pbru00056285_1
brown_pbru00056392_1
cic_35101_1
cic_53051_1

FIGURE 3.11: Sub folders named based on Image_id

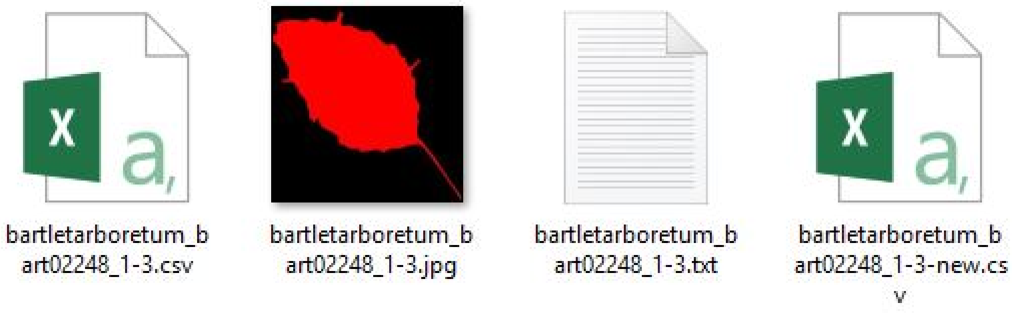
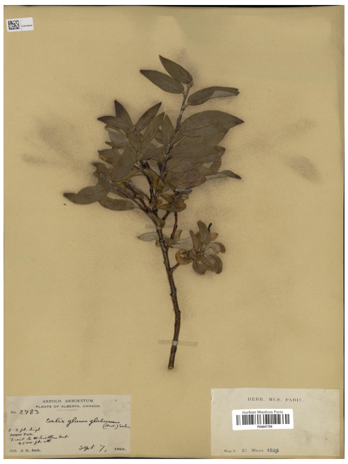
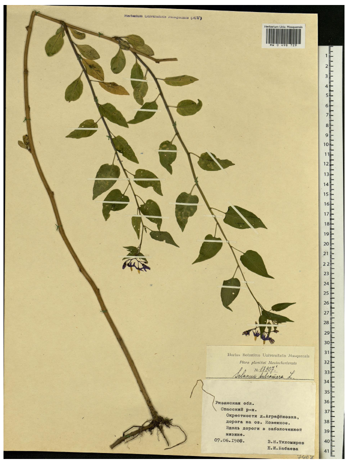

bartletarboretum_bart02248_1-3.csv
bartletarboretum_bart02248_1-3.jpg
bartletarboretum_bart02248_1-3.txt
bartletarboretum_bart02248_1-3-new.csv

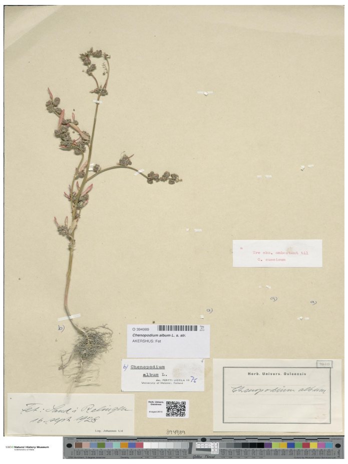
FIGURE 3.12: Folders contains Image, text files, and CSV files

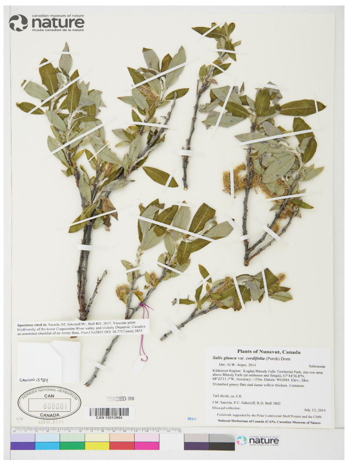
Apart from that, there is also some exceptional case while extracting the measurements from image such as no scale (fig 3.13a), all leaf are overlapped (not possible to measure with TraitEx fig 3.13d), no image (fig 3.13f), sapling (fig 3.13b), Juvenile leaves (fig 3.13e) and no leaves (fig 3.13c).

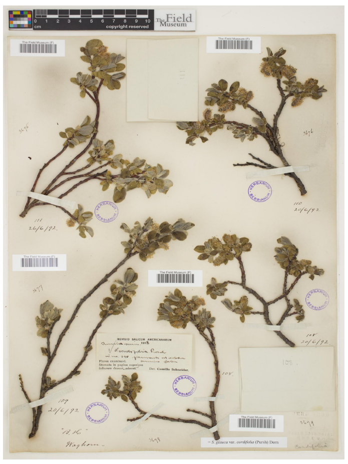
Note: The Juvenile leaves are very small leaves and extracting this leaves doesn't make a fair representation of variations in traits and coming to the sapling these are considered only for tree species and reason for that is if stem contains roots it means the botanists or whoever collected is collected way before the leaves which are not fully grown to their maximum size so this type of leaves also doesn't make a fair representation therefore completely skipped.


(A) No scale


(B) Sapling


(C) No leaves


(D) All leaves are overlapped


(E) Juvenile leaves

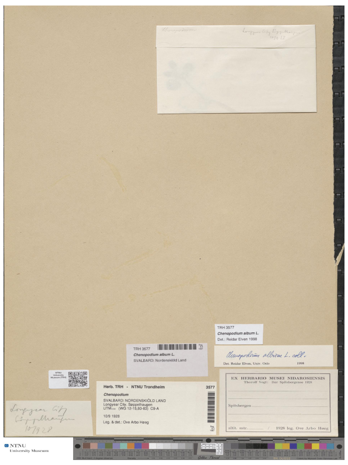
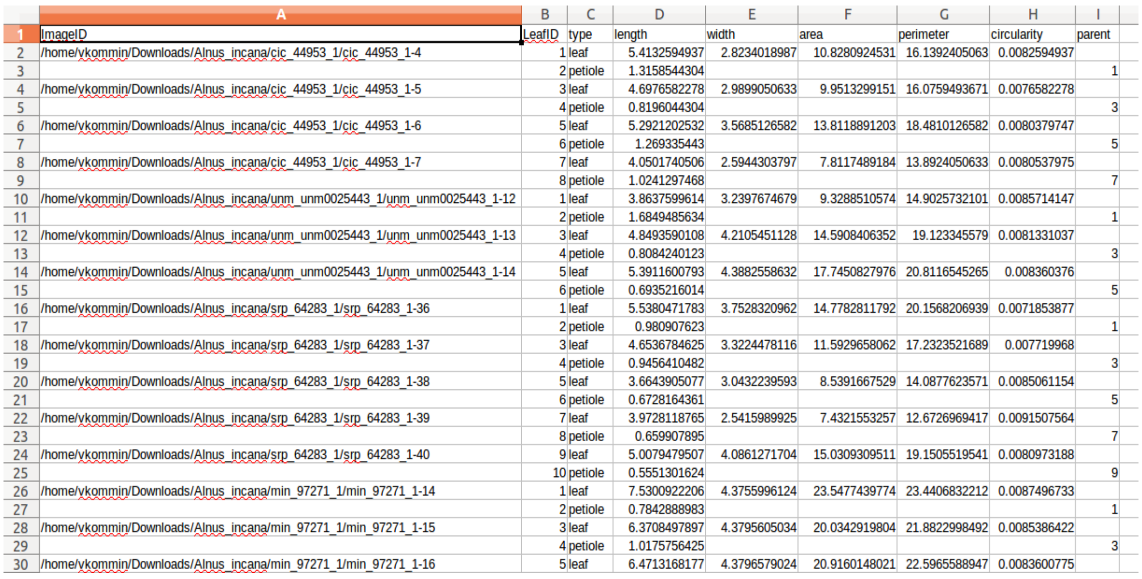

(F) No image

FIGURE 3.13: Exceptional cases for TraitEx Measurement

3.3 Data Pre-Processing

3.3.1 Merging different TraitEx output files

In the data pre-processing all the CSV files in the sub-folders, that each sub folder (see fig 3.11) contains one main CSV files. So, hereby using the python code (see appendix A.2) only main CSV files from each and every sub folder are merged into one final CSV and those files which don't have any CSV files are completely skipped and the final main file looks like as shown in the figure 3.14.



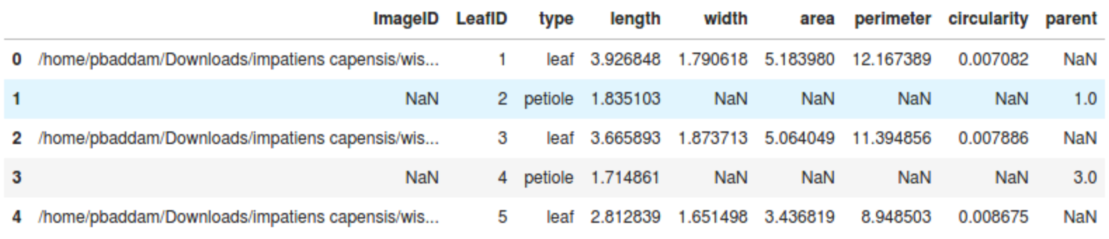
	A	B	C	D	E	F	G	H	I
1	ImageID	LeafID	type	length	width	area	perimeter	circularity	parent
2	/home/vkommin/Downloads/Alnus_incana/cic_44953_1/cic_44953_1-4	1	leaf	5.4132594937	2.8234018987	10.8280924531	16.1392405063	0.0082594937	
3		2	petiole	1.3158544304					1
4	/home/vkommin/Downloads/Alnus_incana/cic_44953_1/cic_44953_1-5	3	leaf	4.6976582278	2.9899050633	9.9513299151	16.0759493671	0.0076582278	
5		4	petiole	0.8196044304					3
6	/home/vkommin/Downloads/Alnus_incana/cic_44953_1/cic_44953_1-6	5	leaf	5.2921202532	3.5685126582	13.8118891203	18.4810126582	0.0080379747	
7		6	petiole	1.269335443					5
8	/home/vkommin/Downloads/Alnus_incana/cic_44953_1/cic_44953_1-7	7	leaf	4.0501740506	2.5944303797	7.8117489184	13.8924050633	0.0080537975	
9		8	petiole	1.0241297468					7
10	/home/vkommin/Downloads/Alnus_incana/unm_unm0025443_1/unm_unm0025443_1-12	1	leaf	3.8637599614	3.2397674679	9.3288510574	14.9025732101	0.0085714147	
11		2	petiole	1.6849485634					1
12	/home/vkommin/Downloads/Alnus_incana/unm_unm0025443_1/unm_unm0025443_1-13	3	leaf	4.8493590108	4.2105451128	14.5908406352	19.123345579	0.0081331037	
13		4	petiole	0.8084240123					3
14	/home/vkommin/Downloads/Alnus_incana/unm_unm0025443_1/unm_unm0025443_1-14	5	leaf	5.3911600793	4.3882558632	17.7450827976	20.8116545265	0.008360376	
15		6	petiole	0.6935216014					5
16	/home/vkommin/Downloads/Alnus_incana/srp_64283_1/srp_64283_1-36	1	leaf	5.5380471783	3.7528320962	14.7782811792	20.1568206939	0.0071853877	
17		2	petiole	0.980907623					1
18	/home/vkommin/Downloads/Alnus_incana/srp_64283_1/srp_64283_1-37	3	leaf	4.6536784625	3.3224478116	11.5929658062	17.2323521689	0.007719968	
19		4	petiole	0.9456410482					3
20	/home/vkommin/Downloads/Alnus_incana/srp_64283_1/srp_64283_1-38	5	leaf	3.6643905077	3.0432239593	8.5391667529	14.0877623571	0.0085061154	
21		6	petiole	0.6728164361					5
22	/home/vkommin/Downloads/Alnus_incana/srp_64283_1/srp_64283_1-39	7	leaf	3.9728118765	2.5415989925	7.4321553257	12.6726969417	0.0091507564	
23		8	petiole	0.659907895					7
24	/home/vkommin/Downloads/Alnus_incana/srp_64283_1/srp_64283_1-40	9	leaf	5.0079479507	4.0861271704	15.0309309511	19.1505519541	0.0080973188	
25		10	petiole	0.5551301624					9
26	/home/vkommin/Downloads/Alnus_incana/min_97271_1/min_97271_1-14	1	leaf	7.5300922206	4.3755996124	23.5477439774	23.4406832212	0.0087496733	
27		2	petiole	0.7842888983					1
28	/home/vkommin/Downloads/Alnus_incana/min_97271_1/min_97271_1-15	3	leaf	6.3708497897	4.3795605034	20.0342919804	21.8822998492	0.0085386422	
29		4	petiole	1.0175756425					3
30	/home/vkommin/Downloads/Alnus_incana/min_97271_1/min_97271_1-16	5	leaf	6.4713168177	4.3796579024	20.9160148021	22.5965588947	0.0083600775	

FIGURE 3.14: Final CSV File

3.3.2 Data Filtering

As mentioned earlier, the final CSV file needs to be converted into a two-dimensional array with rows and columns like a dataframe (table-like data structure) which is more convenient and human-readable.

So as you can see from the figure 3.15 the petioles are in every alternate row of dataframe and it is necessary to remove them from a row and add as a new column with name 'Petiole_length' and as well as removing all the unnecessary columns such as 'LeafID', 'type', 'parent' and 'circularity' and renaming the columns **'length', 'width', 'area', 'perimeter'** as **'Leaf_length', 'Leaf_width', 'Leaf_area', 'Leaf_perimeter'**. Further, from the **'ImageID'** column all the unnecessary text is removed and the new dataframe with only necessary words and numbers with a column named as **'Image_id'** is created (see fig 3.16) refer code in section (appendix A.3).

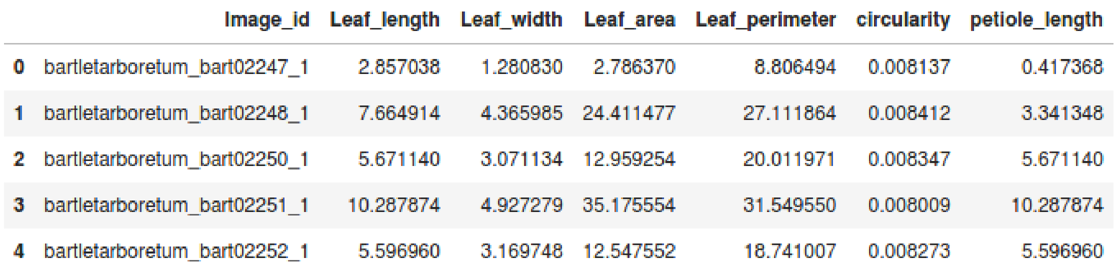


	ImageID	LeafID	type	length	width	area	perimeter	circularity	parent
0	/home/pbaddam/Downloads/impatiens capensis/wis...	1	leaf	3.926848	1.790618	5.183980	12.167389	0.007082	NaN
1	NaN	2	petiole	1.835103	NaN	NaN	NaN	NaN	1.0
2	/home/pbaddam/Downloads/impatiens capensis/wis...	3	leaf	3.665893	1.873713	5.064049	11.394856	0.007886	NaN
3	NaN	4	petiole	1.714861	NaN	NaN	NaN	NaN	3.0
4	/home/pbaddam/Downloads/impatiens capensis/wis...	5	leaf	2.812839	1.651498	3.436819	8.948503	0.008675	NaN

FIGURE 3.15: Example dataframe

3.3.3 Performing Groupby based on Image_id

As you can see from fig 3.14 the dataframe contains multiple measurements per **'ImageID'**. So, the groupby is performed based on the most commonly used statistic measure **'mean & median'**. But, in this project, the median is preferred more over mean and the reason for that is because, mean may not be a fair representation of the data, because the average is easily influenced by outliers very small or large values in the data set that are not typical and on the other hand the median is the point at which there is an equal number of data points whose values lie above and below the median value. Thus, the median is truly the middle of the data set and the file is saved as **'pkl'** file, and the final dataframe looks as in figure 3.16.



	Image_id	Leaf_length	Leaf_width	Leaf_area	Leaf_perimeter	circularity	petiole_length
0	bartletarboretum_bart02247_1	2.857038	1.280830	2.786370	8.806494	0.008137	0.417368
1	bartletarboretum_bart02248_1	7.664914	4.365985	24.411477	27.111864	0.008412	3.341348
2	bartletarboretum_bart02250_1	5.671140	3.071134	12.959254	20.011971	0.008347	5.671140
3	bartletarboretum_bart02251_1	10.287874	4.927279	35.175554	31.549550	0.008009	10.287874
4	bartletarboretum_bart02252_1	5.596960	3.169748	12.547552	18.741007	0.008273	5.596960

FIGURE 3.16: Final dataframe

3.3.4 Merging orginal pkl with pkl file with leaf values

The original pkl file is merged with the pkl file which contains leaf values and then further data pre-processing such as converting **'datecollected'** column into a date and time format for extracting the year and day of year (see appendix A.7.1 lines 6-8) by removing all the unnecessary characters, text and numbers from it (see appendix A.4 lines 12-16).

Also, assigning the blocks & altitude (Digital elevation model) to the existing dataframe based on latitude & longitude (see appendix A.7.1 lines 14-33), converting the longitude to cyclic feature in order to get the realistic predictive outcome otherwise, the algorithm considers it has a two end points (see appendix A.7.2 lines 11-12) and removing the outliers from the dataset using Z-score test technique where the values which are greater than or less than the threshold will be identified and removed from dataset by treating them as an outliers (see appendix A.7.2 lines 17-20).

3.4 Testing for spatial autocorrelation

Spatial autocorrelation as explained in section 2.3.6 to find out the relation or degree of similarity between the near by observations in a given location, the Moran's I index is used. Moran's I is a correlation coefficient that measures the overall spatial autocorrelation by using equation 2.4.

In this case to check the spatial autocorrelation a software related to spatial data analysis called **GeoDa** [16] is used where the Moran's I is calculated based on the wide range of distance (like 5, 10, 50, 111) kilometers in every direction and those observations which are under this ranges are considered as neighbours and calculated Moran's I at each and every range [16].

3.5 Machine Learning Methods and algorithms

In this project python's scikit-learn package is used to implement the random forest regressor [5] and XGBoost regressor [26].

3.5.1 Random forest regressor

The model is fitted on the independent features or predictors such as latitude, longitude (with sin and cosine transformation as a cyclic feature), altitude, year, day of year and the target feature or variable is leaf area and the predicted leaf area values are generated and r-square value is calculated also meanwhile, similarly, the model fit is performed on different combinations of datasets for the better understanding of variations pattern in leaf size and their respective r-square values are calculated (see appendix A.8.1). The r-square of different datasets which are calculated are also used for variation partitioning to explain the variations caused by the different predictor groups such as spatial, temporal and seasonal.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{actual} - \widehat{y_{prediction}})^2}{\sum_{i=1}^{n}(y_{actual} - y_{mean})^2} \tag{3.1}$$

For example, In this case this is how it works let say the original training dataset looks like (figure 3.17a) which as 5 independent features with 5 sample data points, one target feature and random forest builds a multiple decision trees and each of them builds on a random subset of observations like in fig (3.17b,3.17c, & 3.17d) by drawing the samples repeatedly with replacement from original training dataset also know as bagging technique so that means each data point in each feature have an equal chance of getting selected and at the same time the tree which built on subset have no of observations or samples size is same as the original training dataset but the samples are drawn randomly that means same sample are used multiple times in the same subset might possible.

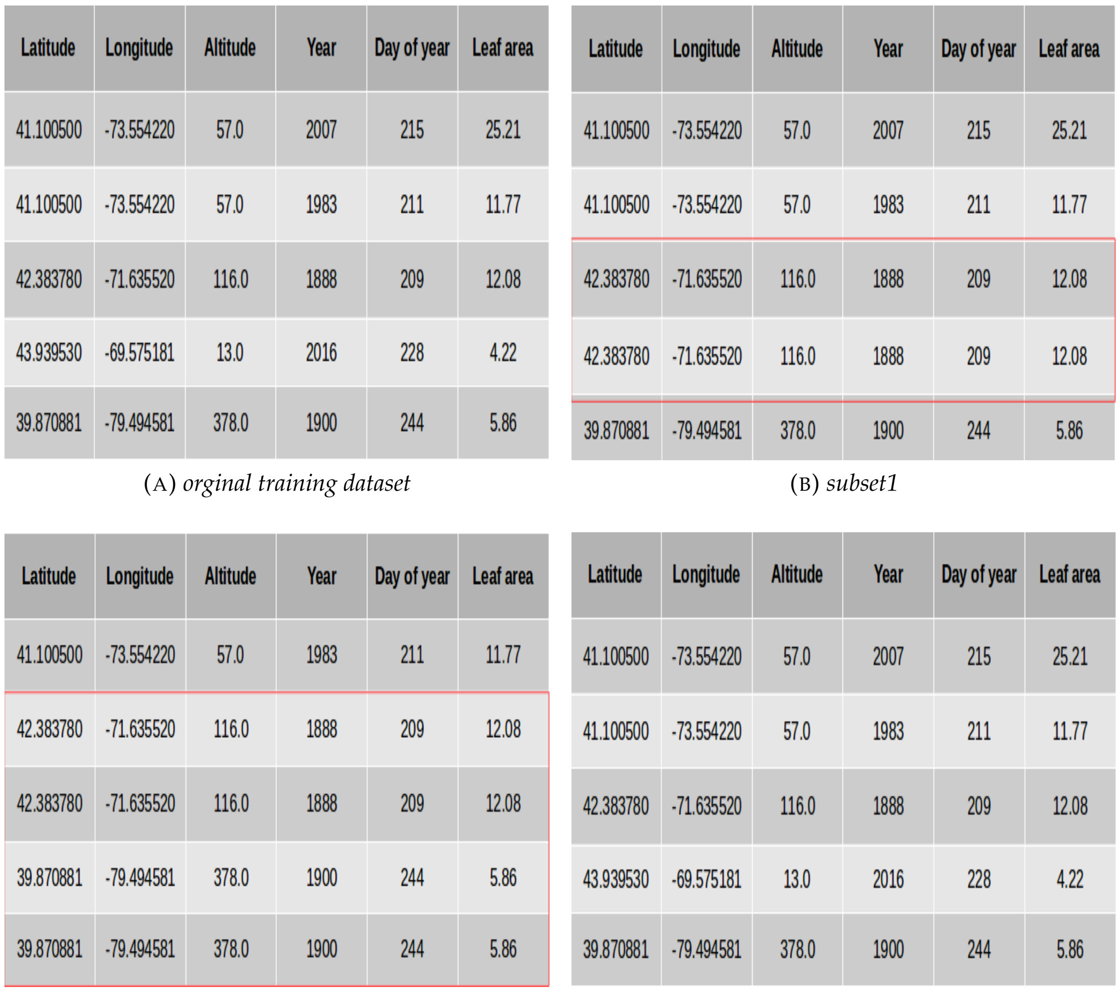
The number of decision trees as well as the subsets on which the decision trees are build is defined in hyperparameter setting 'no_estimators'. For example if the 'no_estimators' set to 3 it means it build the 3 decision tree models and each decision tree model built on respective subsets 1, 2 and 3 like in fig (3.17b,3.17c, & 3.17d) and the data points in this subsets are drawn with replacement from the original training dataset and for example under latitude column in subset 1 have same data points multiple times (see fig (3.17b) in total they have 5 data points but 2 of them are same latitude points and same things happen to the respective rows from the other columns or features too and at the end it average all the decision tree models outputs by using ensemble technique to give the final prediction just like in figure 2.3.

But it is also important to mention that the decision tree builds on the random subset of feature by splitting the nodes and at the each node based on the "max_features" which defined in hyperparameters as 'auto' or 'sqrt' in the case if the max_features = 'auto' is defined. So, this means at the each node before splitting it will consider all features in the subset and it split on feature which as low mean square error and it repeats the similar procedure at each and every node, and this is how it do it first it predict the target values based on the individual feature and then it calculates the mean sqaure error for the predicted target value and actual target value for that particular feature using equation 2.1 and it repeats same thing with every feature in the subset.

Also, at the same time it looks for the best split point within the feature by trying different threshold values between that feature (predictor) and the target feature (actual target value) to predict the target value and then it calculated the mean square error for each and every data point in that threshold and the threshold which has the minimum mean square error is used as best split and it is considered as a candidate for the root node and this process repeats at each and every node. Like in (figure 2.5) where the day of year has a least mean square error out of all features in the dataset, and at the threshold value of 'day of year<= 192.5' the mean square error is low so this split point is considered to be a best split point and a best feature to split on at root node.

Now, Unlike the 'auto', when in hyperparameters if 'max_features= 'sqrt' is defined then it only consider the square root of features in that subset in simply words if there is 4 features in the subset then the square root of 4 is 2 so, at the each node first it randomly consider 2 features out of 4 from the subset and for that 2 random features it starts calculating mean square error and it split the node on feature which as least mean square error and it repeats the same thing at each and every node.

The nodes are splited into further nodes on the feature which has least mean square error and it repeats the same thing at the each node for further splitting the nodes until it reaches the terminal or leaf node where it can't further split the nodes for example like hyperparameter settings 'min_samples_split'=10 or when it reaches the maximum depth as defined in hyperparameter settings like 'max_depth'= 10.



Latitude	Longitude	Altitude	Year	Day of year	Leaf area
41.100500	-73.554220	57.0	2007	215	25.21
41.100500	-73.554220	57.0	1983	211	11.77
42.383780	-71.635520	116.0	1888	209	12.08
43.939530	-69.575181	13.0	2016	228	4.22
39.870881	-79.494581	378.0	1900	244	5.86

(A) *orginal training dataset*

Latitude	Longitude	Altitude	Year	Day of year	Leaf area
41.100500	-73.554220	57.0	2007	215	25.21
41.100500	-73.554220	57.0	1983	211	11.77
42.383780	-71.635520	116.0	1888	209	12.08
42.383780	-71.635520	116.0	1888	209	12.08
39.870881	-79.494581	378.0	1900	244	5.86

(B) *subset1*

Latitude	Longitude	Altitude	Year	Day of year	Leaf area
41.100500	-73.554220	57.0	1983	211	11.77
42.383780	-71.635520	116.0	1888	209	12.08
42.383780	-71.635520	116.0	1888	209	12.08
39.870881	-79.494581	378.0	1900	244	5.86
39.870881	-79.494581	378.0	1900	244	5.86

(C) *subset2*

Latitude	Longitude	Altitude	Year	Day of year	Leaf area
41.100500	-73.554220	57.0	2007	215	25.21
41.100500	-73.554220	57.0	1983	211	11.77
42.383780	-71.635520	116.0	1888	209	12.08
43.939530	-69.575181	13.0	2016	228	4.22
39.870881	-79.494581	378.0	1900	244	5.86

(D) *subset3*

FIGURE 3.17: subset1, 2 and 3

Then in further cases, the block cross-validation (see section 3.6), hyperparameter tuning (see section 3.7) is done and then variation partitioning (see section 3.8) is implemented.

Also, the feature importance plots (see appendix A.11), partial dependency plots refer (appendix A.12) for code and spatial distribution of data (see appendix A.13) for all species are provided.

Note: The R square value can be negative when the sum and square value of regression see the nominator of equation 3.1 is greater than sum and square value of Total see denominator of equation 3.1. This happens when the predicted values of target variables are not close to the actual values of the target variable and the reason for that is model doesn't follow the trend due to the limited data.

3.5.2 XGBoost Regressor

The same whole procedure which is done with random forest regressor (such as model fitting) is also repeated with XGBoost and R-square values of different species are calculated and compared with Random forest regressor R-square values (see code in appendix A.10.1). But here, the parameter tuning is done for some different parameters (see appendix A.10.2) also see 3.7.2. Meanwhile, the monotonic constraint is also performed with XGBoost to make the model more stable against overfitting issues and to increase the model performance and the r-square value with and without monotonic constraint is also calculated for different datasets combinations.

3.6 Block Selection Cross Validation

It is a part of machine learning used to split data as training and testing sets for obtaining independent predicted and observed values on which basis realistic R_square values can be calculated.

As mentioned earlier in section 2.3.7.1 the block cross-validation splits the data strategically despite random by splitting the data into blocks at some central points across the spatial structure itself (as a contiguous geographic space). So by doing that it forces to test on more spatially distant records as well as it decreases spatial dependence by considered the nearby observations either into training set or test set and based on different data and objectives the different type of blocking approaches are necessary [39].

In this project, the blocks of 360 x 720 grids with block sizes of 0.5 degrees are created on two spatially variables such as latitude and longitude.

First, the blocks are generated as 360 x 720 (rows and columns) if the block size is 0.5 and then all those empty blocks are assigned with specific values from 0 to 8. For that, in this case, 9 x 9 Blocks (see figure 3.18) are used to fill all those 360x720 blocks (see figure 3.19) and then indices are created for each row and column based on the block, block_size, latitude, and longitude for code refer (appendix section A.5).

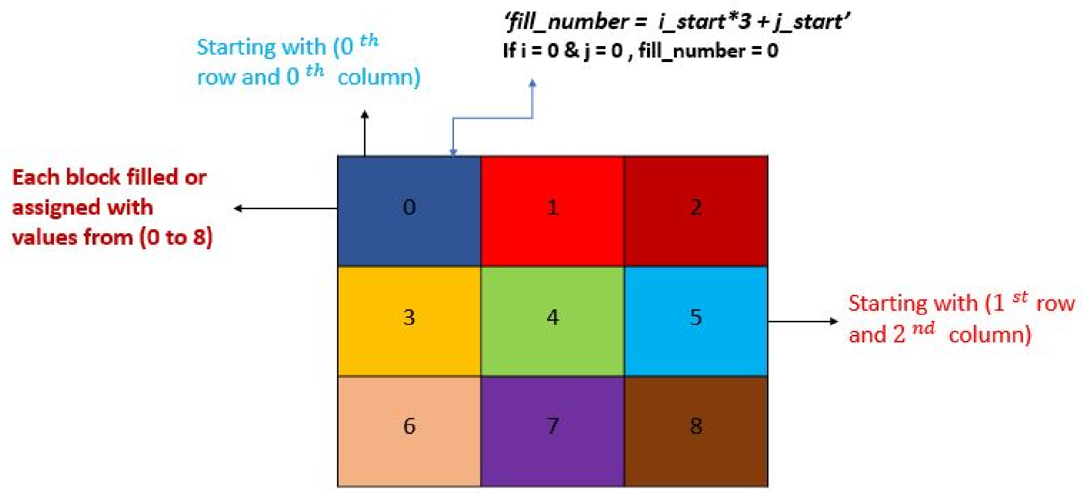
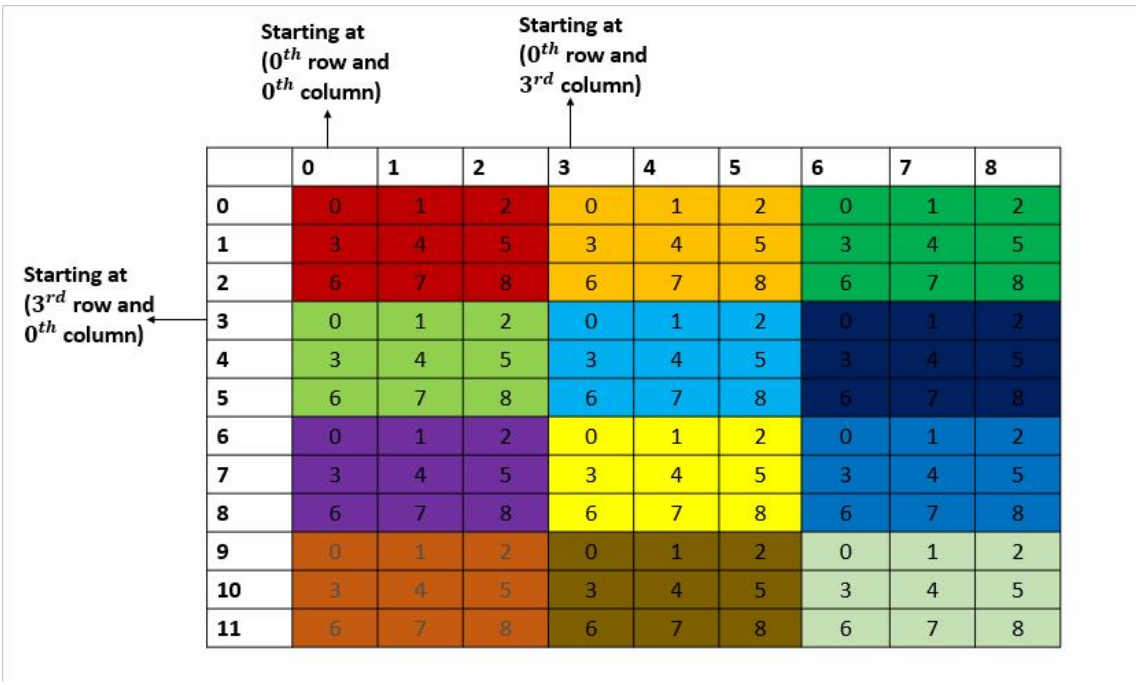

'fill_number = i_start*3 + j_start'
If i = 0 & j = 0 , fill_number = 0
Starting with (0 th row and 0 th column)
Each block filled or assigned with values from (0 to 8)
0
1
2
3
4
5
6
7
8
Starting with (1 st row and 2 nd column)

FIGURE 3.18: example for 9 x 9 block


Starting at (0th row and 0th column)
Starting at (0th row and 3rd column)
Starting at (3rd row and 0th column)

	0	1	2	3	4	5	6	7	8
0	0	1	2	0	1	2	0	1	2
1	3	4	5	3	4	5	3	4	5
2	6	7	8	6	7	8	6	7	8
3	0	1	2	0	1	2	0	1	2
4	3	4	5	3	4	5	3	4	5
5	6	7	8	6	7	8	6	7	8
6	0	1	2	0	1	2	0	1	2
7	3	4	5	3	4	5	3	4	5
8	6	7	8	6	7	8	6	7	8
9	0	1	2	0	1	2	0	1	2
10	3	4	5	3	4	5	3	4	5
11	6	7	8	6	7	8	6	7	8

FIGURE 3.19: Example matrix

Finally, the block cross-validation along with algorithm is implemented with 9 different folds and each fold contains test & train data and the test sets from all 9 folds are stored as an extended list under variable **'ytest'** in the similar way the predicted values from all the 9 folds are stored as an extended list under variable **'ypredic'** and the final r-square score is calculated using equation 3.1 for this ytest (observed value or test sets from all folds) and the ypredic (predicted values from all folds) see code in (appendix A.8.1).

3.7 Hyper Parameter Tuning

As mentioned in section 2.3.8 the main reason for parameter tuning is to optimize the performance of the model to get the better R square value. So to do that **'Randomized search CV'** method is used from scikit-learn package [2] where parameter grids with wide ranges of values with different combinations are defined and k-fold cross-validation is performed until each and every fold appears in a test set once see appendix A.6 and the best parameters with best values are taken for example (see appendix A.6 Listing A.10.)

3.7.1 Random forest Regressor

When doing the hyperparameter tuning only the most important parameters are selected and tuned such as following for Random forest in order to avoid overfitting [51].

- n_estimators = number of trees in the forest.
- max_features = max number of features considered for splitting a node.
- max_depth = max number of levels in each decision tree used to control over-fitting as higher depth will allow the model to learn relations very specific to a particular sample.
- min_samples_split = min number of data points placed in a node before the node is split.
- min_samples_leaf = minimum number of samples required to be at a leaf node.
- min_weight_fraction_leaf = minimum weighted fraction of the sum total of weights required to be at a leaf node.
- bootstrap = method for sampling data points (with or without replacement)
- random_state = Controls both the randomness of the bootstrapping of the samples used when building trees (if bootstrap=True) and the sampling of the features to consider when looking for the best split at each node (if max_features < n_features).

3.7.2 XGBoost Regressor

The hyperparameter tuning process in case of XGBoost should be done carefully otherwise it will lead to overfitting or poor performance of the model (the wide ranges of values for tuning parameters should be defined carefully)[27] and it also uses some of the same features that random forest has. For example, out of all the parameters booster parameters are the most important parameters and these parameters are tuned [27] such as (see appendix A.10.1).

- min_child_weight = the minimum sum of weights of all observations required in a child. Higher values prevent a model from learning relations which might be highly specific to the particular sample selected for a tree. Too high values can lead to under-fitting hence, it should be tuned using CV.
- max_leaf_nodes = maximum number of terminal nodes or leaves in a tree.
- gamma = a node is split only when the resulting split gives a positive reduction in the loss function. Gamma specifies the minimum loss reduction required to make a split.
- subsample = denotes the fraction of observations to be randomly samples for each tree the lower values make the algorithm more conservative and prevents overfitting but too small values might lead to under-fitting the typical values are in the range of 0.5-1.
- learining_rate = makes the model more robust by shrinking the weights on each step and the common values ranges are between 0.01-0.2.
- colsample_bytree = denotes the fraction of columns to be randomly samples for each tree and the typical value range is 0.5-1.

3.8 Variation Partitioning

From the matplotlib package the venn diagrams [6, 47] for the variation partitioning are implemented by specifying the values as described in code (see appendix A.9 listings A.14). Further, these variations are separated in order to see the variations in target variable explained by each individual variables such as spatial variables (lat, lon, alt), climate_change variable (year) and seasonal effects variable (day of year) and plotted the Venn diagram with respect to the percentage of explanation (refer appendix A.9 listing A.15) and also refer (appendix A.9 listings A.16).

3.9 Mixed effects Random Forest

As mentioned in section 2.3.5 specifically in this case interested in the temporal effects and for that mixed effect random forest model (MERF) is implemented from the python's MERF package [25, 42].

As mentioned earlier in this case eco-regions [37] are used as a random process for that the data related to eco-regions for all species are downloaded from the Terrestrial eco-regions of the World (TEOW) and then using the geopandas library the species dataset is spatially joined to the TEOW dataset based on geometry refer (appendix A.14.1 lines 1-31). By doing that, only those eco-regions where the spatial distribution of species data exist are considered and remaining are left out. Meanwhile, the mean annual CO2 data is download from the Historical Co2 datasets (also known as CO2.earth) [7], and merged with species dataset based on year (A.14.1 lines 47-50) and the reason for using CO2 is explained under objectives or see section 1.2.

Then everything else like in random forest is repeated here such as block cross-validation, hyperparameter tuning except but in this case the model is fitted on year, day of year and mean annual CO2 as a fixed features, as usual, leaf areas as a target feature, the random effect feature (eco-regions) represents the spatial effects, for which the ECO_id's are used as clusters and the predicted leaf area values are generated and r-square value is calculated (see appendix A.14.3) also the model fit is performed by leaving out each feature and r-square value is calculated then feature importance plots, the histogram of random effects and partial dependency plots are plotted under (appendix A.14.4).

From, the leaving out feature after calculating the r-square value the mean annual CO2 is removed and the MERF model is again performed with year and day of year features as a fixed effect features because the CO2 does not have any significant impact on leaf size variation for all species as well as at the same time the day of year is transformed into extraterrestrial radiation based on the location in order to account for the problems of having samples from northern and southern hemispheres (see appendix A.14.2).

3.10 Species and their data distribution

Chenopodium album:
Chenopodium album which is also known as a white goosefoot is a shrub (see figure 3.20) and it is widely distributed in both northern and southern hemispheres, occurring from sea level to altitudes of 3600m, and from latitudes 70°N to more than 50°S as in figure 3.21 and it grows mostly in temperate and sub temperate regions such as in Asia as a

vegetable or grain and also behave as a summer annual, mostly viewed as an agricultural weed in Europe, North America as well as other parts of the world also as a fast-growing weedy plant also behave as a winter annual in cold climate regions. It is a common weed of almost all cultivated crops, gardens, horticultural crops and orchards also found on wasteland, in pastures and strips of uncultivated land, and along roadsides and riverbanks [8].

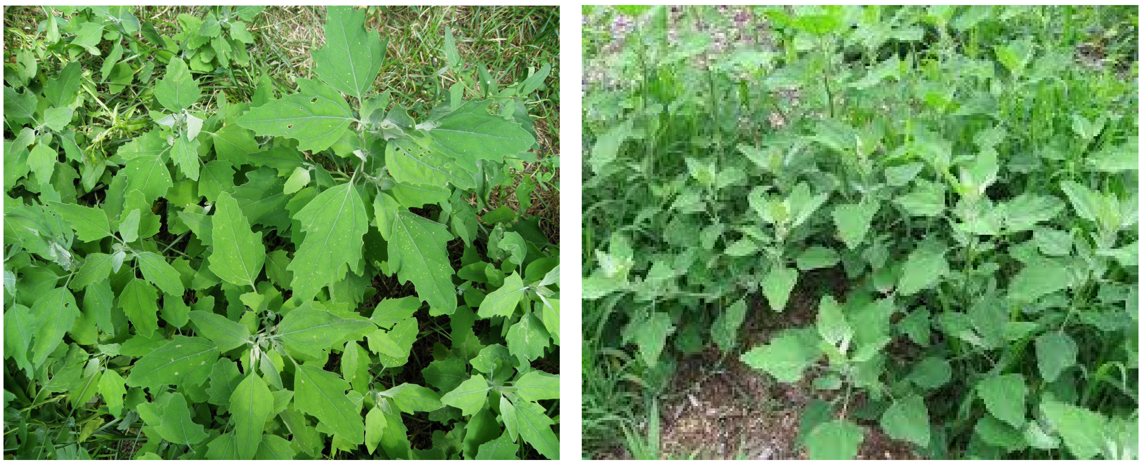
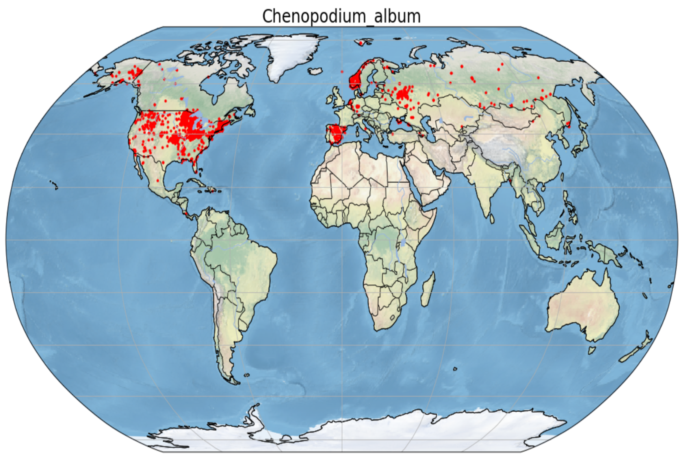
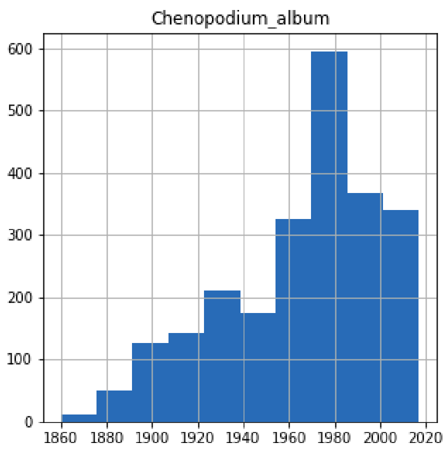


FIGURE 3.20: *Chenopodium album* [8]



(A) Spatial distribution



(B) Histogram of temporal data

FIGURE 3.21: Spatial distribution of herbar specimen for *Chenopodium album* and the histogram of temporal data

Impatiens capensis:

Impatiens capensis is a herb species which are native to North America and spread across from North America to Central America (see figure 3.23 [9]). It is common in bottomland soils, ditches, and along creeks, often growing side-by-side with its less common relative, yellow jewelweed like in figure 3.22 which is most commonly used as domestic weed and it leaves juice are used for external wounds, skin burns, etc however it is considered dangerous for internally use [12].

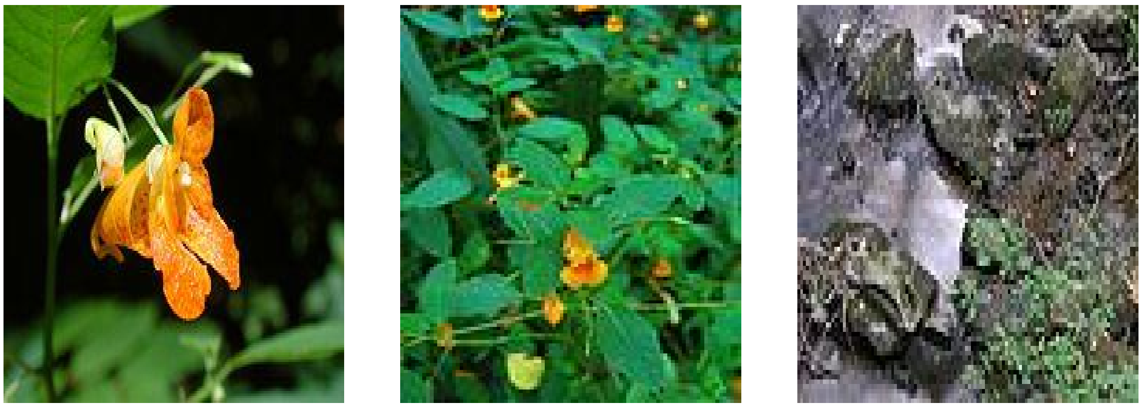
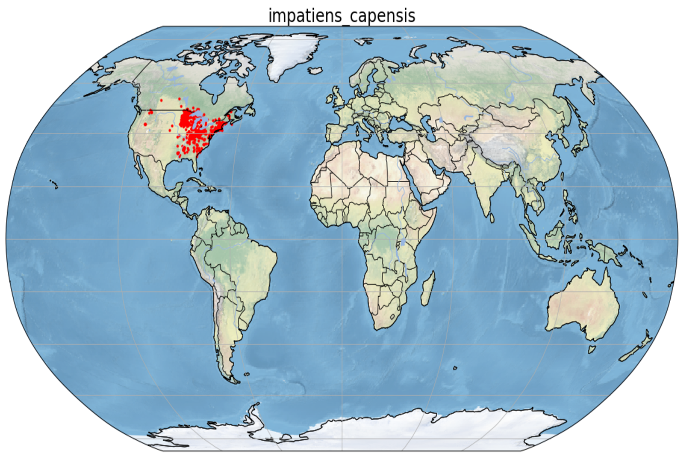
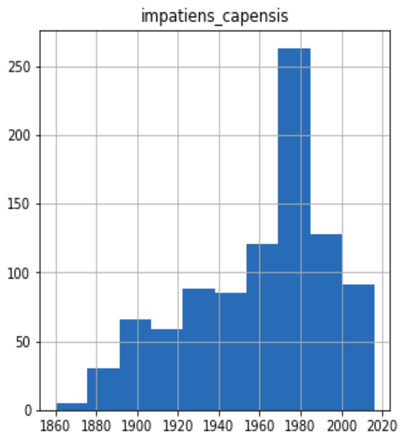


FIGURE 3.22: Orange Jewelweed with leaves and flowers growing on a creek side [12]



(A) Spatial distribution



(B) Histogram of temporal data

FIGURE 3.23: Spatial distribution of herbar specimen for *Impatiens capensis* and the histogram of temporal data

Salix glauca:
Salix glauca is a shrub species of flowering plants in the willow family also known as grey leaf and white willow as in figure 3.24. It is native to North America, where it occurs throughout much of Alaska, northern and western Canada, and the contiguous United States south through the Rocky Mountains to northern New Mexico (see figure 3.25). It can also be found in Greenland, northwestern Europe, and Siberia. In a suitable habitat it can grow to a tree upto 6 meters high also grows in forests, alpines, and subalpine's climates [28].

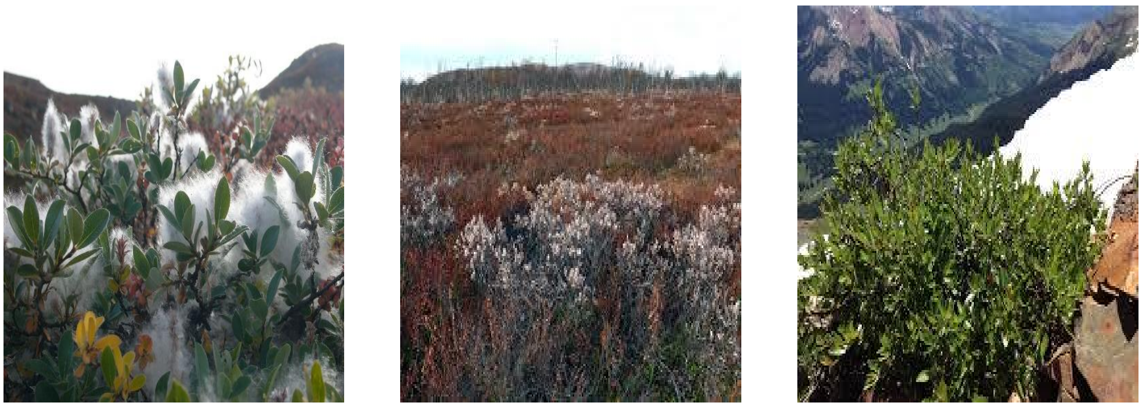
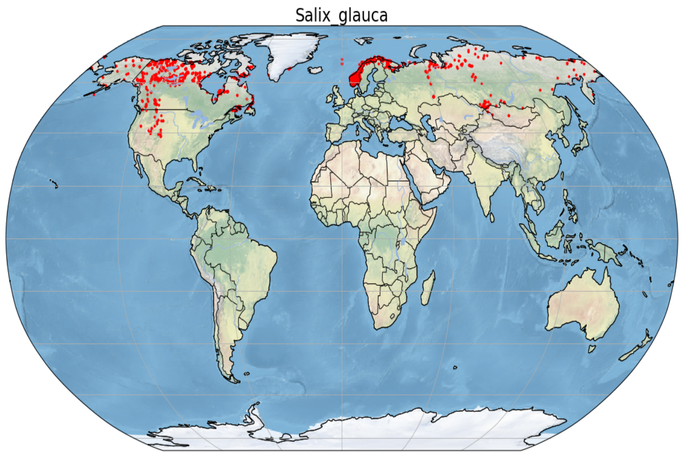
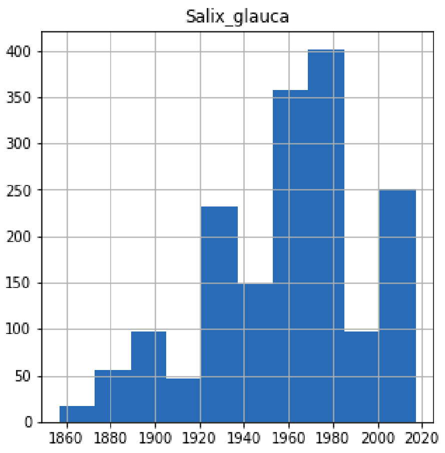


FIGURE 3.24: *Salix glauca* with grey leaf willow and seed on grey leaf willow [28]



(A) Spatial distribution



(B) Histogram of temporal data

FIGURE 3.25: Spatial distribution of herbar specimen for *Salix glauca* and the histogram of temporal data

Solanum dulcamara:
Solanum dulcamara is the shrub species (see figure 3.26) which is native to the North American continent, and it grows on banks, in damp bushes, in riparian forests, on forest fields, scree slopes and moist locations in Europe and Asia (see figure 3.27). It is also used as a remedy for cold, fever, many more and the dried 2 to 3 years old stems are used as a medicinal purpose [36].

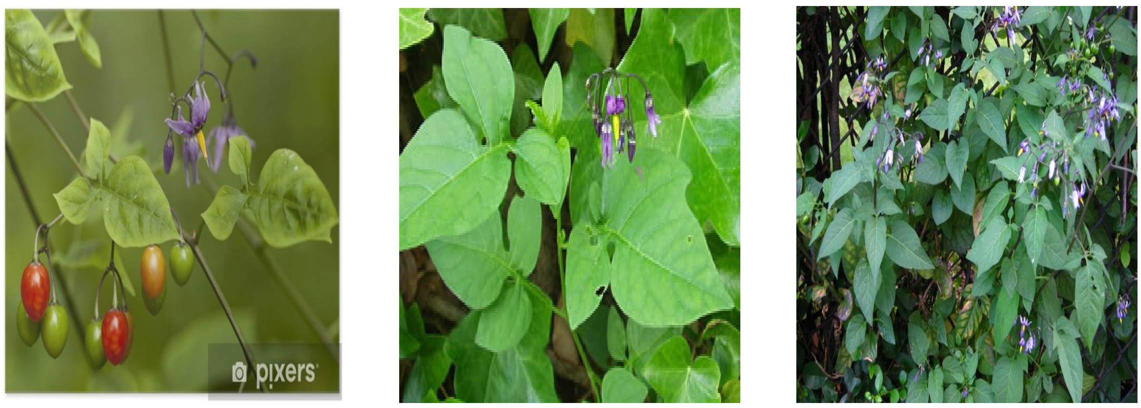
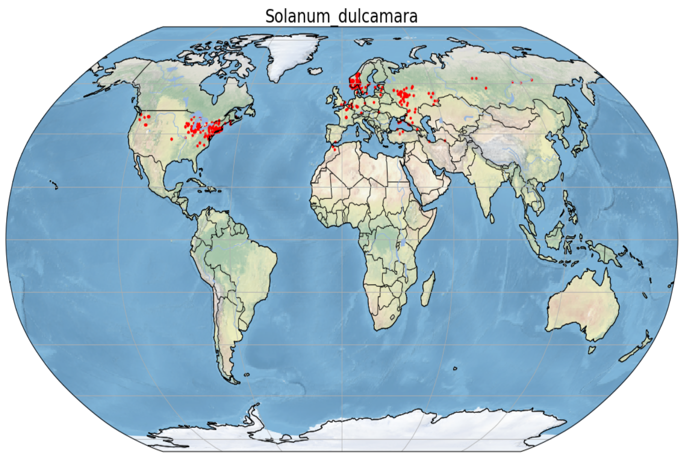
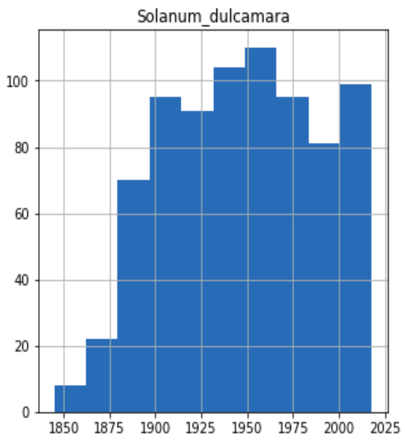


FIGURE 3.26: *Solanum dulcamara* [10, 11]



(A) Spatial distribution



(B) Histogram of temporal data

FIGURE 3.27: Spatial distribution of herbar specimen for *Solanum dulcamara* and the histogram of temporal data

Chapter 4

Results

In this chapter, the results for each species with different dataset combinations from both Random forest regressor, XGBoost regressor, and Mixed effect random forest as well as the feature importance plots, Pdp, variation partitioning are provided. The r-square values for leaf area with mean, median, maximum, and minimum are also calculated, and out of all those leaf areas, the leaf area with median seems a fair representation and the r-square value is high for that and the r-square values for remaining leaf area are not provided.

4.1 Results on Random Forest

TABLE 4.1: R-square values for *Salix glauca* species (shrub and tree).

***Salix glauca* R-square values**	
Datasets	**Random Forest Regressor**
Common_data	0.19
Space	0.17
Climate_change (Year)	-0.05
Seasonal (Day of Year)	-0.12
Climate_change & Seasonal	-0.05
Space&Climate_change	0.17
Space& Seasonal	0.18

Feature importance plots of *Salix glauca* (see fig 4.1) shows that the variations in leaf sizes are mostly due to spatial especially longitudinal.

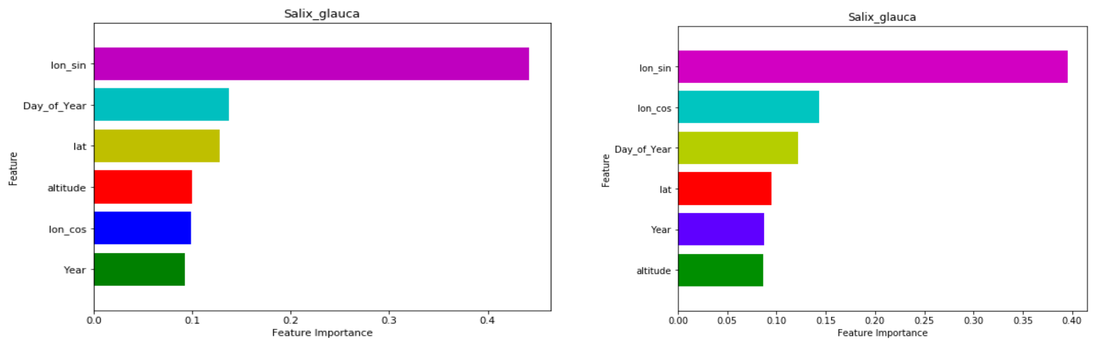


FIGURE 4.1: Feature importance plots for *Salix glauca* using R-square value of Random forest on left and XGBoost on right

As you can see from PDP plots of *Salix glauca* (see fig 4.2) (it is shrub but in proper conditions, it can grow as a tree) the leaf size is decreasing at higher latitude, altitudes, and it tends to have a slightly larger leaf size before the 1900s and then leaf sizes are remained unchanged until early 2000s and with the increasing days of year the leaf sizes are increasing and the reason for that is a leaf out starts (ie. in between June to August) also it reaches to its maximum size by the end of August and again it starts falling or shedding at its mature stage and it happens in early autumn which is in between September to November and it repeats the same procedure every year but this increasing trend in the leaf size with respect to day of year shows that the samples are collected at different stages of the leaf growth.

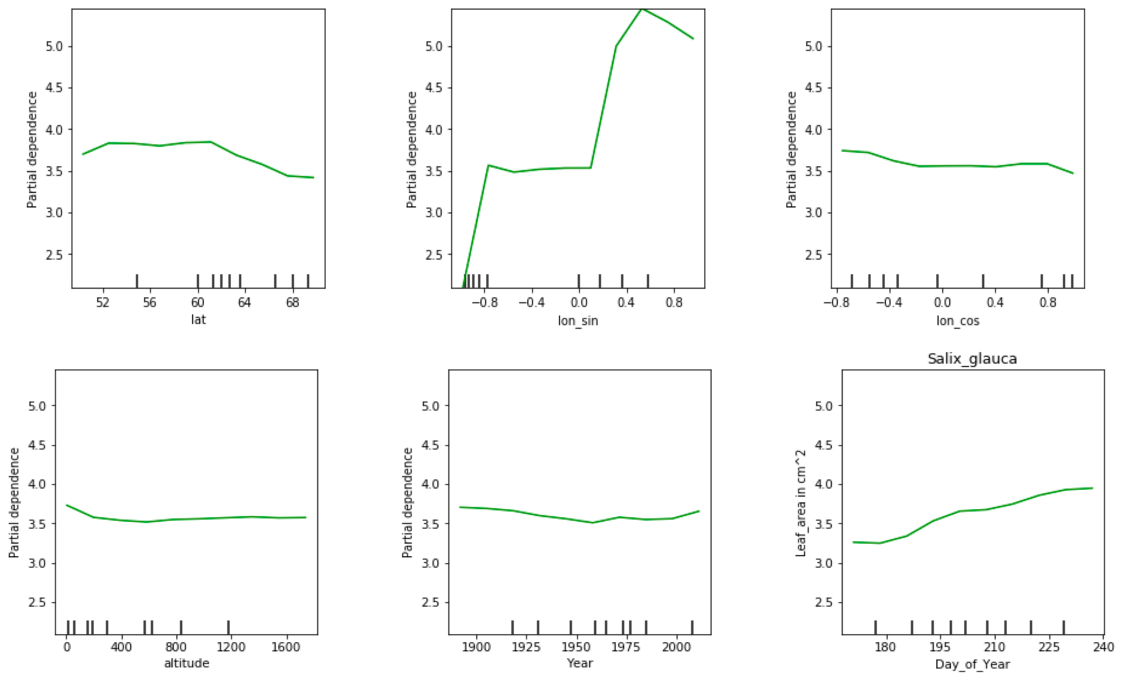


FIGURE 4.2: PDP plots for *Salix glauca* using Random forest

TABLE 4.2: R-square values for *Chenopodium album* species (it can be herb, shrub, or small tree).

***Chenopodium album* R-square values**	
Datasets	**Random Forest Regressor**
Common_data	0.033
Space	-0.012
Climate_change (Year)	-0.06
Seasonal (Day of Year)	-0.05
Climate_change & Seasonal	-0.02
Space&Climate_change	0.009
Space& Seasonal	0.023

Feature importance plots of *Chenopodium album* (see fig 4.3) shows that the variations in leaf sizes are mostly due to spatial and seasonal especially day of year, longitude and latitude, and this species.

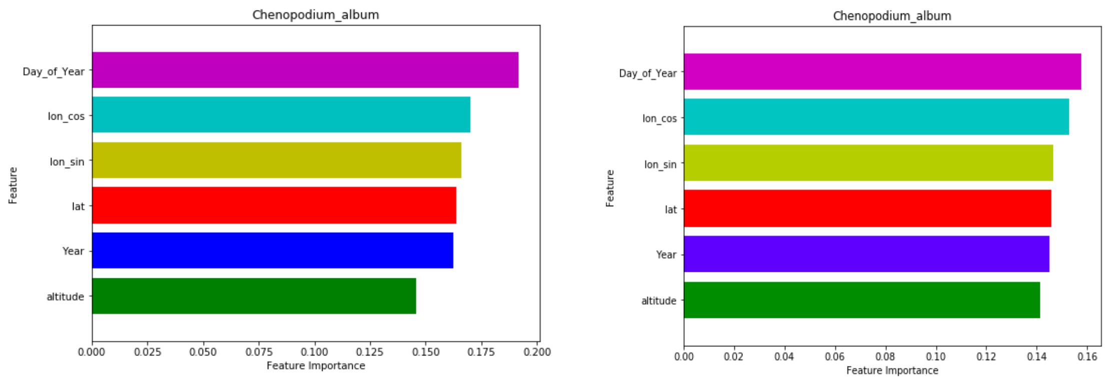


FIGURE 4.3: Feature importance plots for *Chenopodium album* using R-square value of Random forest on left and XGBoost on right

As you can see from PDP plots of *Chenopodium album* (see fig 4.4) the leaf size is decreasing at higher altitudes but meanwhile, it also shows some variations in leaf size with respect to latitude in the graph that happens due to the strong influence of other feature jointly with latitude and altitude on leaf size. Also, depending on the region at high and medium altitudes plants grow their leaves to maximum size in mid or late summer months.

Leaves are bigger in size before 1900 or around 1915 to 1925 then the leaf sizes are kept decreasing until the early 2000s and the leaf size is increasing with increasing temperatures from early 2000s. The Leaf sizes are decreasing with the increasing day of year and the reason for that is the leaf out for this species generally occurs in Months of March to May but it may differ because this species spread across all continents in Northern Hemisphere and as mentioned in earlier sections this species act as a winter annual in cold regions such as Europe and North America, summer annual in hot and sub temperate regions such as in Asia so the leaf out months may vary also in hot regions very water is limited the plants adapt small leaves because small leaves have better water usage efficiency.

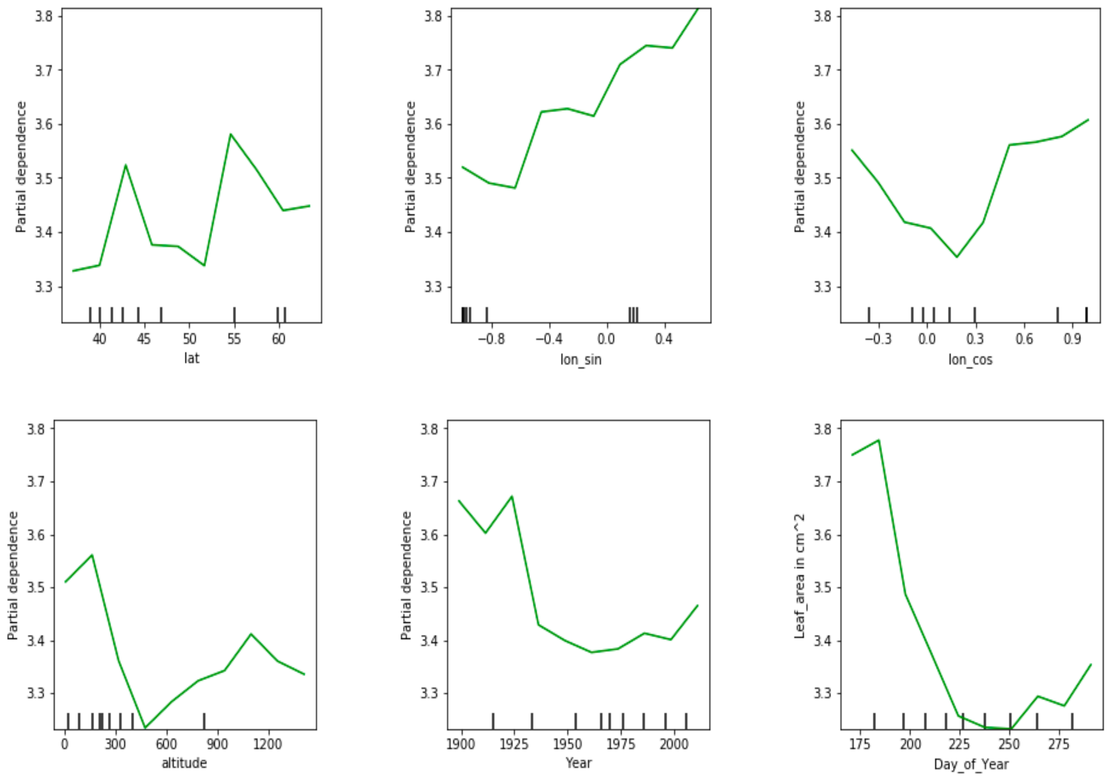


FIGURE 4.4: PDP plots for *Chenopodium album* using Random forest

TABLE 4.3: R-square values for *Impatiens capensis* species (herb).

***Impatiens capensis* R-square values**	
Datasets	**Random Forest Regressor**
Common_data	0.11
Space	-0.01
Climate_change (Year)	-0.05
Seasonal (Day of Year)	0.02
Climate_change & Seasonal	0.09
Space&Climate_change	0.015
Space& Seasonal	0.10

Feature importance plots of *impatiens capensis* (see fig 4.5) shows that the variations in leaf sizes are mostly due to seasonal especially day of year and this species shows more seasonal pattern.

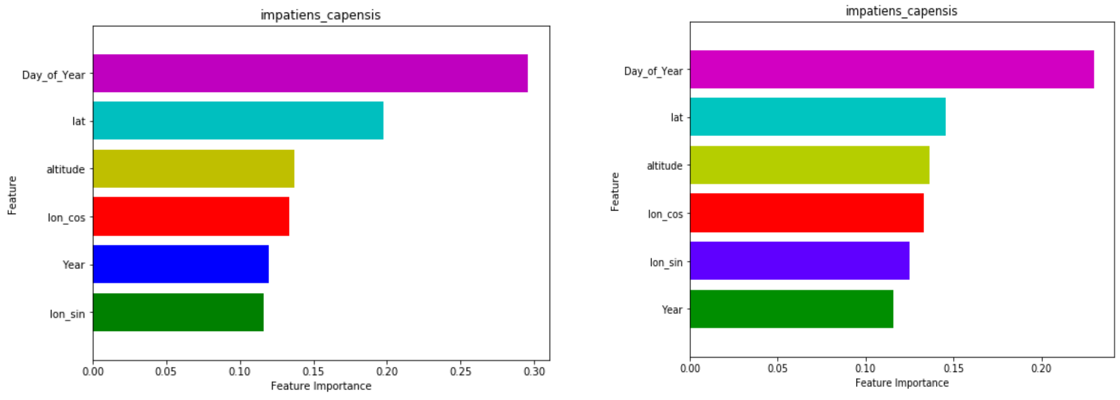


FIGURE 4.5: Feature importance plots for *Impatiens capensis* using R-square value of Random forestIon left and XGBoost on right

As you can see from PDP plots of *Impatiens capensis* (see fig 4.6) the leaf size is decreasing at a higher latitude and the reason for that is in cool climates where there is plenty of water, leaf sizes get smaller with distance from the equator because at cooler regions during night time the leaves lose their heat to the night sky and to compensate for energy loss it draws the heat from surroundings in order to avoid frost damage which larger leaf sizes don't have due to thicker boundary layers. The variations in leaf size are also decreasing with increasing temperatures and the leaf size is decreasing with the increasing day of year. These species are native to North America Continent where the leaf out occurs in the months of March and reaches to its maximum size in late May or by mid-June but it may vary depending on the climate conditions during that time of the year and these samples are collected at different stages of leaf growth but most of them in the summer months (i.e in between June - August).

Also, in the summer due to limited water the plants adapt smaller leaves and smaller leaves have better water usage efficiency as well as better ability of losing and gaining heat with surroundings because in summer months due to excessive heat and droughts the leaf temperatures get high and in order to survive it needs to maintain optimum leaf temperature by losing it heat to surroundings this happen through transpiration process that way it can avoid leaf brunt's or any damage which larger leaves don't have because of it's thick boundary layer.

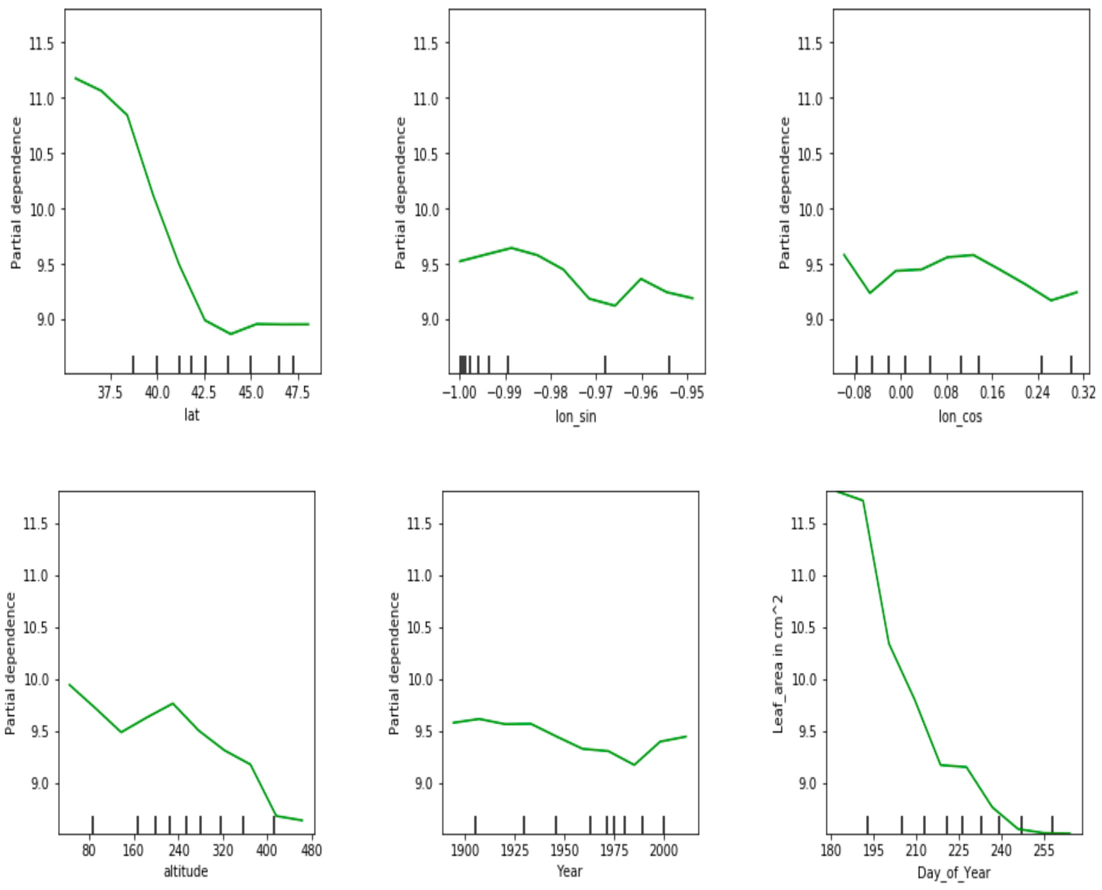


FIGURE 4.6: PDP plots for *Impatiens capensis* using Random forest

TABLE 4.4: R-square values for *Solanum dulcamara* species (shrub).

***Solanum dulcamara* R-square values**	
Datasets	**Random Forest Regressor**
Common_data	0.04
Space	0.006
Climate_change (Year)	-0.07
Seasonal (Day of Year)	-0.07
Climate_change & Seasonal	-0.05
Space&Climate_change	0.02
Space& Seasonal	0.04

Feature importance plots of *Solanum dulcamara* (see fig 4.7) shows that the variations in leaf sizes are mostly due to spatial features especially longitude and latitude.

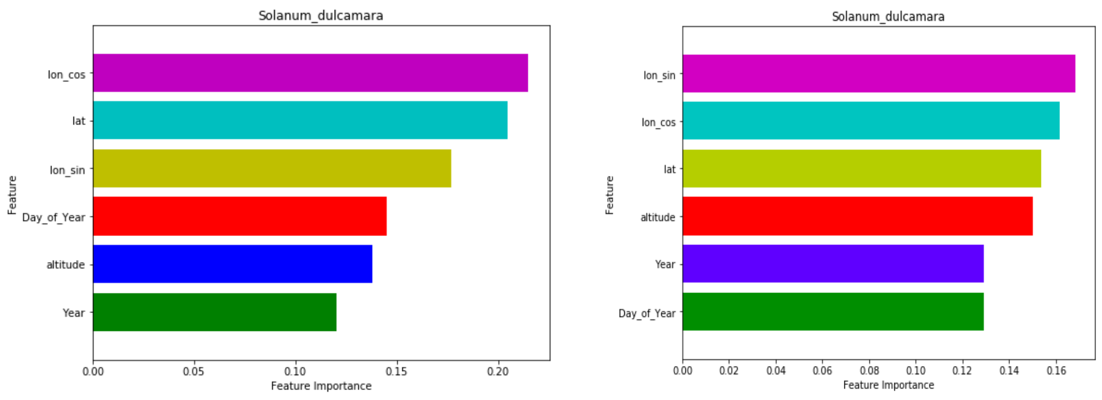


FIGURE 4.7: Feature importance plots for *Solanum dulcamara* using R-square value of Random forest on left and XGBoost on right

From PDP plots of *Solanum dulcamara* (fig 4.8) the leaf size is decreasing at higher latitude, but increasing in case of altitudes first leaf sizes are decrease and then increase with altitude this might happen due to the strong interactions of plant and its environmental factors such as temperature,amount of water availability, amount of sunlight absorbed, soil nutrition, etc., based on that some species have ability of growing well at low and medium elevations and most of the plants at low attitude grow their leaves to their maximum size in the spring months also at the same time during the growing season the nitrogen and phosphorous concentration is decreases with altitude especially above 500 meters and as a result leaf sizes get small.

Nitrogen and phosphorous are very important nutrients for plant growth. The nitrogen helps the plants to grow their leaves and also responsible for the chlorophyll and phosphorous help in storing energy, root development, fruits etc., and plants absorb these two elements both from soil and air.

The variations in leaf size yearly wise also decreasing and the leaf size is increasing with the increasing day of year and as mentioned earlier the leaf out process may vary based on the during that time of the year as well as with the region.

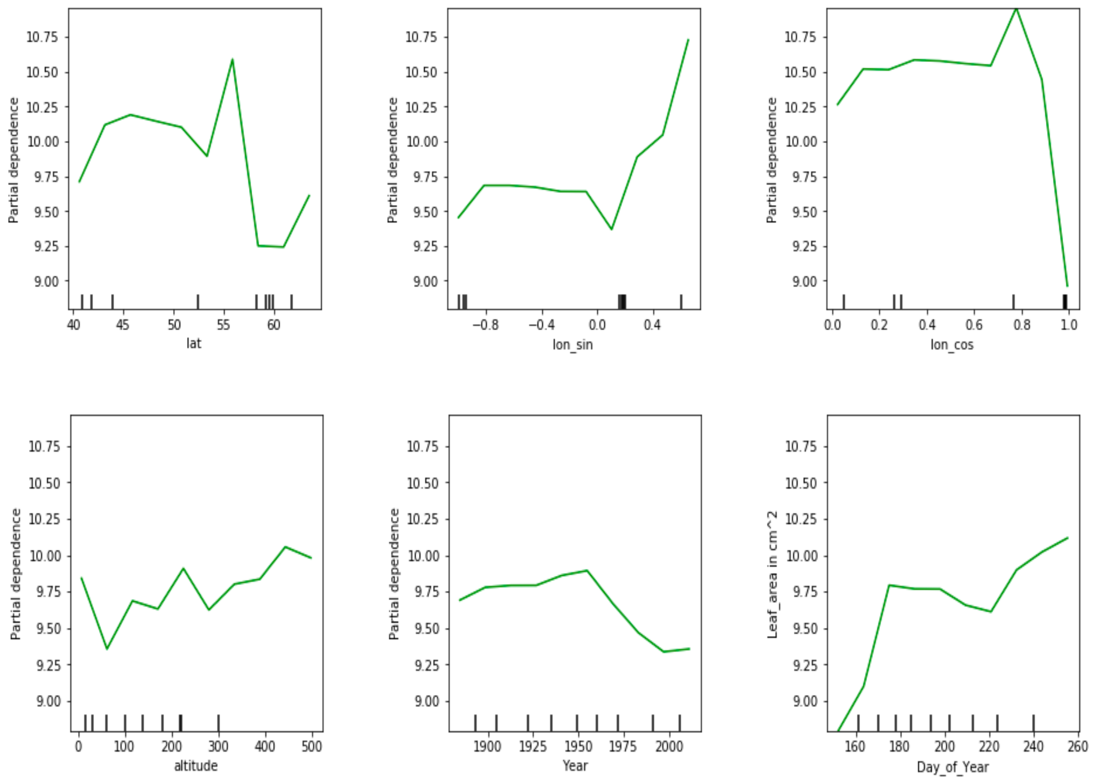


FIGURE 4.8: PDP plots for *Solanum dulcamara* using Random forest

4.2 Results on XGBoost models

In this section only the r-square values of different datasets combinations are provided but not the feature importance, partial dependency and variation partitioning plots and the reason for that are due to the r-square of both the models with different combinations of datasets are very much similar and as mentioned earlier the XGBoost is used to compare the results of random forest regressor and in 3 out of 4 cases, the random forest regressor results are better compared to both XGBoost model with and without out monotonic constraint.

TABLE 4.5: R-square values of *Salix glauca* species (shrub and tree).

***Salix glauca* R-square values**		
Datasets	**XGBoost Regressor**	**XGBoost Regressor with monotonic constraint**
Common_data	0.19	0.20
Space	0.17	0.18
Climate_change (Year)	-0.02	-0.01
Seasonal (Day of Year)	-0.04	-0.04
Climate_change & Seasonal	-0.03	0.005
Space&Climate_change	0.18	0.19
Space& Seasonal	0.19	0.20

TABLE 4.6: R-square values of *Chenopodium album* species (it can be herb, shrub, or small tree).

***Chenopodium album* R-square values**		
Datasets	**XGBoost Regressor**	**XGBoost Regressor with monotonic constraint**
Common_data	0.013	0.010
Space	-0.012	0.007
Climate_change (Year)	-0.017	-0.004
Seasonal (Day of Year)	-0.013	-0.002
Climate_change & Seasonal	-0.018	-0.004
Space&Climate_change	-0.006	0.009
Space& Seasonal	0.007	0.008

TABLE 4.7: R-square values of *Impatiens capensis* species (herb).

***Impatiens capensis* R-square values**		
Datasets	**XGBoost Regressor**	**XGBoost Regressor with monotonic constraint**
Common_data	0.09	0.11
Space	-0.03	-0.01
Climate_change (Year)	-0.04	-0.01
Seasonal (Day of Year)	0.04	0.08
Climate_change & Seasonal	0.04	0.05
Space&Climate_change	-0.011	0.009
Space& Seasonal	0.07	0.09

TABLE 4.8: R-square values of *Solanum dulcamara* species (shrub).

***Solanum dulcamara* R-square values**		
Datasets	**XGBoost Regressor**	**XGBoost Regressor with monotonic constraint**
Common_data	0.012	0.04
Space	-0.04	0.03
Climate_change (Year)	-0.06	-0.01
Seasonal (Day of Year)	-0.03	-0.01
Climate_change & Seasonal	-0.04	-0.01
Space&Climate_change	-0.03	0.03
Space& Seasonal	-0.007	0.03

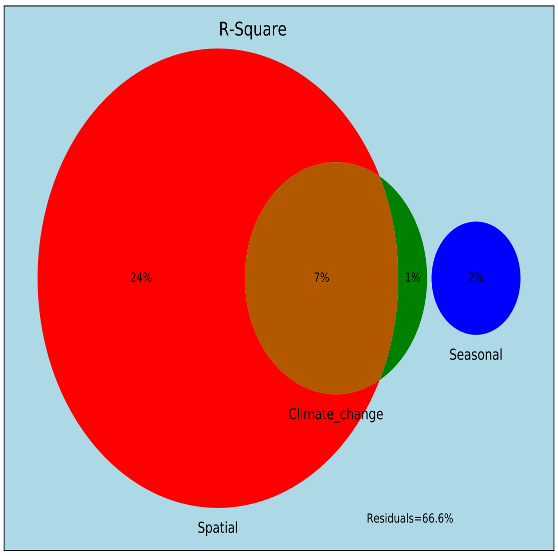
4.3 Results on Variation Partitioning

From the variation partitioning plot of *Salix glauca* species (see fig 4.9a) that the 24% variation in target variable (Leaf size) was occurred due to spatial effects and nearly 77% of that is explained by three spatial variables (such as latitude, longitude, and altitude), among that 23% of variations are explained jointly by three spatial variables and one climate_change variable (Year) and the 1% variations are explained by the climate_change variable (year), the seasonal effects (day of year) contribute 2% of the variations in Leaf sizes and the left out 67% (residuals) are explained by the other variables.

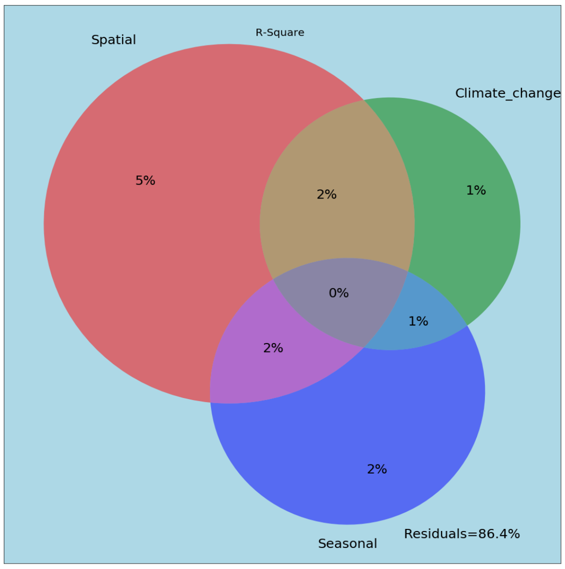
From the results of variation partitioning related to species of shrubs and trees at intraspecific level (see figures 4.9a, 4.9b, 4.9d) it clears shows that these species are more spatial structured (it means the variations in the leaf size are more from place to place and also the adaptions of plant functional traits due to changing environment are different for each species.

Similarly, the variation partitioning of *Impatiens capensis* species which is a herb (fig 4.9c) in that 9% of the variations are explained by the seasonal effects (day of year) among that 13% & 27% are jointly explained by seasonal_temporal variables & spatial_ seasonal variables each respectively and the remaining 2% & 1% of the variations in the target variable (leaf size) are explained by three spatial and one climate_change variable (year) each and among that nearly 75% are explained jointly by spatial and climate_change variables.

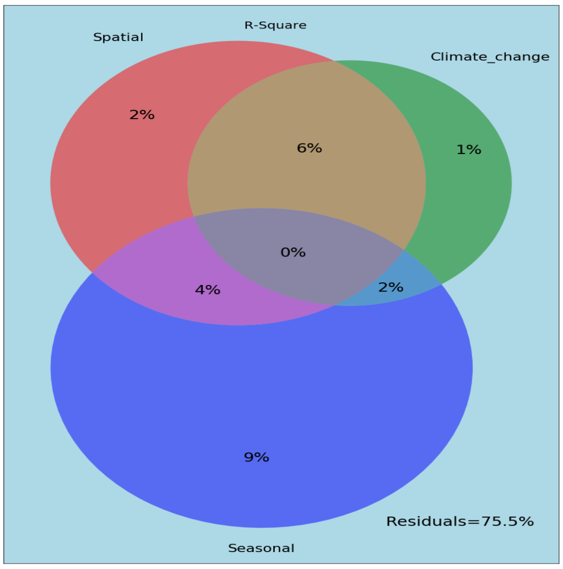
The negative or zero values indicate that all three effects(spatial, temporal, and seasonal) explain variations in the target variable (leaf size) much better as an individual effects than together. The remaining left out (Residuals) 75% of the variations in the target variable is explained by other environmental variables such as temperatures, soil, precipitation, etc., and in this case, it is more seasonal structured it means the variations are more seasonal than spatial and it has quite opposite pattern compared to shrub and tree species.



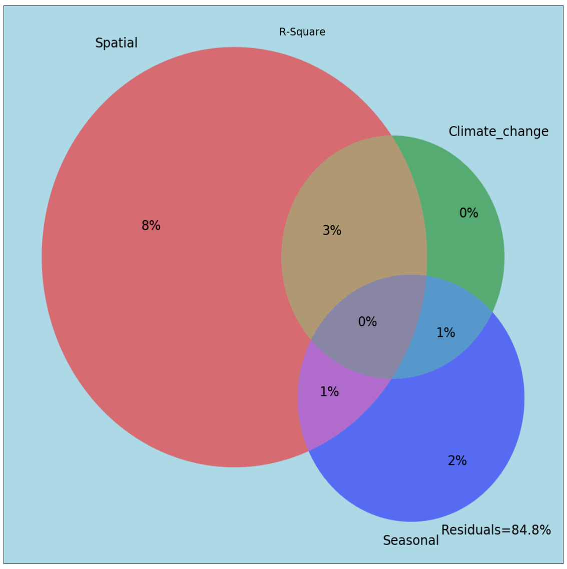
(A) *Salix glauca*



(B) *Chenopodium album*



(C) *Impatiens capensis*



(D) *Solanum dulcamara*

FIGURE 4.9: Variation Partitioning for all species using Random forest regressor

4.4 Results on Mixed effect Random forest models

The r-square values only with features temporal (year) and seasonal (day of year or extraterrestrial radiation) are calculated using the MERF model and the r-square values are a little bit higher compared to random forest model.

TABLE 4.9: R-square values of all species using MERF models.

Mixed effect Random Forest model	
Species	**R-square value**
Salix glauca	0.15
Chenopodium album	0.003
Impatiens capensis	0.10
Solanum dulcamara	-0.05

From figure 4.10 it clearly shows that the leaf sizes are increased over time irrespective of increasing temperatures and the leaf sizes are decreasing with increasing radiation figure 4.10 due to the increasing radiation the leaf growth stops and it also depends for how long it exposed to radiation the higher the expose the higher the damage as well as at higher radiation the leaf temperatures get high and leaves with bigger size have thick boundary layer so, it makes very hard for leaves to lose heat and as a result leaves get brunt thus why, plants tends to adapt smaller leaves with increasing radiation.

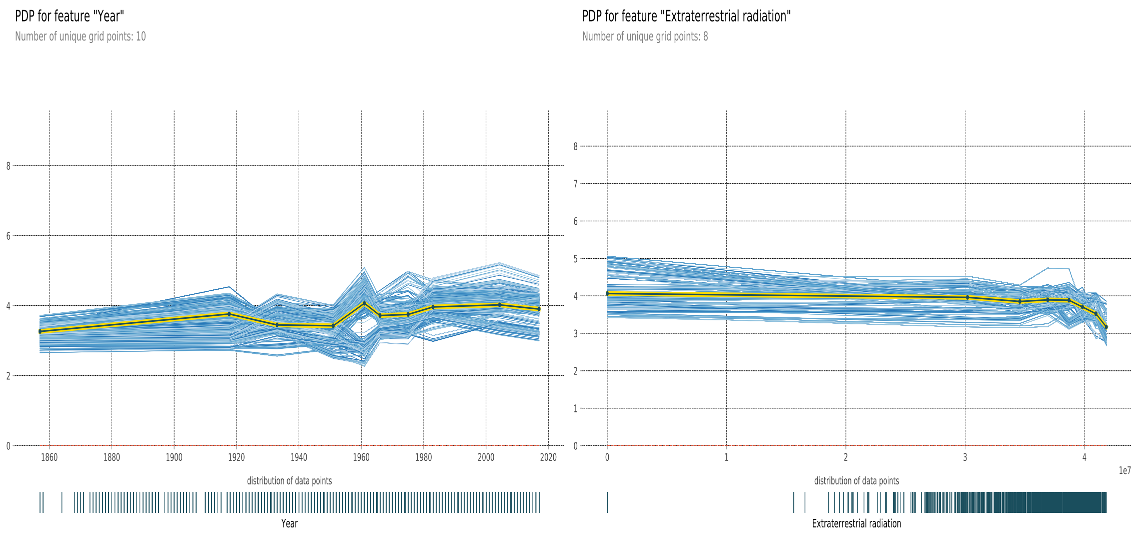


FIGURE 4.10: Partial dependency plots of *Salix glauca* species

From the partial dependency plots of *Chenopodium album* (see figure 4.11) the leaf sizes are larger before the 1920s and after 2005s in between leaves become a little bit shorter and remained constant and plants tend to adopt shorter leaves with increasing radiation because larger leaves have thicker boundary layer and it makes difficult to lose and gain heat and due to the higher radiations the leaf temperature increases and it may lead to leaf damage or leaf brunt. But, in this case the leaf sizes are rather increasing with increasing radiation and this happens depending on the type of the solar radiation plant absorb and plants mostly absorb the red and blue lights from the solar radiation which are good for the plant growth.

Also, It is important to mention that when plant expose to high radiations it means high sunlight good for photosynthesis but, exposing to high ultraviolet radiation stops the plant growth it also vary with species and very powerful radiation also contaminate the soil. The plants even get some additional benefits from the high radiation such as protecting from dangerous diseases and killing the microorganisms.

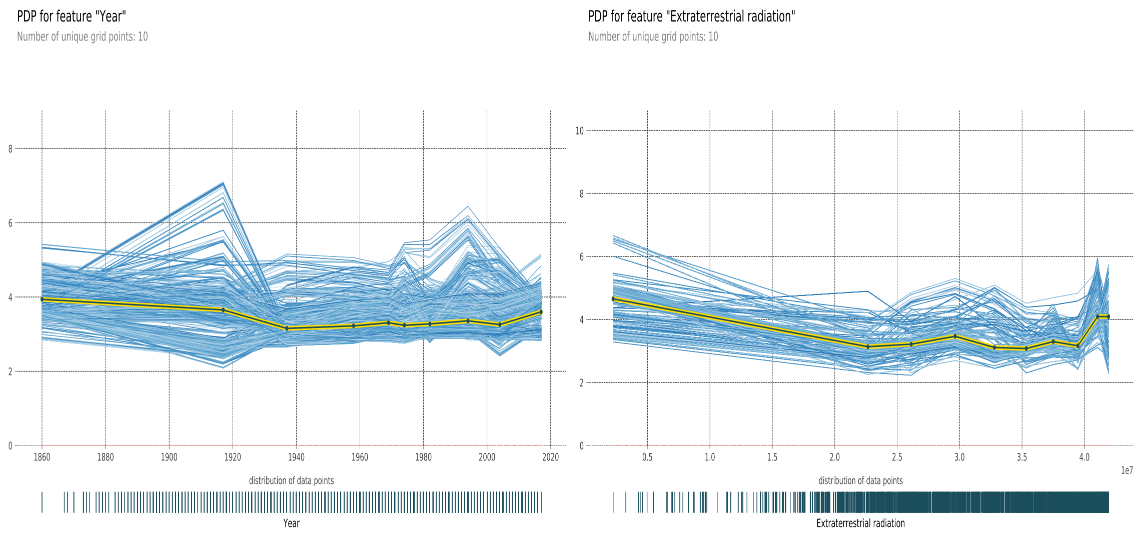


FIGURE 4.11: Partial dependency plots of *Chenopodium album* species

As you can see from figure 4.12 the species *Impatiens capensis* tends to have bigger leaf size around 1930 and after 2003 in between the leaf size are small and with increasing radiation the leaf size of this species is getting bigger and it seems like this species absorbing more visible rays especially red and blue light from solar radiation which are good for plant growth also, it seems like their is no problem with water availability which is required for the transpiration process to cool themselves due to excessive heat and good soil nutrition.

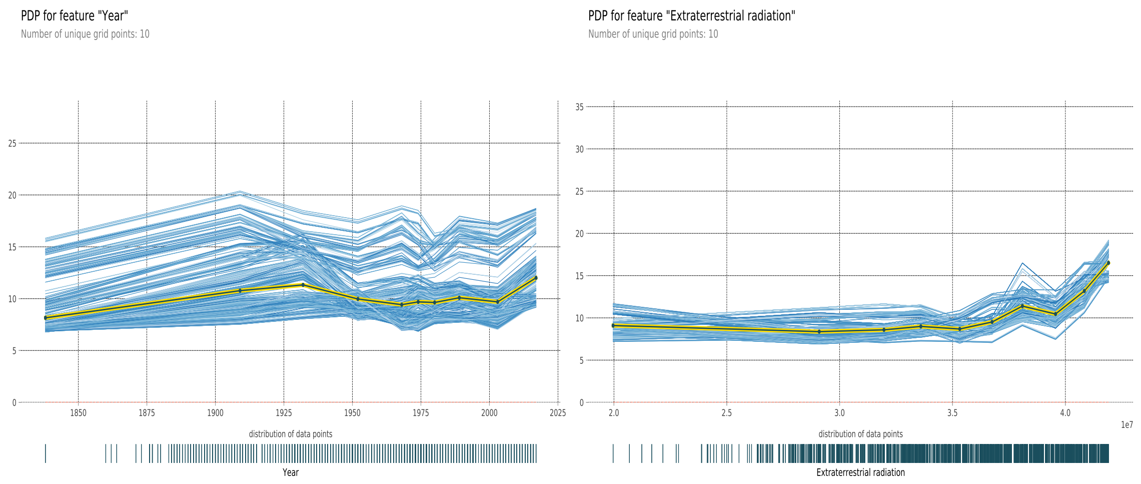


FIGURE 4.12: Partial dependency plots of *Impatiens capensis* species

From the partial dependency plots of *Solanum dulcamara* (fig 4.13), the leaf size is decreasing over the time until early 2000s then it started increasing also,the leaf sizes are increased with increasing extraterritorial radiation with some fluctuations this might happen due to the strong influence of the environmental factors such as temperature, water, light, soil, etc.,

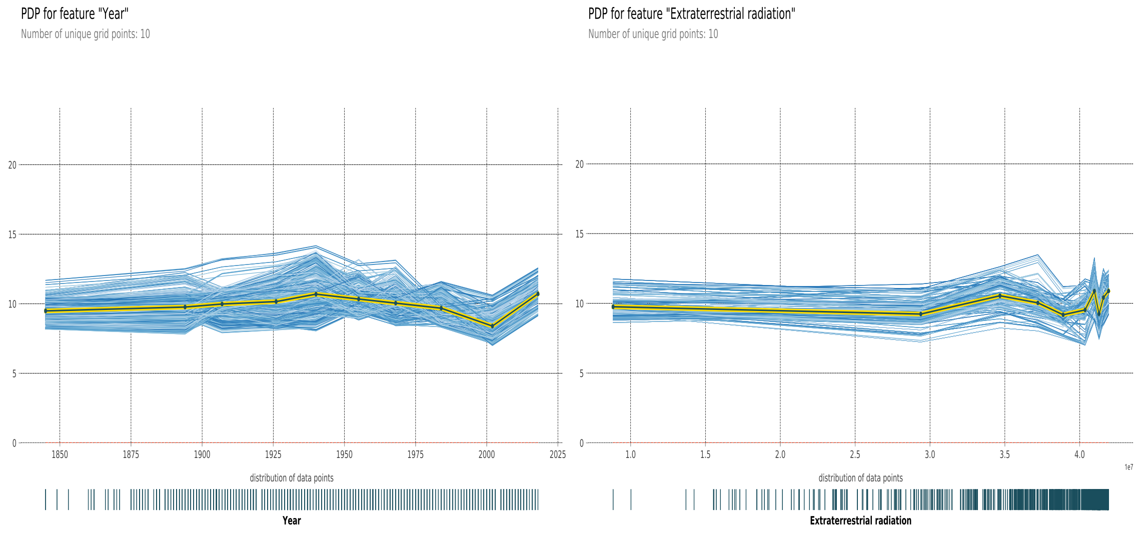


FIGURE 4.13: Partial dependency plots of *Solanum dulcamara* species

Chapter 5

Discussion

From the results, you can see the variation in the Leaf size at the intraspecific level is different for each species which includes species related to 1 tree, 2 shrubs, and 1 herb species. Also, it clearly shows that spatial, temporal and seasonal effects are significantly contributing in explaining and understanding the variation in leaf size due to environmental changes and the variation in leaf size at the intraspecific level are different for different species even though 3 out of 4 of them are spatial patterned and leaf size of different species varied similarly along latitude after accounting for effects of collection time decreased with increasing latitude, leading to smaller leaves towards the north, while latitudinal variation in leaf size differed across species.

Discussion of Results:

The most important drivers which are responsible for variations in leaf size at the intraspecific level are spatial variable such as latitude, longitude, and altitude compared to temporal and seasonal. This could be due to the fact that environmental factors are varied more strongly in space than in time.

With the help of variation partitioning technique along with the random forest regressor, the disentangle of spatial, temporal, and seasonal variations is done and it seems like a much better way to disentangle the temporal variations from the spatial variations compared to the Mixed effect random forest. Because in the case of the mixed effect random forest models only the variations in leaf size for time are done and remaining spatial features such as latitude, longitude and altitude are not considered here which are replaced with ecoregions to account for spatial features as explained in earlier sections. So, when the question of which factors contribute more comes then it will be more difficult to explain with mixed effect random forest model where only the analysis of the model done with certain features such as only with temporal and seasonal features like a fixed effect features by leaving out the spatial features.

Leaf size variation across space:
The variation in leaf size across all three space features latitude, longitude and altitude are negatively correlated and plants tend to have larger leaf size at lower latitude & altitudes and smaller leaf sizes at higher latitude & altitudes.

Meanwhile, the variations in leaf sizes with climate change differed among the species such for example shrub species like *Chenopodium album* (figure 4.2), *Solanum dulcamara* (figure 4.6) poses nonlinear variation in leaf size itself clearly shows that at the intraspecific level the species at different locations are adapting different leaf sizes to different climate conditions in order to survive by increasing their photosynthesis capacity and it might also happens due to strong influence of other environmental factors such as precipitation, temperatures etc., and at the same, their leaf sizes are increasing with increasing temperatures compared to the species like *Impatiens capensis* (figure 4.4), and *Salix glauca* species (figure 4.8) which are rather decreasing with increasing temperatures and this kind of leaf size pattern among different species with climate change is very similar to the previous studies such as Wright et al [50] on global climatic drivers of leaf size at the interspecific level, and Yaoqi Li et al [31] on spatiotemporal variations in leaf size response to climate change at intraspecific level.

Most of the previous studies have been focused on the variations in leaf size due to climate change at interspecific level and ignored at the intraspecific level and our results show that there are some significant contribution exists when coming to explaining about the variations in leaf size with respect to climate change across both space and time, and ignoring this might have consequences such as not so very accurate analysis results.

Leaf size change due to climate changes:
From the results of both random forest and mixed effect random forest models, firstly leaf sizes of all species are slightly decreased due to increasing temperature in between the years 1900s to around 1990s and then leaf sizes of all species are increasing with increasing temperatures around early 2000s and it clearly shows that the different species reacted differently to increasing temperatures by adapting leaf sizes according to the changing environments.

Also leaf size of all species varied non linearly with time and the reasons for that might be climate change due to sudden changes in atmospheric CO2 concentration, temperature, precipitation, etc., over time but in general which hasn't analyzed or studied in this thesis work due to limitations of data and you can see it from the PDP year plots of species such as *Chenopodium album* (fig 4.4 & 4.11), *Impatiens capensis* (fig 4.6 & 4.12) & *Solanum dulcamara* (fig 4.8 & 4.13) and *Salix glauca* species see (fig 4.2 & 4.10).

Besides that, the other factors such as mean annual CO2 concentration doesn't seem to have an impact over the variations in leaf size which was analyzed using a mixed effect random forest model. Also, the variations in leaf sizes are slightly better explained with mixed effect random forest compared to the random forest model because as explained in earlier section 2.3.5 the mixed effect random forest model is used only to analyze the variations in leaf size with regarding to temporal features by separating and making them independent of spatial features.

Moreover, the results of this study are reasonably similar to the previous studies such as by Stropp et al [43] on Drier climate shifts leaf morphology in Amazonian trees and Morison et al [33] on interactions between CO2 concentration and temperature on plant growth.

Leaf size change with season:
The leaf size have differed for different species and almost all these species which used in this project are from the Northern hemisphere that means the leaf out starts in spring months that is in between March and grows to its full size by the end on May but this leaf out process may differ and some plants reaches there maximum leaf size in mid or late summer months depending on the impact of the climate change due to temperature, radiation, etc., during that time of year.

Species such as *Salix glauca* (see fig 4.2) & *Solanum dulcamara* (see fig 4.8) tend to heave increasing leaf sizes the reason for that might be the scientists and botanists have been collecting them at different stages of leaf growth process way before they reaching there maximum sizes and species like *Chenopodium album* (see fig 4.4) & *Impatiens capensis* (see fig 4.6) are rather decreasing with increasing leaf areas and this might happen due to drought conditions in summer months due to less availability of water the plants may tend to adopt small leaf sizes because small leaves to keep themselves cool they evaporate the water from their surface and at the same time they have better water usage efficiency compared to larger leaf size [45] because larger leaves heat up more compared to small leaves at less water sites.

As explained in section 3.9 the day of year in case of the MERF model is converted into extraterrestrial radiation as you can see from the figure 4.10 for species *Salix glauca* the leaf size is decreasing with increasing radiation and it happens due to the higher radiation the leaf temperatures rises and in order to avoid the damage due to heat the species have adopted small leaf size over the time and other species such as *Chenopodium album* (fig 4.11), *Impatiens capensis* (fig 4.12) and *Solanum dulcamara* (fig 4.13) their leaf sizes are firstly decreased with increase radiation and then they started to adopt the bigger leaves with increasing radiation.

It important to mention that the variation in leaf sizes with respect to season according to the results of this study is more important in the herb species (*Impatiens capensis*) compared to other three species which are more spatially varied. In perspective of time the variations in leaf size are seems to be mostly due to seasonal than temporal.

Spatial autocorrelation:
The moran's I values are very low for all species (i.e at 5 km distance the Moran's I value is 2% and at 111 km distance it is less than 1%) so it tells that there is no spatial autocorrelation exists in the datasets for all species in case if there any correlation exists it is also removed by using block selection cross-validation.

Goodness of fit:
As mentioned in section 3.5.1 the r-square can be negative and that means it fits worse than the horizontal line (mean) and in this particular case, it happened because the chose model does not follow the trend of the data due to limited data. The models can explain the variations in leaf size around 3 to 20% and it varied depending on the quality of the species data and comparing the explaining of variability in leaf sizes due to climate change with respect to space and time both the random forest and XGBoost regressor models are very similar and when compared to the MERF model in case of explaining variations in leaf size only due to temporal effects the MERF models are little bit better.

Discussion of Methods

Quality of data:
Firstly the very limited availability of species data in terms of temperature over the years. Also, very large sample bias exists in the species data such as botanists or ecologists often collect the data before leaves reaches to its maximum size as well as collecting species near roadsides and populated areas are not very representative traits.

A very large number of herbar specimen images are not good to extract the trait data from it and while working with TraitEx it happens to have some of the problems in such as leaves overlap, leaf damage, no scale, sapling, and juvenile leaves, etc.

Statistical methods:
All the extracted traits values from TraitEx are groupby using mean, maximum, minimum, and median statistic measures calculated and compared the r-square values for all static measures but the median statistics measure is chosen to represent the every small and large leaf values exists in the dataset.

Chapter 6

Conclusion

The main objective of this thesis is to analysis the variations in leaf size with respect to changing environmental conditions by using space for time hypothesis is successfully implemented and coming to analyses with this approach on intraspecifc level the model can explain and predictive the outcomes around 5-25% depending on the type of species (for example tree, shrub, herb) and in most of the cases the variations in the plant functional traits (Leaf sizes) are mostly explained by three spatial features (lat, lon and altitude) and in case of herbs in this project only one species related to herb is analyzed which is more seasonal patterned than spatial based on this it is very tough to conclude because a herb of different species may show different variation pattern, but all in all the most of the species at an intraspecific level showing the variations in traits are spatial rather than seasonal and temporal.

The climate change has its impact on both spatially and temporally, and the impact of spatial features are more compared to the temporal and seasonal based on this project.

Future Work:

In future by taking those unexplained environmental variables such as soil data, temperature, perception, annual CO2 concentration etc., as a spatial, temporal and seasonal features along with latitude, longitude, altitude, year and day of year as well as by using the space for time hypothesis, herbar specimen images and variation partitioning it can explain variations in plant functional traits (leaf area) more precisely and the model predictions are more realistic.

Appendix A

Appendix A

A.1 Downloading herbar specimen images

```
# Modules or libraries required for downloading the Images using URL's
import pandas as pd
import requests
from PIL import Image
import threading
import time
from threading import Thread
from _thread import start_new_thread, allocate_lock
import shutil
from PIL import ImageFile
import warnings
from collections import Counter
```

```
#The path where the original pkl files from Gbif and Idigbio stored
path=r'/home/pbaddam/Downloads/pkl'
#This is the location where all those downloaded images are stored
finalpath= '/home/pbaddam/Downloads/download_herber_specimen_images'
#files in this path (In my case I have files of different species)
for file in os.listdir(path):
    final_df='/home/pbaddam/Downloads/pkl/{}'.format(file)
    file=pd.read_pickle(final_df)
#It keep the first Url's with same address are drop the remaning
    file.drop_duplicates(subset = ['accessuri'],keep = 'first', inplace =
   True)
    file['institutioncode'] = file['institutioncode'].str.replace('\W', '')
    file['catalognumber'] = file['catalognumber'].str.replace('\W', '')
    file['Sample_id'] = file['institutioncode'].astype(str) + '_' +
               file['catalognumber'].astype(str)
```

```python
    file['Image_id'] = file.Sample_id + '_' + (file.groupby('Sample_id').cumcount()+1).astype(str)
    file.reset_index(level = 0, inplace = True)
    file.drop(['index'],axis = 1,inplace = True)
# The below code is used to check if there is any null or NaN values in column 'accessuri'
    sum(pd.isnull(file['accessuri']))
# In case if null or NaN values exist then by using the following code those rows can be dropped from that dataframe
    file = file.dropna(axis=0, subset=['accessuri'])

```

LISTING A.1: Python example

A.1.1 Saving images into specific folders

```python
    for index, row in file.iterrows():
        Spieces_name= row.scientificname
#The return value is the concatenation of path and speciesname with exactly one directory separator ('/').
        parent_folder = os.path.join(finalpath, Spieces_name)
#In case if the above mentioned parent_folder does not exist then it try to create a parent_folder as specied else it will skip this part.
        if not os.path.exists(parent_folder):
            try:
                os.mkdir(parent_folder)
            except:
                pass
        file= row.Image_id
        url = row.accessuri
        print(url)
        result = requests.get(url, stream=True)
# It suspends execution for the given number of seconds for each iteration.
        time.sleep(4)
#For each 200 iterations it creates a folder as specified and store the images in it.
        if result.status_code == 200:
            child_folder = os.path.join(parent_folder, file)
            if not os.path.exists(child_folder):
                try:
                    os.mkdir(child_folder)
                except:
                    pass
# It creates the folder and save image in that folder with image name as specified and at the end to that name it will add '.jpg' extension
            full_path = os.path.join(child_folder, '{}.jpg'.format(file))
```

```
# It open the image file from the specified path and  writes the content to
     image
            with open(full_path, 'wb') as f:
                result.raw.decode_content = True
                shutil.copyfileobj(result.raw, f)
            try:
                image =  Image.open(full_path)
                image = image.resize((1900, 2400))
                ImageFile.LOAD_TRUNCATED_IMAGES = True
                image.save(full_path)
            except:
                pass
    print('done')
```

LISTING A.2: Saving images into specific folders based on Image_id

A.2 Merging all CSV files generated by TraitEx per specimen which contains Leaf values into one main CSV file

```
#Joining all the csv files (of each Image) into one main csvfile
path = r'/home/pbaddam/Downloads/csv'
#In the first step by using this piece of code 'for f in os.listdir(path)'
    it will bring all the folders (by the way folders of different species)
     in that particular path and then an empty dataframe is created in
    order to store all the appended values from that particular species
    folder into this empty dataframe.
for f in os.listdir(path):
    df = pd.DataFrame()
#Then by using below function it will bring the those specific folders (
    within each species folders) it brings the subfolder.
    for file in os.listdir(os.path.join(path,f)):
#Then With this condition of 'expected_csv_path' it specifically looks for
    the csv file in that sub folders which contains all trait values for
    that image.
        expected_csv_path = os.path.join(path, f,file, file + '.csv')
        csv_exists = os.path.isfile(expected_csv_path)
#Then using this condition 'csv_exists = os.path.isfile(expected_csv_path)'
     if that file exists it read that file and append all those files in
    that empty dataframe and that dataframe is named after that particular
    species name in the similary way it do for all species in case if there
     exists any in that path.
        if csv_exists:
#In my case I used multiple delimiter or seperators such as ',' or ';' etc
    in that case additionally this two (sep='[;,]',engine='python') are
    used like in below code.
```

```
            x = pd.read_csv(expected_csv_path,header=0,sep='[;,]',engine='python',index_col=False,names=['ImageID','LeafID','type','length','width','area','perimeter','circularity','parent'])
            df = df.append(x)
    df.reset_index(inplace=True,drop=True)
    df.to_csv('/home/pbaddam/Downloads/csv_merged/{}.csv'.format(f), index=False)

```

LISTING A.3: Main CSV File

A.3 Filtering the main CSV file and saving as pkl file

```
#Manipultion(filtering, storing) of csv_file with Leaf measurements into human readable form
#path in which all main csv files for different species exists
path= r'/home/pbaddam/Downloads/csv_merged'
for files in os.listdir(path):
    try:
        final_df='/home/pbaddam/Downloads/csv_merged/{}'.format(files)
        dataset = pd.read_csv(final_df, dtype={'a': str}, engine='python', sep='[,;]')
        dataset_1= dataset.drop(columns=['LeafID', 'parent'])
        dataset_2= pd.DataFrame({'petiole_length': dataset['length'].iloc[1::2].values})
        dataset_2.reset_index(drop=True, inplace=True)
        dataset_3= dataset_1[dataset_1.type != 'petiole']
        dataset_3.reset_index(drop=True, inplace=True)
        new_dataset= dataset_3.join(dataset_2)
        finaldata= new_dataset.rename(columns={'length': 'Leaf_length', 'width': 'Leaf_width', 'area': 'Leaf_area', 'perimeter': 'Leaf_perimeter'})
        finaldata_1=new_dataset['ImageID'].str.split('/', n=6, expand=True)
        finaldata_1['Image_id'] = finaldata_1[5]
        finaldata_2= finaldata_1.drop(columns=[0,1,2,3,4,5,6])
        finaldata_3=finaldata.drop(columns=['ImageID', 'type'])
        finaldata_4= finaldata_2.join(finaldata_3)
        finaldata_5= finaldata_4.groupby(['Image_id']).mean()
        finaldata_6 = finaldata_5.reset_index()
        finaldataset_file_path= finaldata_6.to_pickle('/home/pbaddam/Downloads/pklleaf/{}.pkl'.format(files.rsplit( ".", 1 )[ 0 ]))
    except:
        pass

```

LISTING A.4: Filtering main CSV file and saving as pkl file

A.4 Merging metadata file from Gbif and Idigbio with the pkl files which contain trait values

```
#Merging the orginal pkl file from idigbio and Gbif with the pkl file
   which contains leaf measurements based on the 'Image\_id. Also, the '
   Image\_id' and 'sample\_id' for original pkl file are created.
for files in os.listdir('/home/pbaddam/Downloads/mainpkl'):
    df_3= '/home/pbaddam/Downloads/mainpkl/{}'.format(files)
    df_4=pd.read_pickle(df_3)#original pkl file which contains information
   about species
    df_4.drop_duplicates(subset=['accessuri'],keep='first', inplace=True)
    df_4['institutioncode']=df_4['institutioncode'].str.replace('\W', '')
    df_4['catalognumber']=df_4['catalognumber'].str.replace('\W', '')
    df_4['Sample_id']= df_4['institutioncode'].astype(str) + '_' +df_4['
   catalognumber'].astype(str)
    df_4['Image_id']= df_4.Sample_id + '_' +(df_4.groupby('Sample_id').
   cumcount()+1).astype(str)
    df_4.reset_index(level=0, inplace=True)
    df_4.drop(['index'], axis=1, inplace=True)
    df_5= df_4['datecollected'].str.split('T', n=1, expand=True)
    df_6= df_5.drop(columns=[1])
    df_6['datecollected']=df_6[0]
    df_7= df_6.drop(columns= [0])
    df_8 = df_4.drop(columns='datecollected')
    df_9=df_8.join(df_7)
#This are pkl files with leaf values
path= r'/home/pbaddam/Downloads/pklleaf'
for files in os.listdir(r'/home/pbaddam/Downloads/pklleaf'):
    #print(files)
    df_1='/home/pbaddam/Downloads/pklleaf/{}'.format(files)
    df_2=pd.read_pickle(df_1)#pkl file which contain leaf measurements
    #final=df_2.join(df_9, lsuffix='Image_id')
    final=df_2.merge(df_9)
    final_path= final.to_pickle(r'/home/pbaddam/Downloads/merged_pkl_files
   /{}'.format(files))

```

LISTING A.5: Merging both original pkl file and pkl file with leaf values

A.5 Creating Blocks for block selection cross validataion

A.5.1 Generating 360x720 Matrix with block size of 0.5 degree and assigning values to those blocks

```
def generate_block(block_size):
```

```
2      nrows = int(180/block_size)
3      ncols = int(360/block_size)
4      data = np.zeros((nrows,ncols))
5  #Creating 9x9 blocks and assigning values to those blocks
6  #The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number in this case(range(3)) so it returns the values (0,1,2)
7  #now with this conditions see lines 8-9 it creates a nested for loop and each value in rows is iterated against all values in columns and it return all those empty 9x9 blocks with values from 0 to 8
8      for i_start in range(3):
9          for j_start in range(3):
10             fill_number = i_start * 3 + j_start
11             data = fill_one_number(data, i_start, j_start, fill_number, nrows, ncols)
12     return(data)
13
```

LISTING A.6: Example 9x9 Blocks

A.5.2 Assigining values to 360x720 matrix

```
#Now in order to do the same thing for the entire 360 x 720 blocks. the below function  contains 'data' which is a empty dataframe with 360 x 720 blocks, i\_start=0, j\_start=0, nrows=360, ncols=720.
def fill_one_number(data, i_start, j_start, fill_number, nrows, ncols):
#But the only difference is increment value it means i\_start=0, nrows=360, but increment value is 3 that means it returns the i values like this (0,3,6,9,12,15,...)
    for i in range(i_start, nrows, 3):
#similarly j values j\_start=0, ncols=720, increment is 3 so it brings the sequence of numbers from 0 to 720 as (0,3,6,9,12,...)
        for j in range(j_start, ncols, 3):
#Now for example if i = 0 and j = 0 and by using the 'fill\_number' definition below that means at each nested loop as a starting point from i=0th row and j=0th column the blocks are assigned with 0 to 8 values (9x9 blocks for (0 row,0 col with value 0), (0 row,1 col with value 1), (0 row,2 col with value 2), (1 row,0 col with value 3), (1 row,1 col with value 4), (1 row,2 col with value 5), (2 row, 0 col with value 6), (2 row, 1 col with value 7), (2 row, 2 col with value 8)) in the similarly way in the next iteration if the condition is met like for example if i = 0 and j = 3 it start filling all those blocks (starting with 0th row and 3 rd col) with values from (0 to 8) until the next iteration and in the similar way it repeats the same procedure for the entire (360 X 720) blocks
            data[i,j] = fill_number
    return data
```

```

```

LISTING A.7: example 360x720 matrix

A.5.3 Creating Indices for each row and column

```python
#Here indicies for each row and each column are created and those row and
    column indexes are created based on the lat, lon, block and Block_size
    that way the data points with similar lat and lon get same index value
    and while doing the cross validation either this values taken into
    training set or test set but not one into train and other into test and
     that how the dependence between the data points are reduced.
def get_index(block, block_size, lng, lat):
    irow = int((90 + lat)/ block_size)
    icol = int((lng + 180)/ block_size)
    return block[irow,icol]
```

LISTING A.8: creating indicies

A.6 Hyperparameter Tuning using Randomized Search CV method

```python
#for hypertuning of parameters
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import KFold, cross_val_score
```

```python
for files in os.listdir('/home/pbaddam/Downloads/final_pkl_files'):
    df_16='/home/pbaddam/Downloads/final_pkl_files/{}'.format(files)
    dfout=pd.read_pickle(df_16)
    x= dfout[['lat', 'lon_sin', 'lon_cos', 'altitude', 'Year', 'Day_of_Year
    ']]
    y= dfout[['Leaf_area']]

     # Number of trees in random forest
    n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000,
   num = 10)]
    # Number of features to consider at every split
    max_features = ['auto', 'sqrt']
    # Maximum number of levels in tree
    max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
    max_depth.append(None)
    # Minimum number of samples required to split a node
    min_samples_split = [2, 5, 10, 15]
    # Minimum number of samples required at each leaf node
    min_samples_leaf = [1, 2, 4, 5, 6]
    # Method of selecting samples for training each tree
    bootstrap = [True, False]
    #oob_score = [True, False]
    # Create the random grid
    randomstate = [123,10, 42]
    random_grid = {'n_estimators': n_estimators,'max_features': max_features,'max_depth': max_depth,
                   'min_samples_split': min_samples_split,'min_samples_leaf': min_samples_leaf,
                   'bootstrap': bootstrap,'random_state' : randomstate}

    rf = RandomForestRegressor()
    # Random search of parameters, using 10 fold cross validation,
    # search across 100 different combinations, and use all available cores
    rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, cv = 10,random_state=42)
    # Fit the random search model
    rf_random.fit(x,y)
```

LISTING A.9: Hyperparameter Tuning

```
rf_random.best_params_

{'random_state': 42,
 'n_estimators': 400,
 'min_samples_split': 5,
 'min_samples_leaf': 5,
 'max_features': 'auto',
 'max_depth': 10,
 'bootstrap': True}
```

LISTING A.10: Best parameters

A.7 Data Pre-Processing

A.7.1 Implementing blocks and DEM (altitude)

```
#for files in os.listdir('OBG/Pramod/Merged_pkl_files'):
for files in os.listdir('/home/pbaddam/Downloads/merged_pkl_files'):
    final_df='/home/pbaddam/Downloads/merged_pkl_files/{}'.format(files)
    #final_df='OBG/Pramod/Merged_pkl_files/{}'.format(files)
    df = pd.read_pickle(final_df)
    df['datecollected']=pd.to_datetime(df['datecollected'])
    df['Year']=df['datecollected'].dt.year
    df['Day_of_Year']=df['datecollected'].dt.dayofyear
# the following code check if there is any null values in the particular
    features of dataset and in case if there is any it will return the
    results with that feature as true
    df.isnull().any(axis=0)
#If there is any null values it is filled with mean values with respect to
    that particular feature
    df1=df.fillna(df.mean())
#This is the file location where the altitude dataset is stored
    ds=xr.open_dataset('/Net/Groups/BGI/data/DataStructureMDI/DATA/grid/
    Global/0d0083_static/GTOPO30/not_defined/Data/GTOPO30.43200.21600.nc')
    edf=pd.DataFrame()#for DEM
    edf1=pd.DataFrame()#for blocks
    for index, row in df1.iterrows():
        lat=row.lat
        lng=row.lon
        #for DEM or altitude
        x_1=ds.sel(lat=lat, lon=long, method='nearest')
        p=x_1.to_array().values
        ext=(pd.DataFrame({'DEM':p[0]}, index=[0]))
        edf=edf.append(ext)
        # for blocks
        block_size = 0.5
#generating the blocks based on the specified block size
        block= generate_block(block_size)
#It get the indicies
        x_2=get_index(block, block_size, lng, lat)
        some=(pd.DataFrame({'Block':x_2}, index=[0]))
# appending all the indicies values under the column named as 'Block' to
    the existing 'edf' dataframe
        edf=edf.append(some)

    edf.reset_index(drop=True, inplace=True)
    final=df1.join(edf, how='left', lsuffix='left')
    final=df1.join(edf1, how='left', lsuffix='left')
    final.to_pickle('/home/pbaddam/Downloads/processed_pkl_files/{}'.format
    (files))
    #final.to_pickle('OBG/Pramod/processed_pkl_files_with_blocks_&_DEM/{}'.
    format(files))
```

LISTING A.11: Hyperparameter Tuning

A.7.2 Encoding Cyclic features and removing outliers

```
for file in os.listdir('/home/pbaddam/Downloads/processed_pkl_files'):
    final_df='/home/pbaddam/Downloads/processed_pkl_files/{}'.format(file)
    #final_df='OBG/Pramod/Merged_pkl_files/{}'.format(files)
    df_13 = pd.read_pickle(final_df)
    #df2=df1.rename(columns={'DEM': 'altitude', 'GTOPO30': 'altitude'})
    #df2= df1[['lat', 'lon', 'altitude', 'Year', 'Day_of_Year', 'Block','Image_id', 'Leaf_area']]
    df2= df_13[['lat', 'lon', 'Year', 'Day_of_Year', 'Block','Image_id', 'Leaf_area']]
    #df2.isnull().any(axis=0)
    #df3=df2.fillna(df2.mean())
    #cyclic feature here 'pi' value is 180 and 360 is the total range of longitude (0-360 degress)
    df2['lon_cos']=np.cos(df2.iloc[:,1]*(2.*np.pi/360))
    df2['lon_sin']=np.sin(df2.iloc[:,1]*(2.*np.pi/360))
    #col=['lat','lon', 'lon_sin', 'lon_cos', 'altitude', 'Year', 'Day_of_Year', 'Block',  'Leaf_area']
    col=['lat','lon', 'lon_sin', 'lon_cos', 'Year', 'Day_of_Year', 'Block', 'Leaf_area']
    df3=df2[col]
    #Removing outliers from data using Z-score technique here if the Z-score value is greater than or less than threshold(3 or -3) respectively it will identify or treate as a outliers
    z=np.abs(stats.zscore(df3))
    threshold=3
    #print(np.where(z>3))
    df_outliers=df3[(z<3).all(axis=1)]
    #df_outliers.hist(column='Year', figsize=(5,5))
    #plt.title('{}'.format(file.rsplit( ".", 1 )[ 0 ]))
    #plt.show()

    file_to_save= df_outliers.to_pickle('/home/pbaddam/Downloads/final_pkl_files/{}'.format(file))
```

LISTING A.12: Hyperparameter Tuning

A.8 Implementing Blocks along with Machine learning algorithm

A.8.1 Organizing datasets, model fitting and getting R2 score

```
for files in os.listdir('/home/pbaddam/Downloads/final_pkl_files'):
    df_16='/home/pbaddam/Downloads/final_pkl_files/{}'.format(files)
    dfout=pd.read_pickle(df_16)
```

```
#In the next step the spatial, temporal and seasonal features or variables
    such as (latitude, longitude, altitude), (Year), (Day of year) and
    target feature (Leaf area) are organized in such a way as follows: For
    example as mentioned below \textbf{'Common_ data'} it is a dataset
    contains all features such as latitude, longitude, altitude, Year, Day
    of year and target feature Leaf area in it as a features. In the
    similar way \textbf{'Space'} dataset contains only spatial features
    such as latitude, longitude, altitude and target feature Leaf\_area.
    Like that all the datasets are prepared in each and every possible
    combinations to find out the R-square values with respect to target
    feature Leaf area for the better understanding of the variations
    through the predicted R-square value from test data.

    Common_data = dfout.iloc[:,[0,2,3,4,5,6,7,8]]
    Space = dfout.iloc[:,[0,2,3,6,7,8]]
    Year_and_Day_of_Year = dfout.iloc[:, [4,5,6,7,8]]
    Space_and_Year = dfout.iloc[:,[0,2,3,4,6,7,8]]
    Space_and_Day_of_Year = dfout.iloc[:,[0,2,3,5,6,7,8]]
    Year = dfout.iloc[:,[4,6,7,8]]
    Day_of_Year = dfout.iloc[:,[5,6,7,8]]
    datasets = [Common_data, Space, Year_and_Day_of_Year, Space_and_Year,
    Space_and_Day_of_Year, Year, Day_of_Year]
    datasets_names = ['Common_data', 'Space', 'Year_and_Day_of_Year', '
    Space_and_Year', 'Space_and_Day_of_Year', 'Year', 'Day_of_Year']
    # Implementing Block selection cross validation and Machine Learning
    algorithm
    d = {}
    for (dataset, datasetname) in zip (datasets, datasets_names):
#Here while implementing cross validation along with algorithm. In the
    first step to the folds variable and empty lists is assigned
        folds = []
# then to i variable (i = np.arange(0,9,1)) it means it returns the array
    with values (0,1,2,3,4,5,6,7,8) at an increment of 1 value.
        i = np.arange(0,9,1)
#Then by using the below condition for values (k) in i variable a 'x'
    variable is created in such a way for example in the dataset under
    column 'Block' if values equals to K values (from  i variables) all
    those values are appended into one fold like that it repeated for all
    values from (0 to 8) in that way 9 folds are created and this folds are
     stored in those empty folds list which is created earlier.
        for k in i:
            x= dataset[dataset['Block']==int(k)]
            folds.append(x)
        model = RandomForestRegressor(random_state= 42,n_estimators=400,
    min_samples_split= 5,min_samples_leaf= 5,max_features= 'sqrt',
        max_depth= 10,bootstrap= True, oob_score= True)
#All the test sets from all 9 folds are stored
        ytest = []
```

```
# the predicted values from all 9 folds are stored
        ypredic = []
        for fold in range(len(folds)):
#The training and testing sets are implemented using this folds. Here out
    of 9 folds, 8 folds are used as train sets and remaining one fold used
    as test set and the 1 fold which used as a test set is deleted from the
     train sets like that it brings the r2 score for each and every
    possible fold and the final r2 score is the average of that.
            train = folds.copy()
            test = folds[fold]
            del train[fold]
            train = pd.concat(train, sort=False)
            x_train = train.copy()
            x_test = test.copy()
            y_train = x_train.pop('Leaf_area').values
            y_test = x_test.pop('Leaf_area').values
            ytest.extend(y_test)
            model.fit(x_train, y_train)
            y_predicted = model.predict(x_test)
            ypredic.extend(y_predicted)
        print('r2_score_of/{}'.format(files.rsplit(".", 1)[ 0 ]),
            datasetname, 'is',  r2_score(ytest,ypredic))
        d['{}'.format(files.rsplit( ".", 1 )[ 0 ]),datasetname]
            = r2_score(ytest,ypredic)
```

LISTING A.13: Organizing datasets and calculating R-square value for all datasets

A.9 Variation Partitioning

```
#for venn diagrams using matplotlib libraries
from matplotlib_venn import venn2, venn2_circles
from matplotlib_venn import venn3, venn3_circles
from matplotlib_venn import venn3_unweighted
from matplotlib_venn._math import *
from matplotlib_venn._common import *
from matplotlib_venn._region import VennCircleRegion, VennEmptyRegion
```

```
#A is the variable which contains R-square value related to three spatial
    variables.
    A = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Space')]
#B contain R-square value related to variables  climate\_change (year).
```

```
    B = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Year')]
#C contains R-square value related to seasonal variables (day of year ).
    C = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Day_of_Year')]
#AB contains R-square value related to spatial (lat,lon, alt) and climate (
    year) variables.
    AB = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Space_and_Year')]
#AC contain R-square value related to space (lat,lon,alt) and seasonal (day
     of year) variables.
    AC = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Space_and_Day_of_Year
    ')]
#BC contain R-square value related to climate (year) and seasonal (day of
    year) variables.
    BC = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Year_and_Day_of_Year'
    )]
#totalexpl contain  R-square values related to three spatial, climate and
    seasonal variables
    totalexpl = d[('{}'.format(files.rsplit( ".", 1 )[ 0 ]),'Common_data')]
```

LISTING A.14: Variation Partitioning

```
    a = totalexpl - BC # pure spatial (removing climate change and seasonal
     variables)
    b = totalexpl - AC # pure climate_change
    c = totalexpl - AB # pure seasonal effects
 # part shared and explained by three spatial variables and one
    climate_change variable
    d = totalexpl - a - b - C
# part shared and explained by one climate-change variable and one seasonal
     variable
    e = totalexpl - b - c - A
# part shared and explained by three spatial variables and one seasonal
    variable
    f = totalexpl - a - c - B
# part explained by all three spatial, climate_change and seasonal
    variables together
    g = totalexpl - a - b - c - d - e - f
# part explained by other environmental variables other then those spatial,
     climate_change and Seasonal.
    Residuals = 1- totalexpl
    #Residuals=1-a-b-c-d-e-f-g #(alternative)

    df=[a,b,c,d,e,f,g,Residuals]
# with this condition the values in 'df' list which are equals to or less
    than zero are treated as 0
    df = [0 if i <= 0  else i for i in df]
```

LISTING A.15: Variation Partitioning

```
    fig = plt.figure(figsize = (7,7))
    total = 1
    #V = venn3(subsets= (df[0],df[1],df[3],df[2],df[5],df[4],df[6]),
   set_labels=('Spatial', 'Temporal', 'Seasonal'), alpha=0.5, normalize_to
   =1.0, ax=None, subset_label_formatter=lambda x: f"{(x/total):1.0%}")

#In case if you want to draw three circles with same size irrespective of
   their values use 'venn3_unweighted' and here using this same function
   you can also set the circles with respect to their values by simply
   using 'Subset_areas' like as in below code
    V = venn3_unweighted(subsets= (df[0],df[1],df[3],df[2],df[5],df[4],df
   [6]), set_labels=('Spatial', 'Temporal', 'Seasonal'), subset_areas = (
   df[0],df[1],df[3],df[2],df[5],df[4],df[6]), alpha=0.5, normalize_to
   =1.0, ax=None, subset_label_formatter=lambda x: f"{(x/total):1.0%}")
    #c = venn3_circles(subsets =(a,b,d,c,e,f,g), linestyle ='dashed',
   linewidth=1, color='grey')
    plt.title('Variation Partitioning {}'.format(files.rsplit( ".", 1 )[ 0
   ]), fontsize=15)
    #setting position of 'R-sqaure'
    plt.text(0, 0.52, 'R-Square', horizontalalignment='center',fontsize=15,
    color='black')
    #setting poistion of 'Residuals'
    plt.text(0.65,-0.61,r'Residuals={:.1%}'.format(df[7]),
   horizontalalignment='right', verticalalignment='bottom')
    #for text in V.subset_labels:text.set_fontsize(8)
    plt.gca().set_facecolor('lightblue')
    plt.gca().set_axis_on()
    plt.show()
```

LISTING A.16: Variation Partitioning

A.10 Fitting model to the data with XGBoost Regressor

```
#modules for XGBoost
import xgboost as xgb
```

A.10.1 Hyper Parameter Tuning for XGBoost

```
for files in os.listdir('/home/pbaddam/Downloads/final_pkl_files'):
    df_16='/home/pbaddam/Downloads/final_pkl_files/{}'.format(files)
    dfout=pd.read_pickle(df_16)
    x= dfout[['lat', 'lon_sin', 'lon_cos', 'altitude', 'Year', 'Day_of_Year
   ']]
    y= dfout[['Leaf_area']]

    parameters = {
    # Maximum number of levels in tree
     "max_depth"         : [ 3, 4, 5, 6, 8, 10, 12, 15],
     "min_child_weight" : [ 1, 3, 5, 7],
     "gamma"             : [ 0.0, 0.1, 0.2 , 0.3, 0.4],
     "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7, 1],
    # Minimum number of samples required to split a node
     "min_sample_split" : [100,200,500,1000,1200],
     # Minimum number of samples required at each leaf node
     "min_samples_leaf" : [10, 15, 20, 25, 30, 35, 40, 50, 60, 70, 80],
     "min_weight_fraction_leaf":[0.1, 0.2, 0.3],
     "max_depth"         : [5,6,7,8,9],
    # Number of features to consider at every split
     "max_feature"       : ['sqrt', 'auto'],
     "learning_rate"     : [0, 0.05 ,0.1, 0.2, 0.5],
     "reg_alpha":[0, 0.001, 0.005, 0.01, 0.05],
     "subsample"         : [0.8],
     # Create the random grid
     "randomstate"       : [1, 10, 42, 123],
      # Number of trees in random forest
     "n_estimators"      : [100, 200, 500, 1000, 1500]}

    model = xgb.XGBRegressor()
    rf_random = RandomizedSearchCV(estimator = model, param_distributions =
     parameters, n_iter = 1000, cv = 10, verbose=1, random_state=42, n_jobs
     = 5)
    # Fit the random search model
    rf_random.fit(x,y)
    rf_random.best_params_
```

LISTING A.17: Parameter Tuning for XGBoost

A.10.2 XGBoost

```
for files in os.listdir('/home/pbaddam/Downloads/final_pkl_files'):
    df_16='/home/pbaddam/Downloads/final_pkl_files/{}'.format(files)
    dfout=pd.read_pickle(df_16)
    Common_data = dfout.iloc[:,[0,2,3,4,5,6,7]]
    Space = dfout.iloc[:,[0,2,3,6,7]]
    Year_and_Day_of_Year = dfout.iloc[:, [4,5,6,7]]
    Space_and_Year = dfout.iloc[:,[0,2,3,4,6,7]]
    Space_and_Day_of_Year = dfout.iloc[:,[0,2,3,5,6,7]]
    Year = dfout.iloc[:,[4,6,7]]
```

```
 Day_of_Year = dfout.iloc[:,[5,6,7]]
 datasets = [Common_data, Space, Year_and_Day_of_Year, Space_and_Year,
Space_and_Day_of_Year, Year, Day_of_Year]
 datasets_names = ['Common_data', 'Space', 'Year_and_Day_of_Year', '
Space_and_Year', 'Space_and_Day_of_Year', 'Year', 'Day_of_Year']

 # Implementing Block selection cross validation and Machine Learning
algorithm
 d = {}
 for (dataset, datasetname) in zip (datasets, datasets_names):
     folds = []
     i = np.arange(0,9,1)
     for k in i:
         x= dataset[dataset['Block']==int(k)]
         folds.append(x)


     feature_monotones = [-1,0,0,-1,0,1]
     params = {'monotone_constraints': '(' + ','.join([str(m) for m in
feature_monotones]) + ')'}
     model = xgb.XGBRegressor(random_state= 10, subsample= 0.8,
n_estimators= 100,min_weight_fraction_leaf= 0.3,
     min_samples_leaf= 35,min_sample_split= 500,min_child_weight= 5,
max_feature= 'auto',max_depth= 5,learning_rate= 0.05,
     gamma= 0.2,colsample_bytree= 0.5, **params)

     ytest = []
     ypredic = []

     for fold in range(len(folds)):
         train = folds.copy()
         test = folds[fold]
         del train[fold]
         train = pd.concat(train, sort=False)
         x_train = train.copy()
         x_test = test.copy()
         y_train = x_train.pop('Leaf_area').values
         y_test = x_test.pop('Leaf_area').values
         ytest.extend(y_test)
         model.fit(x_train, y_train)
         y_predicted = model.predict(x_test)
         ypredic.extend(y_predicted)


     print('r2_score_of/{}'.format(files.rsplit(".", 1)[ 0 ]),
datasetname, 'is', r2_score(ytest,ypredic))
```

```
        d['{}'.format(files.rsplit( ".", 1 )[ 0 ]),datasetname] = r2_score(
    ytest, ypredic)
```

LISTING A.18: Parameter Tuning for XGBoost

A.11 Feature importance plots

```
#modules related to bar plots
import os
import numpy as np
from matplotlib import pyplot as plt
```

```
#to plot feature importance plots for full dataset
for files in os.listdir('/home/pbaddam/Downloads/final'):
    df_17='/home/pbaddam/Downloads/final/{}'.format(files)
    dfout=pd.read_pickle(df_17)


    x= dfout[['lat', 'lon_sin', 'lon_cos', 'altitude', 'Year', 'Day_of_Year
    ']]
    y= dfout[['Leaf_area']]
    #**params is a dictionary where the tuned parameters are stored
    model = RandomForestRegressor(**params)
    model.fit(x,y)
    features=x.columns
    feature_names = x.columns.values

    importances = model.feature_importances_
    # It return the features which is more important at top in a
    decescending order and the less important feature at bottom
    indices = np.argsort(importances)
    #indice = np.argsort(importances)
    # to choose certain features use below code
    #indices=indice[1:7]
    plt.figure(figsize=(8,6))
    plt.title('{}'.format(files.rsplit( ".", 1 )[ 0 ]))
    plt.barh(range(len(indices)), importances[indices], color = 'gbrycm',
    align='center')
    plt.yticks(range(len(indices)), features[indices])
    #plt.yticks(range(len(indices)), [features[i] for i in indices])
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature')
    plt.ylim(-1, len(indices))
```

LISTING A.19: feature importance plots for full dataset

A.12 Partial dependency plots

```
#modules related to bar plots
import os
from matplotlib import pyplot as plt
from sklearn.ensemble.partial_dependence import partial_dependence, plot_partial_
from sklearn.inspection import plot_partial_dependence
from pdpbox import pdp, get_dataset, info_plots
```

```
#Partial dependency plots on full dataset
for files in os.listdir('/home/pbaddam/Downloads/final'):
    df_17='/home/pbaddam/Downloads/final/{}'.format(files)
    dfout=pd.read_pickle(df_17)
    x= dfout[['lat', 'lon_sin', 'lon_cos', 'altitude', 'Year', 'Day_of_Year
   ']]
    y= dfout[['Leaf_area']]
    model = RandomForestRegressor(**params)
    model.fit(x,y)
    features=x.columns
    #feature= features.drop(['Block'])
    feature_names = x.columns.values
    plot_partial_dependence(model, x, feature, feature_names,
   grid_resolution = 10)
    plt.subplots_adjust(left=8, right= 10, bottom = 3, top = 5, wspace
   =0.5, hspace = 0.5)
    plt.ylabel('Leaf_area in cm^2')
    plt.title('{}'.format(files.rsplit( ".", 1 )[ 0 ]))
```

LISTING A.20: pdp plots

A.13 Saptial Distribution of data on Cartopy map with Robinson projection

```
#modules related to cartopy and shape files
import shapefile
import cartopy.io.shapereader as shpreader
import cartopy.crs as ccrs
import cartopy
from shapely.geometry import point
from shapely.geometry import Polygon
import pyproj
```

```
#cartopy map visulization
#for files in os.listdir('OBG/Pramod/processed_pkl_files_with_blocks_&_DEM
    '):
for files in os.listdir('/home/pbaddam/Downloads/final_pkl_files'):
    final_df='/home/pbaddam/Downloads/final_pkl_files/{}'.format(files)
    df_10=pd.read_pickle(final_df)
    fig = plt.figure(figsize=(20, 20))
    ax = plt.axes(projection=ccrs.Robinson())
    ax.stock_img()
    ax.coastlines()
    ax.set_global()
    lons = df_10['lon']
    lats = df_10['lat']
    ax.scatter(lons,lats,color='red',marker='.', transform =ccrs.
    PlateCarree())
    ax.legend(labels=['Leaf_area'],loc = 3)
    ax.gridlines()
    ax.add_feature(cartopy.feature.LAND)
    ax.add_feature(cartopy.feature.OCEAN)
    ax.add_feature(cartopy.feature.COASTLINE)
    ax.add_feature(cartopy.feature.BORDERS)
    ax.add_feature(cartopy.feature.LAKES)
    ax.add_feature(cartopy.feature.RIVERS)
    #ax.add_feature(cartopy.feature.STATES, edgecolor='gray')
    plt.title('{}'.format(files.rsplit( ".", 1 )[ 0 ]))
    plt.show()
```

LISTING A.21: spatial distribution

A.14 Mixed effect Random Forest

A.14.1 Spatial joining of species with ecoregions

```
#modules required for MERF
import pandas as pd
from mpl_toolkits.basemap import Basemap
import numpy as np
import mapclassify.classifiers as classifiers
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

```
import geopandas as gpd
from geopandas import GeoDataFrame
import matplotlib.pyplot as plt
%matplotlib inline
from shapely.geometry import point
from shapely.geometry import Polygon
from descartes import PolygonPatch
from matplotlib.patches import Polygon
from matplotlib.collections import PathCollection
import shapefile
```

```
#First import the file of species
#pkl file of Impatiens capensis and its path
df = pd.read_pickle('/home/pbaddam/anaconda3/Impatiens_capensis_blocks.pkl'
    )

#Then converting the pandas dataframe to geopandas dataframe
#Spatial distribution of  impatiens capensis species using geopandas
gdf = gpd.GeoDataFrame(df, geometry=gpd.points_from_xy(df.lon,df.lat))
world= gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
cities= gpd.read_file(gpd.datasets.get_path('naturalearth_cities'))
ax =world[world.continent=='North America'].plot(color='white', edgecolor='
    black')
gdf.plot(ax=ax)
gdf.head()

#Then import shape file of ecoregions and its the file path
df1=gpd.read_file('/home/pbaddam/anaconda3/official/wwf_terr_ecos.shp')
#to check current projection of map
df1.crs
#plotting ecoregions data of world map
df1.plot(column='ECO_NAME', cmap=None)

# spatial distribution of impatiens capensis species on top of eco regions
f, ax = plt.subplots(1, figsize=(25,25))
df1.plot(ax=ax, column = 'ECO_NAME')
gdf.plot(ax=ax,column='Leaf_area',cmap=None, legend = True, scheme='
    quantiles', legend_kwds={'title': 'Leaf_Size in cm2','title_fontsize'
    :12,'loc': 'lower left'})
plt.title ('ECO regions for impatiens_capensis', fontsize= 25)
plt.show()
#f.savefig('ECO regions for impatiens capensis world.png', qaulity=100, dpi
    =300)
```

```python

#Then joining two datasets spatially (dataset of species and  eco regions
    dataset) based on geometry
#only common geometry points are taken
X = gpd.sjoin(df1, gdf, how="inner", op='intersects')
#only required columns are taken and created new dataset
cols = ['lat', 'lon', 'altitude', 'Year', 'Day_of_Year', 'Block', 'geometry
    ', 'ECO_NAME','ECO_ID', 'Leaf_area']
#newdatset with required features as columns
X2 = X[cols]
X3 = X2.reset_index()
X4 = X3.drop(columns='index')
#X3= X2.drop(X2.columns[], axis=1)
#removing and replacing the unwanted special characters from column '
    ECO_NAME'
X4['ECO_NAME'] = X4['ECO_NAME'].str.replace('\W', '_')
#new dataset with required features as columns is created again
X5=X4[['lat','lon','altitude', 'Year', 'Day_of_Year', 'ECO_NAME', 'ECO_ID',
     'Leaf_area']]
#assinging the constant value in order to account for random effects
X5['Z']= 1.0

#Then pkl file which contain yearly mean co2 values is imported
df_10= pd.read_pickle('/home/pbaddam/Downloads/Yearly_co2_mean.pkl')
df_11=df_10.rename(columns={'year':'Year'})
#created the new dataset by merging the CO2 to existing dataset
df_12= X5.merge(df_11, how='left')
#Then in the next cases the day of year is transformed into
    extraterrestrial radiation.
```

LISTING A.22: MERF

A.14.2 Transforming Day of year to Extraterristial radiation

```python
import xarray as xr
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as pltp
import scipy
import scipy as sp
#import xesmf as xe
pd.set_option('display.max_colwidth', -1)
import math
```

```python
def _arraytest(*args):
    rargs=[]
    for a in args:
        if isinstance(a, (list, tuple)):
            rargs.append(scipy.array(a))
        else:
            rargs.append(a)
    if len(rargs) == 1:
        return rargs[0] # no unpacking if single value, return value i/o
list
    else:
        return rargs

def sun_NR(doy=scipy.array([]),\
              lat=float):
    # Test input array/value
    doy,lat = _arraytest(doy,lat)

    # Set solar constant [W/m2]
    S = 1367.0 # [W/m2]
    # Print warning if latitude is above 67 degrees
    if abs(lat) > 67.:
        print('WARNING: Latitude outside range of application (0-67 degrees
).\n)')
    # Convert latitude [degrees] to radians
    latrad = lat * math.pi / 180.0
    # calculate solar declination dt [radians]
    dt = 0.409 * scipy.sin(2 * math.pi / 365 * doy - 1.39)
    # calculate sunset hour angle [radians]
    ws = scipy.arccos(-scipy.tan(latrad) * scipy.tan(dt))
    # Calculate sunshine duration N [h]
    N = 24 / math.pi * ws
    # Calculate day angle j [radians]
    j = 2 * math.pi / 365.25 * doy
    # Calculate relative distance to sun
    dr = 1.0 + 0.03344 * scipy.cos(j - 0.048869)
    # Calculate Rext
    Rext = S * 86400 / math.pi * dr * (ws * scipy.sin(latrad) * scipy.sin(
dt)\
              + scipy.sin(ws) * scipy.cos(latrad) * scipy.cos(dt))
    return Rext

def Filename_check(row):
    if abs(row['lat']) < 67:
        return sun_NR(row['Day_of_Year'],row['lat'])
    else:
        return np.nan

# Then radiation is assigned to existing dataset by creating a column '
    Extraterrestrial radiation' in existing dataset
df_12.loc[:,'Extraterrestrial radiation'] = df_12.apply(Filename_check,
    axis=1)
#Then in the next cases the creating blocks for block cross validation and
    performing hyper parameter tuning is done.
```

LISTING A.23: code for transforming day of year to radiation

A.14.3 Fitting MERF model to data

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import scipy as sp
from merf.utils import MERFDataGenerator
from merf.merf import MERF
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_regression
```

```
#Blocks are assigned along with MERF model
# Full model
d_1 = {}
folds = []
i = np.arange(0,9,1)
for k in i:
    x= final[final['Block']==int(k)]
    folds.append(x)
#mixed effect random forest model
mrf = MERF(rf_params=rf_params1)

feature_importance=pd.DataFrame()
ytest = []
```

```python
ypredic = []
for fold in range(len(folds)):
    train = folds.copy()
    test = folds[fold]
    del train[fold]
    train = pd.concat(train, sort=False)
#fixed features
    x_train = train[['Year', 'Extraterrestrial radiation']]
    x_test = test[['Year',  'Extraterrestrial radiation']]
#target feature
    y_train = train['Leaf_area']
    y_test = test['Leaf_area']
#random feature
    z_train = train[['Z']]
    z_test = test[['Z']]
#cluster_id
    clusters_train = train['ECO_ID']
    clusters_test = test['ECO_ID']
    ytest.extend(y_test)
    mrf.fit(x_train, z_train, clusters_train, y_train)
    y_predicted = mrf.predict(x_test, z_test, clusters_test)
    ypredic.extend(y_predicted)

# This is the R2 score for full model
print('Total_r2_score_of_model',r2_score(ytest,ypredic))
d_1['Total_r2_score_of_model'] = r2_score(ytest, ypredic)
```

LISTING A.24: example MERF model which is very similar to random forest model

A.14.4 Plotting using MERF model

```python
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import seaborn as seabornInstance
from pdpbox import pdp
import pdpbox
```

```python
# for plotting histogram of random effects
mrf.trained_b.hist()
plt.title('Distribtuion of random effects b')

# for feature importance plot
features=['Year', 'Extraterrestrial radiation']
```

```python
feature_names = ['Year',  'Extraterrestrial radiation']
importances = mrf.trained_rf.feature_importances_
indices = np.argsort(importances)
plt.figure(figsize=(8,6))
plt.barh(range(len(indices)), importances[indices], color = 'br', align='
    center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.ylim(-1, len(indices))

#for pdp plots
final1= final[['Year', 'Extraterrestrial radiation','ECO_ID']]
#creating new column with prediction values from MERF model
final1['predicted_leaf_area']= ypredic
b=mrf.trained_b.reset_index()
b1=b.rename(columns={'index':'ECO_ID',0:'random_effect_b'})
final2=final1.merge(b1)

data = final2.copy()
features=['Year', 'Extraterrestrial radiation']
model = mrf.trained_rf
target=['predicted_leaf_area']

for f in features:
    pdp_rf = pdp.pdp_isolate(model=model,dataset=data,model_features=
    features,feature=f)
    fig, axes = pdp.pdp_plot(pdp_rf,f,plot_lines=True,ncols=data['
    random_effect_b']],center=False, x_quantile=False,
                                plot_pts_dist=True)
```

LISTING A.25: code for plotting with MERF model

Bibliography

[1] URL: https://www.ismll.uni-hildesheim.de/lehre/ml-09w/script/ml-04-decisiontrees-2up.pdf (visited on 02/15/2020).

[2] URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (visited on 01/15/2020).

[3] URL: https://scikit-learn.org/stable/modules/partial_dependence.html (visited on 02/19/2020).

[4] URL: https://geopandas.org/ (visited on 02/19/2020).

[5] URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (visited on 02/01/2020).

[6] URL: https://python-graph-gallery.com/venn-diagram/ (visited on 04/08/2020).

[7] URL: https://www.co2.earth/historical-co2-datasets (visited on 02/25/2020).

[8] URL: https://www.cabi.org/isc/datasheet/12648 (visited on 04/20/2020).

[9] URL: https://www.fs.fed.us/wildflowers/plant-of-the-week/impatiens_capensis.shtml (visited on 06/09/2020).

[10] URL: http://www.plant-identification.co.uk/skye/solanaceae/solanum-dulcamara.htm (visited on 04/09/2020).

[11] URL: https://www.flickr.com/photos/evelynfitzgerald/3804420858 (visited on 04/12/2020).

[12] Meerb. (2003). Impatiens capensis. In: Plants For A Future. URL: https://pfaf.org/user/Plant.aspx?LatinName=Impatiens+capensis (visited on 04/09/2020).

[13] Esri. (2012). How Spatial Autocorrelation (Global Moran's I) works. In: ArcGIS Online Ressource - Desktop 10, pp.5–9. URL: https://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-statistics-toolbox/h-how-spatial-autocorrelation-moran-s-i-spatial-st.htm.

[14] Xgboost developers (2019). XGBoost Documentation. In: xgboost 1.0.0-SNAPSHOT documentation. URL: https://xgboost.readthedocs.io/en/latest/ (visited on 02/19/2020).

[15] Jain A. (2016). A Complete Tutorial on Ridge and Lasso Regression in Python. In: Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2016/01/ridge-lasso-regression-python-complete-tutorial/ (visited on 02/25/2020).

[16] Sydorchuk A., Smart J., Roebling R., Arya S., Mount D., Sunday D., Seweryn H. W., and Sobczak and M. GeoDa An Introduction to Spatial Data Analysis. In: Journal of Graphics Tools, pp.1–6. URL: https://geodacenter.github.io/.

[17] Tianqi C and Carlos G. (2016). XGBoost: A scalable tree boosting system. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Vol. 13-17, pp.785–794. ISBN: 9781450342322. DOI: 10.1145/2939672.2939785. arXiv: 1603.02754. URL: http://dx.doi.org/10.1145/2939672.2939785.

[18] Christoph M. (2019). 5.1 Partial Dependence Plot (PDP). In: Interpretable Machine Learning. URL: https://christophm.github.io/interpretable-ml-book/pdp.html (visited on 02/19/2020).

[19] Georgios D. (2019). Decision Tree Regressor explained in depth. In: GDCoder-Data Science. URL: https://gdcoder.com/decision-tree-regressor-explained-in-depth/ (visited on 01/20/2020).

[20] Pieter D.F., Bente J. G., Francisco R.S., Annette K., Olivier C., Guillaume D., Hanne D. K., An D. S., Martin D., Ove E., Robert G., Martin H., Jonathan L., Jan P., David A. C., and Kris V. (2013). Latitudinal gradients as natural laboratories to infer species responses to temperature. In: Journal of Ecology. Vol. (101). 3, pp.784–795. ISSN: 00220477. DOI: 10.1111/1365-2745.12074.

[21] Malézieux E., Lamanda N., Laurans M., and Tassin J. (2007). Plant functional traits and types: their relevance for a better understanding of the functioning and properties of agroforestry systems. In: International Symposium on Multi-strata Agroforestry Systems with Perennial Crops, pp.17–21. URL: https://agritrop.cirad.fr/540962/.

[22] Tadashi F. and David A. W. (2005). Long-term ecological dynamics: Reciprocal insights from natural and anthropogenic gradients. In: The Royal Society. Vol. (272). 1577. The Royal Society, pp.2105–2115 ISSN: 14712970. DOI: 10.1098/rspb.2005.3277.

[23] Jitendra G., Abdelaziz T., and Bassem B. (2019). Measuring Morphological Functional Leaf Traits From Digitized Herbarium Specimens Using TraitEx Software. In: Biodiversity Information Science and Standards. Vol. (3). Pensoft Publishers, e37091. DOI: 10.3897/biss.3.37091. URL: https://doi.org/10.3897/biss.3.37091.

[24] DRAKOS G. (2019). Random Forest Regressor explained in depth. In: GDCoder-Data Science. URL: https://gdcoder.com/random-forest-regressor-explained-in-depth/ (visited on 02/19/2020).

[25] Ahlem H., François B., and Denis L. (2014). Mixed-effects random forest for clustered data. In: Journal of Statistical Computation and Simulation. Vol. (84). 6, pp.1313–1328. ISSN: 00949655. DOI: 10.1080/00949655.2012.741599.

[26] Brownlee J. (2020). Gradient Boosting with Scikit-Learn, XGBoost, LightGBM, and CatBoost. In: Machine Learning Mystery. URL: https://machinelearningmastery.com/gradient-boosting-with-scikit-learn-xgboost-lightgbm-and-catboost/ (visited on 04/08/2020).

[27] Aarshay J. (2016). Complete guide to parameter tuning in XGBoost (with codes in Python). In: Analytics Vidhya, pp.1–20. URL: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.

[28] Thomas G. J.(2013). Salix glauca (Grey Willow) - Featured Native Plant of the Week. In: Tree and Landscape Company-Jackson Hole.
URL: https://www.treeandlandscapecompany.com/native-plant-of-the-week/salix-glauca-grey-willow-featured-native-plant-of-the-week/ (visited on 04/20/2020).

[29] Evan L. (2017). Ensemble Methods in Machine Learning: What are They and Why Use Them?. In: Towards Data Science. (2017). URL: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f (visited on 02/19/2020).

[30] Pierre L. and J. (2012) Louis L. F. Numerical Ecology. In: Elsevier-Google Books. URL: https://books.google.rw/books?id=DKlUIQcHhOsC&printsec=copyright&hl=no#v=onepage&q=Variation%20partitioning&f=false (visited on 02/19/2020).

[31] Yaoqi L., Dongting Z., Nawal S., Xiaoting X., Qinggang W., Wen J., and Zhiheng W. (2019). Spatiotemporal variation in leaf size and shape in response to climate. In: Journal of Plant Ecology. Vol. (13). 1. Oxford Academic, pp.87–96. DOI: 10.1093/JPE/RTZ053.

[32] Christoph M. (2019). Permutation Feature Importance. In: Interpretable Machine Learning. URL: https://christophm.github.io/interpretable-ml-book/feature-importance.html (visited on 02/19/2020).

[33] James I.L. M. and Lawlor D. W.(1999). Interactions between increasing CO2 concentration and temperature on plant growth. In: Plant, Cell and Environment. Vol. (22). 6. John Wiley & Sons, Ltd, pp.659–682. ISSN: 01407791. DOI: 10.1046/j.1365-3040.1999.00443.x.

[34] Luiza M. A. (2016). Identifying plant functional traits to assist ecological intervention in a drying landscape, Edith Cowan University. URL: https://ro.ecu.edu.au/cgi/viewcontent.cgi?article=2946&context=theses.

[35] Vishal M. (2018). XGBoost Algorithm: Long May She Reign!. In: Towards Data Science. URL: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (visited on 02/19/2020).

[36] Mai M. (2019). Dulcamara. In: Homopathische Anwendung - NetDoktor. URL: https://www.netdoktor.de/homoeopathie/dulcamara/ (visited on 04/04/2020).

[37] David M. O., Eric D., Eric D. W., Neil D. B., George V. N. P., Emma C. U., Jennifer A. D., Illanga I., Holly E. S., John C. M., Colby J. L., Thomas F. A., Taylor H. R., Yumiko K., John F. L., Wesley W. W., Prashant H., and Kenneth R. K. (2001). Terrestrial Ecoregions of the World: A New Map of Life on Earth. In: BioScience. 11. Oxford Academic, pp.933–938. ISSN: 0006–3568. DOI: 10.1641/0006-3568(2001)051[0933:TEOTWA]2.0.CO;2.

[38] Legendre P. (2008). Studying beta diversity: ecological variation partitioning by multiple regression and canonical analysis. In: Journal of Plant Ecology. Vol. (1). 1. Oxford University Press (OUP), pp.3–8. ISSN: 1752–9921. DOI: 10.1093/jpe/rtm001. URL: https://academic.oup.com/jpe/article-lookup/doi/10.1093/jpe/rtm001.

[39] David R. R., Volker B., Simone C., Mark S. B., Jane E., Gurutzeta G. A., Severin H., José J. L. M., Boris S., Wilfried T., David I. W., Brendan A. W., Florian H., and Carsten F. D. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. In: Ecography. Vol. (40). 8. Blackwell Publishing Ltd, pp.913–929. ISSN: 16000587. DOI: 10.1111/ecog.02881. URL: http://doi.wiley.com/10.1111/ecog.02881.

[40] Juhi R. (2019). Simple guide for ensemble learning methods. In: Towards Data Science. URL: https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2 (visited on 02/19/2020).

[41] Sunil R. (2018). Improve Your Model Performance using Cross Validation (in Python/R). In: Analytics Vidhya, pp.1–29. URL: https://www.analyticsvidhya.com/blog/2018/05/improve-model-performance-cross-validation-in-python-r/.

[42] Dey S. (2017). Mixed Effects Random Forests in Python.In: Towards Data Science. URL: https://towardsdatascience.com/mixed-effects-random-forests-6ecbb85cb177 (visited on 03/15/2020).

[43] Juliana S., Isiane M. D. S., Ricardo A. C., Jhonatan G. D. S., Thainá L.P. S., Janisson W. D. S., Richard J. L., and Ana C.M. M. (2017). Drier climate shifts leaf morphology in Amazonian trees. In: Oecologia. Vol. (185). 3. Springer Verlag, pp.525–531 ISSN: 00298549. DOI: 10.1007/s00442-017-3964-7. URL: https://pubmed.ncbi.nlm.nih.gov/29038861/.

[44] Mangale S. (2017). Decision Tree-Pruning-Cost Complexity Method. In: Medium. URL: https://medium.com/@sanchitamangale12/decision-tree-pruning-cost-complexity-method-194666a5dd2f (visited on 03/20/2020).

[45] Christine Scoffoni, Michael Rawls, Athena Mckown, Hervé Cochard, and Lawren Sack. "Decline of leaf hydraulic conductance with dehydration: Relationship to leaf size and venation architecture". In: *Plant Physiology* 156.2 (2011), pp. 832–843. ISSN: 00320889. DOI: 10.1104/pp.111.173856.

[46] Abdelaziz T., Jitendra G., and Bassem B. (2019). Manual for TraitEX. From MIRACL/CRNS-University of Sfax, Tunisia, German Centre for Integrative Biodiversity Research (iDiv), and Friedrich Schiller University Jena, Germany. pp.1–10. URL: https://bitbucket.org/traitExTool/traitextool/src/master/.

[47] Konstantin T. (2017). matplotlib-venn. In: Python Package Index. URL: https://pypi.org/project/matplotlib-venn/ (visited on 01/10/2020).

[48] Camilla W., Stefano C., Giandiego C., Sandor B., Marco G., Francesco S., and Roberto C. (2013). Intraspecific phenotypic variability of plant functional traits in contrasting mountain grasslands habitats. In: Biodiversity and Conservation. Vol. (22). 10, pp.2353–2374. ISSN: 09603115. DOI: 10.1007/s10531-013-0484-6.

[49] Charles G. W., Elizabeth R. E., Richard B. P., Charles C. D., Katelin D. P., Amanda S. G., Jenn M. Y., Gil N., Susan J. M., Natalie L. R., Tim H. S., and Pamela S. S. (2017). Old Plants, New Tricks: Phenological Research Using Herbarium Specimens. In: Trends in Ecology and Evolution. Vol. (32). 7, pp.531–546. ISSN: 01695347. DOI: 10.1016/j.tree.2017.03.015. URL: http://dx.doi.org/10.1016/j.tree.2017.03.015.

[50] Ian J. W., Ning D., Vincent M., I. Colin P., Mark W., Sandra D., Rachael V. G., Bonnie F. J., Robert K., Elizabeth A. L., Michelle R. L., Ülo N., Peter B. Reich, Lawren S., Rafael V., Han W., and Peter W. (2017). Global climatic drivers of leaf size. In:Science. Vol.(357). 6354, pp..917–921. ISSN:10959203. DOI: 10.1126/science.aal4760.

[51] Will K. (2018). Hyperparameter Tuning the Random Forest in Python. In: Towards Data Science. URL: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 (visited on 02/20/2020).

[52] Carson Y. (2018). How does the popular XGBoost and LightGBM algorithm enforce monotonic constraint. In:Towards Data Science. URL: https://towardsdatascience.com/how-does-the-popular-xgboost-and-lightgbm-algorithms-enforce-monotonic-constraint-cf8fce797acb (visited on 02/19/2020).

[53] Sohaib Y., Claus W., Robert H., Stefan D.r, Thomas H., Bernhard S., and Marco S. (2018). Taxon and trait recognition from digitized herbarium specimens using deep convolutional neural networks. In: Botany Letters. Vol. 165. 3-4, pp.377–383. ISSN: 23818115. DOI: 10.1080/23818107.2018.1446357.

[54] Pavlov L. Y. (2019). Random forests. In: Random Forests, pp.1–122. DOI: 10.1007/978-3-662-56776-0_10.

[55] Zixuan Z. (2019).Boosting Algorithms Explained. In:Towards Data Science. URL: https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30 (visited on 02/19/2020).