**KNOWLEDGE EXTRACTION IN AGRICULTURE USING MACHINE LEARNING ALGORITHMS**

A BRIEF REPORT

***Rukmangadan D***

***SR.NO.13305***

***MTECH/WREE***

***Supervised by: Prof. M. Sekhar***

**KNOWLEDGE EXTRACTION IN AGRICULTURE USING MACHINE LEARNING ALGORITHMS**

***Report submitted to***

***Indian Institute of Science, Bangalore***

***for the award of the degree***

***of***

**Master of Technology**

**in Civil Engineering with Specialization**

**in “Water Resources and Environmental Engineering”**

***by***

**Rukmangadan D**



**DEPARTMENT OF CIVIL ENGINEERING**

**INDIAN INSTITUTE OF SCIENCE, BANGALORE**

**FEB 2018**

© 2018, Rukmangadan D. All rights reserved.

**DECLARATION**

I certify that

1. the work contained in this report is original and has been done by me under the guidance of my supervisor.
2. the work has not been submitted to any other Institute for any degree or diploma.
3. I have followed the guidelines provided by the Institute in preparing the report.
4. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.
6. I hereby declare that I am the sole author of this thesis. I authorize Indian Institute of Science to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature of the Student

**CERTIFICATE**

This is to certify that the Dissertation Report entitled, “**KNOWLEDGE EXTRACTION IN AGRICULTURE USING MACHINE LEARNING ALGORITHMS**” submitted by Mr. “Rukmangadan D” to Indian Institute of Science, Bangalore, India, is a record of bonafide Project work carried out by him/her under my/our supervision and guidance and is worthy of consideration for the award of the degree of Master of Technology in Civil Engineering with Specialization in “Water Resources and Environmental Engineering” of the Institute.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor

Date:

**ACKNOWLEDGEMENTS**

I would like to take this opportunity to express my deep sense of gratitude and profound feeling of admiration to my thesis supervisor. Many thanks to all those who helped me in this work.

**CONTENTS**

Title Page i

Declaration ii

Certificate by the Supervisor iii

Acknowledgement iv

Chapter 1 Introduction 1

Chapter 2 Review of Literature 2

Chapter 3 Methodology, Analysis and Extraction 4

Chapter 4 Results and Discussions 31

Chapter 5 Conclusions and Future Scope of Study 33

References 34

**Chapter 1**

**Introduction**

Machine Learning algorithms are being widely used nowadays in all fields of science wherever huge amount of data is generated and agriculture is also not an exception for this. Farmers in many parts of India are largely dependent on timely rainfall for harvest and subsequent profits. Uncertainty surrounding this phenomenon has also haunted them since the beginning of civilization. Over time however, this uncertainty had reduced significantly as farmers back in the day could almost accurately plant crops based on previous experience with weather conditions. This wisdom has been passed on from one generation of farmers to the other. Gradual onset of global warming and climate changes over the last century has slowly-yet steadily put this wisdom out of use. As for rain-fed farmers preparing for agriculture, soil-water equation is fragile and any delay in rainfall could easily destroy the harvest. When age old systems fail, look to the future.

One of the fastest growing areas under the discipline of “artificial intelligence” is machine learning. And this technology is being deployed across modern agriculture to create solutions with greater accuracy and at unprecedented scale. Of the millions of combinations, advanced software greatly narrows the search. Machine learning can be used draw conclusions from various sets of raw data. Researchers in modern agriculture are testing their theories at greater scale and helping make more accurate, real-time predictions. Modern agriculture has the potential to discover even more ways to conserve water, use nutrients and energy more efficiently, and adapt to climate change. With the advent of technology, numerous advancements have taken place so that now we are in a place where we can measure field soil moisture using Remote Sensing satellites. The main goal in this study is to arrive at an interesting pattern from the variation between relative soil moisture obtained from the satellite data and field soil moisture obtained from ‘Berambadi’ Region.

**Chapter 2**

**Literature Review**

Though we can rely on moisture data given by satellite to a certain extent, we cannot rely on it completely as it is affected by many factors and it may not give the actual soil moisture. H. McNairna,\*, C. Duguayb, B. Briscoc, T.J. Pultza examined the effect of soil and crop residue characteristics on polarimetric radar response. This study examines the sensitivity of linear polarizations and polarimetric parameters to conditions present on agricultural fields during the period of preplanting and postharvest. The co-polarizations signature plots are also discussed. Results indicate that the dominant scattering mechanism from these fields varies depending on the type and amount of residue cover, and whether the crop had been harvested. Radar parameters most sensitive to volume and multiple scattering perform best at characterizing these surface conditions. The scattering mechanisms associated with standing senesced vegetation, no-till fields, and tilled fields varied. Double-bounce, multiple, and volume scattering were all present in standing vegetation, while for no-till fields multiple scattering dominated. The pedestal height was also unique for each of these classes, with larger pedestals associated with standing crops and no-till fields. This confirms the sensitivity of pedestal height to multiple and volume scattering.

Jun Wen, Zhongbo Su examined that the radar backscattering coefficient is mainly determined by surface soil moisture, vegetation and land surface roughness under a given configuration of the satellite sensor. It is observed that the temporal variations of the three variables are different, the variation of vegetation and roughness are at the longer temporal scales corresponding to climate and cultivation practices, while soil moisture varies at a shorter temporal scale in response to weather forcing. Relative soil moisture is a function of field soil moisture and saturation capacity of the soil. The results show that the estimated relative soil moisture corresponds closely to vegetation and land surface roughness.

B.J. Choudury, T.J. Schmugge, R.W. Newton and A.Chang studied the effect of surface roughness on the brightness temperature of a moist terrain through the modification of Fresnel reflection coefficient and using the radiative transfer equation. The modification involves introduction of a single parameter to characterize the roughness. It is shown that this parameter depends on both the surface height variance and the horizontal scale of the roughness. Model calculations are in good quantitative agreement with the observed dependence of the brightness temperature on the moisture content in the surface layer.

J.R. Wang, P.E. O'Neill, T.J. Jackson, E.T. Engman conducted experiment on remote sensing of soil moisture content was conducted over bare fields with microwave radiometers at the frequencies of 1.4 GHz, 5 GHz, and 10.7 GHz. Three bare fields with different surface roughnesses and soil textures were prepared for the experiment. Ground truth acquisition of soil temperatures and moisture contents for 5 layers down to the depths of 15 cm was made concurrently with radiometric measurements. The experimental results show that the effect of surface roughness is to increase the soils' brightness temperature and to reduce the slope of regression between brightness temperature and moisture content. The slopes of regression for soils with different textures are found to be comparable, and the effect of soil texture is reflected in the difference of regression line intercepts at brightness temperature axis.

L. Breiman Bagging predictors is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote when predicting a class. The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets. Tests on real and simulated data sets using classification and regression trees and subset selection in linear regression show that bagging can give substantial gains in accuracy. The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

Thus, it is seen that the relative soil moisture determined by the satellites cannot be blindly correlated with the field soil moisture measured manually even though the saturation capacity of the soil is known. The main factors that lie in between the correlation of relative soil moisture and field soil moisture are the type of crop, crop coverage or crop growth basically how much the crop has covered the field at a given instance and also to the soil class based on texture and type of soil to which it belongs.

**Chapter 3**

**Methodology, Analysis and Extraction**

The field soil moisture data is obtained from 112 sites, from each site manually. These are the sites where the surface soil moisture data is available for year 2016 and 2017.

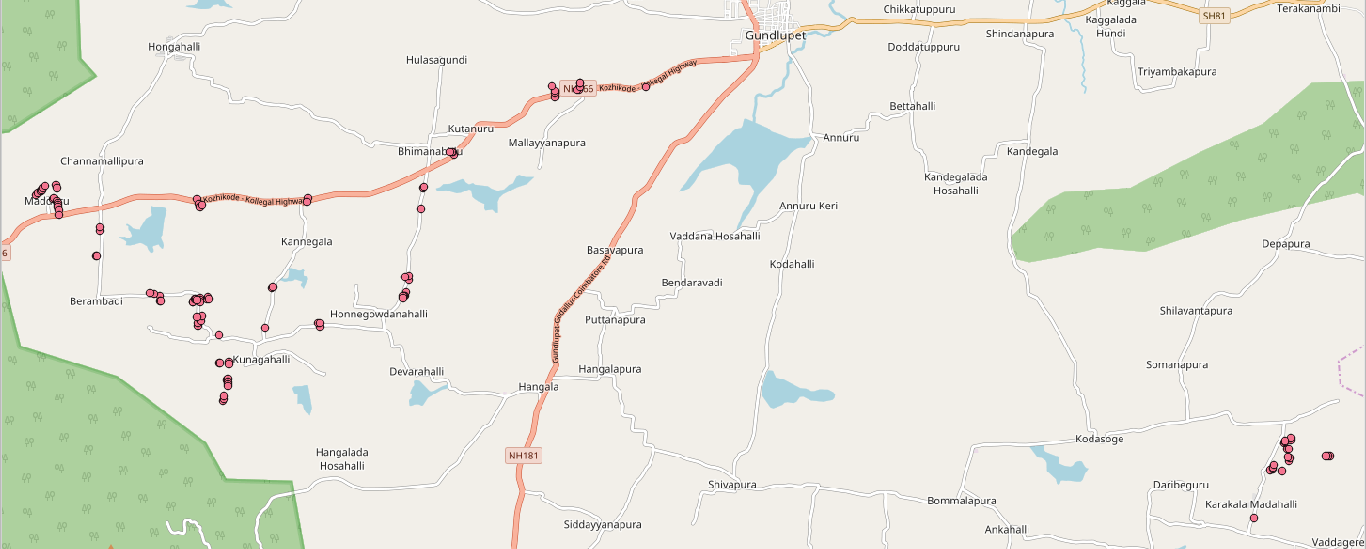


Image Showing 112 sites where soil moisture is measured routinely

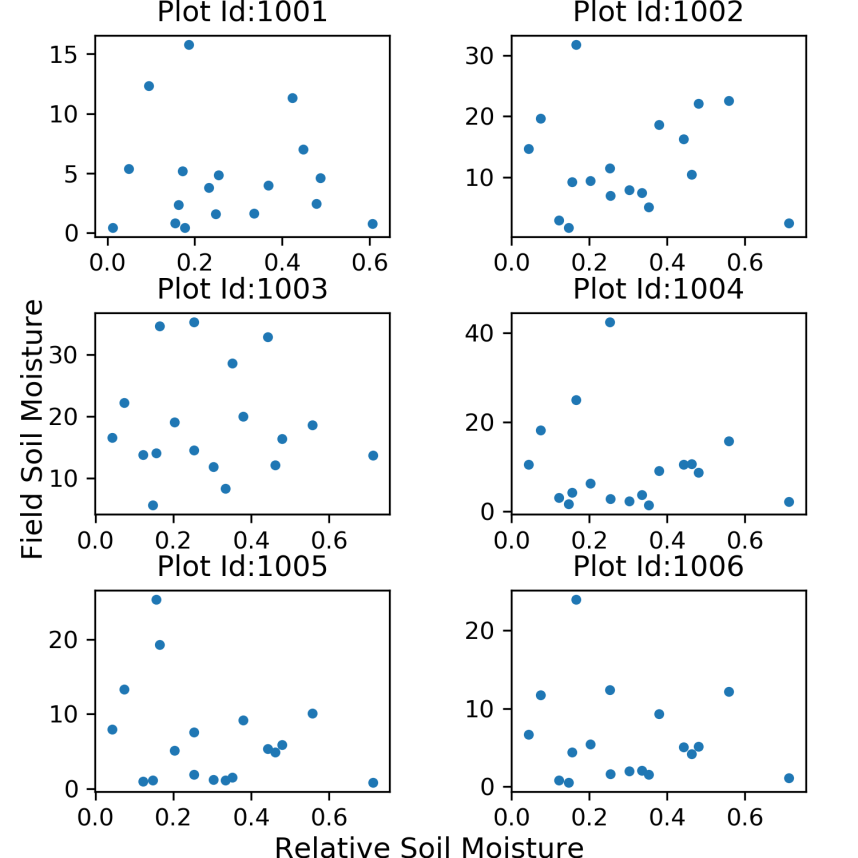
Out of the 112 sites, 92 sites fall on the Berambadi region of Gundlupet Taluk and 20 sites fall in Vaddagare region of Koratagere Taluk.

Each site is given a Plot ID for analysis purpose. The GPS coordinates of all the 112 sites are listed in the table below:

| **Plot\_Id** | **Latitude** | **Longitude** | **Plot\_Id** | **Latitude** | **Longitude** |
| --- | --- | --- | --- | --- | --- |
| 1001 | 11.79922 | 76.668 | 1057 | 11.78055 | 76.55742 |
| 1002 | 11.79856 | 76.65578 | 1058 | 11.7807 | 76.55765 |
| 1003 | 11.79863 | 76.65544 | 1059 | 11.78117 | 76.55781 |
| 1004 | 11.79923 | 76.65604 | 1060 | 11.78136 | 76.55815 |
| 1005 | 11.79984 | 76.656 | 1061 | 11.78165 | 76.56029 |
| 1006 | 11.79986 | 76.65602 | 1062 | 11.78109 | 76.56032 |
| 1007 | 11.79728 | 76.65138 | 1063 | 11.77904 | 76.55967 |
| 1008 | 11.79809 | 76.65148 | 1064 | 11.77926 | 76.55979 |
| 1009 | 11.79844 | 76.65142 | 1065 | 11.7787 | 76.56049 |
| 1010 | 11.79927 | 76.65085 | 1066 | 11.77832 | 76.56054 |
| 1011 | 11.787 | 76.63294 | 1067 | 11.77797 | 76.56057 |
| 1012 | 11.78756 | 76.63284 | 1068 | 11.77771 | 76.5606 |
| 1013 | 11.78752 | 76.6326 | 1069 | 11.77713 | 76.56074 |
| 1014 | 11.7875 | 76.63216 | 1070 | 11.74682 | 76.59141 |
| 1015 | 11.78108 | 76.62736 | 1071 | 11.74689 | 76.5917 |
| 1016 | 11.77725 | 76.62684 | 1072 | 11.74646 | 76.59168 |
| 1017 | 11.7624 | 76.62413 | 1073 | 11.74611 | 76.59158 |
| 1018 | 11.76174 | 76.62406 | 1074 | 11.74571 | 76.59169 |
| 1019 | 11.76173 | 76.62377 | 1075 | 11.74298 | 76.59072 |
| 1020 | 11.76173 | 76.62354 | 1076 | 11.74327 | 76.59087 |
| 1021 | 11.75631 | 76.60838 | 1077 | 11.74382 | 76.59097 |
| 1022 | 11.75692 | 76.60808 | 1078 | 11.74978 | 76.58994 |
| 1023 | 11.75693 | 76.60837 | 1079 | 11.74981 | 76.59027 |
| 1024 | 11.75613 | 76.59836 | 1080 | 11.74996 | 76.5918 |
| 1025 | 11.76326 | 76.59971 | 1081 | 11.74967 | 76.59184 |
| 1026 | 11.76336 | 76.59979 | 1082 | 11.72205 | 76.77913 |
| 1027 | 11.75485 | 76.58993 | 1083 | 11.77626 | 76.56072 |
| 1028 | 11.75639 | 76.58618 | 1084 | 11.73057 | 76.78211 |
| 1029 | 11.75696 | 76.58621 | 1085 | 11.73088 | 76.78255 |
| 1030 | 11.75735 | 76.58667 | 1086 | 11.73107 | 76.78261 |
| 1031 | 11.76143 | 76.58652 | 1087 | 11.73156 | 76.78283 |
| 1032 | 11.76101 | 76.58652 | 1088 | 11.7305 | 76.78417 |
| 1033 | 11.76069 | 76.58655 | 1089 | 11.73321 | 76.79298 |
| 1034 | 11.76132 | 76.58789 | 1090 | 11.73319 | 76.79266 |
| 1035 | 11.76166 | 76.58801 | 1091 | 11.73318 | 76.7923 |
| 1036 | 11.76118 | 76.58812 | 1092 | 11.73217 | 76.78549 |
| 1037 | 11.7607 | 76.58531 | 1093 | 11.73271 | 76.78565 |
| 1038 | 11.76074 | 76.58531 | 1094 | 11.73299 | 76.78532 |
| 1039 | 11.76127 | 76.58556 | 1095 | 11.73435 | 76.78511 |
| 1040 | 11.76129 | 76.5858 | 1096 | 11.73442 | 76.78544 |
| 1041 | 11.76101 | 76.58595 | 1097 | 11.73601 | 76.78595 |
| 1042 | 11.76172 | 76.57917 | 1098 | 11.73629 | 76.78589 |
| 1043 | 11.76088 | 76.57921 | 1099 | 11.73549 | 76.78456 |
| 1044 | 11.76086 | 76.57944 | 1100 | 11.7357 | 76.78471 |
| 1045 | 11.76213 | 76.57811 | 1101 | 11.73586 | 76.78471 |
| 1046 | 11.76885 | 76.56762 | 1102 | 11.57372 | 76.78369 |
| 1047 | 11.769 | 76.56771 | 1103 | 11.78131 | 76.62754 |
| 1048 | 11.77918 | 76.58606 | 1104 | 11.76464 | 76.62472 |
| 1049 | 11.77825 | 76.58636 | 1105 | 11.76534 | 76.62481 |
| 1050 | 11.77766 | 76.58646 | 1106 | 11.76518 | 76.62397 |
| 1051 | 11.77808 | 76.58685 | 1107 | 11.76144 | 76.62363 |
| 1052 | 11.77931 | 76.60627 | 1108 | 11.75825 | 76.5869 |
| 1053 | 11.77863 | 76.60614 | 1109 | 11.758 | 76.58597 |
| 1054 | 11.77993 | 76.55673 | 1110 | 11.76232 | 76.57731 |
| 1055 | 11.7799 | 76.55658 | 1111 | 11.77346 | 76.56834 |
| 1056 | 11.78011 | 76.55695 | 1112 | 11.77403 | 76.56823 |

While reviewing the literature, it is noted that relative soil moisture is a function of Field Soil Moisture (FSM) and saturation capacity of the soil. Hence, at a particular site, the relative soil moisture should have given a proper correlation with field soil moisture, but it has not.

Here, ‘Relative Soil Moisture’ (RSM) refers to the soil moisture data obtained through satellite which is provided by VATI project of ‘Aapah innovations’. ‘Field Soil Moisture’ (FSM) refers to soil moisture measured from the site manually.



Plot showing Field Soil Moisture vs Relative Soil Moisture (for 6 sites)

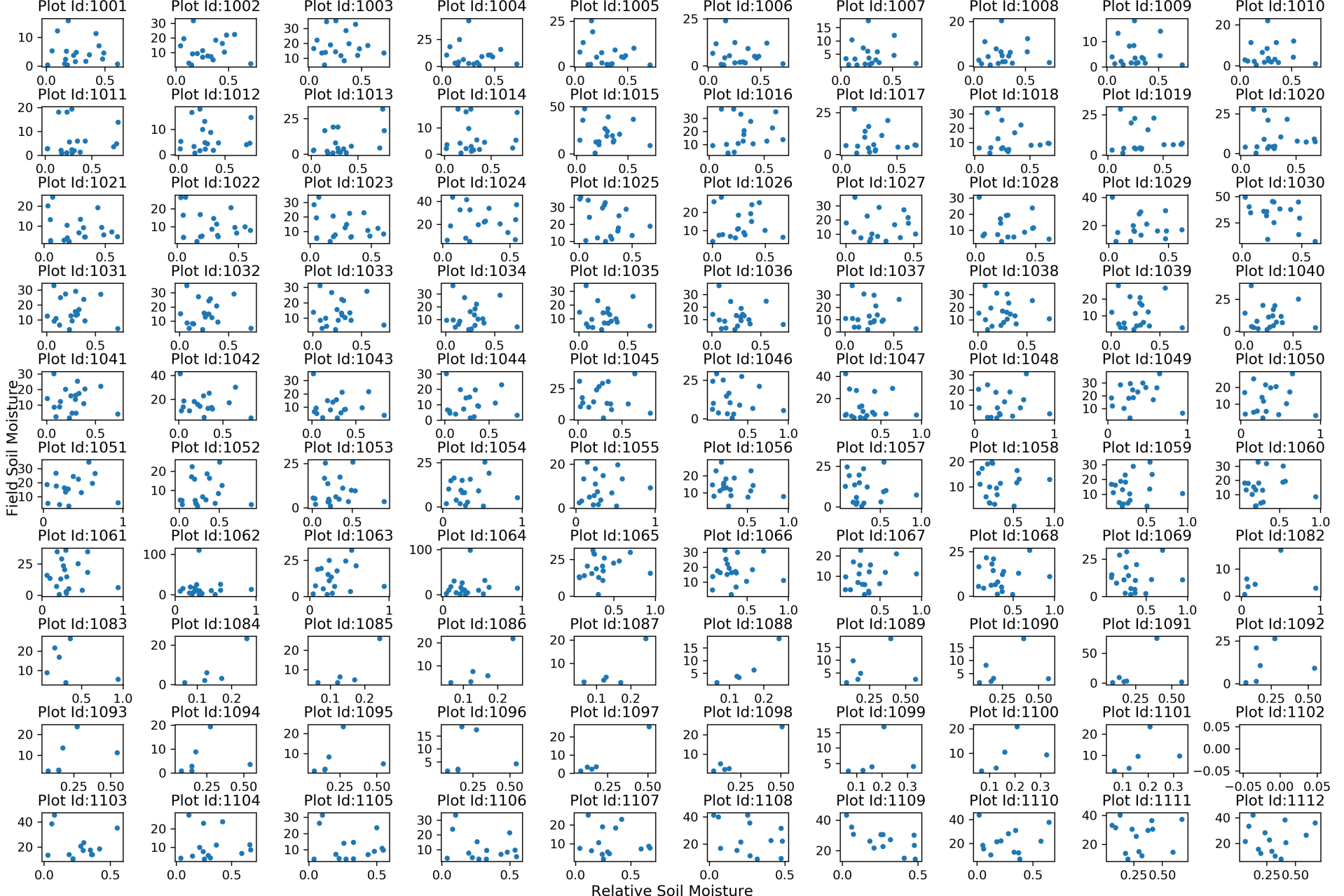
But, when we try to make scatter plots of ‘relative soil moisture’ vs ‘field soil moisture’, it does not show a good correlation. And, this leads to further thought why such variation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Plot\_ID | Correlation | Plot\_ID | Correlation | Plot\_ID | Correlation | Plot\_ID | Correlation |
| 1001 | -0.066 | **1029** | -0.099 | **1057** | -0.092 | **1085** | 0.850 |
| 1002 | -0.004 | **1030** | -0.468 | **1058** | -0.016 | **1086** | 0.881 |
| 1003 | -0.030 | **1031** | -0.088 | **1059** | 0.135 | **1087** | 0.816 |
| 1004 | -0.127 | **1032** | 0.009 | **1060** | 0.101 | **1088** | 0.936 |
| 1005 | -0.308 | **1033** | 0.010 | **1061** | -0.057 | **1089** | 0.223 |
| 1006 | -0.169 | **1034** | -0.032 | **1062** | -0.007 | **1090** | 0.300 |
| 1007 | 0.038 | **1035** | -0.071 | **1063** | 0.176 | **1091** | 0.342 |
| 1008 | 0.059 | **1036** | -0.075 | **1064** | 0.082 | **1092** | 0.194 |
| 1009 | -0.016 | **1037** | -0.084 | **1065** | 0.197 | **1093** | 0.438 |
| 1010 | 0.016 | **1038** | -0.036 | **1066** | 0.088 | **1094** | 0.190 |
| 1011 | 0.088 | **1039** | 0.014 | **1067** | 0.238 | **1095** | 0.232 |
| 1012 | 0.082 | **1040** | -0.058 | **1068** | 0.110 | **1096** | 0.092 |
| 1013 | 0.456 | **1041** | -0.023 | **1069** | 0.030 | **1097** | 0.987 |
| 1014 | 0.103 | **1042** | -0.178 | **1070** |  | **1098** | 0.973 |
| 1015 | -0.142 | **1043** | -0.104 | **1071** |  | **1099** | 0.262 |
| 1016 | 0.145 | **1044** | -0.042 | **1072** |  | **1100** | 0.479 |
| 1017 | -0.174 | **1045** | -0.058 | **1073** |  | **1101** | 0.465 |
| 1018 | -0.131 | **1046** | -0.156 | **1074** |  | **1102** |  |
| 1019 | -0.089 | **1047** | -0.115 | **1075** |  | **1103** | -0.138 |
| 1020 | -0.063 | **1048** | 0.028 | **1076** |  | **1104** | -0.040 |
| 1021 | -0.182 | **1049** | 0.090 | **1077** |  | **1105** | -0.190 |
| 1022 | -0.209 | **1050** | 0.043 | **1078** |  | **1106** | -0.282 |
| 1023 | -0.239 | **1051** | 0.144 | **1079** |  | **1107** | -0.093 |
| 1024 | -0.058 | **1052** | 0.083 | **1080** |  | **1108** | -0.355 |
| 1025 | -0.357 | **1053** | 0.072 | **1081** |  | **1109** | -0.700 |
| 1026 | -0.009 | **1054** | 0.109 | **1082** | 0.217 | **1110** | 0.068 |
| 1027 | -0.023 | **1055** | 0.131 | **1083** | -0.338 | **1111** | 0.007 |
| 1028 | -0.172 | **1056** | -0.217 | **1084** | 0.860 | **1112** | 0.096 |

Table showing correlation between ‘relative soil moisture’ and ‘field soil moisture’ for each site

From the correlation table, it is seen that most of the sites have very poor correlation, mostly having R2 values of less 0.1. Thus, to bring a good result, just comparing between these two is not sufficient, but it is required to bring other parameters which cause this problem.

It is seen in the literature review that relative soil moisture measured using satellite gets affected by various factors such as the type of crop, crop coverage or crop growth basically how much the crop has covered the field at a given instance and also to the soil class based on texture and type of soil to which it belongs.



Field Soil Moisture vs Relative Soil Moisture Scatter Plot (for all 112 sites)

Now, to understand how this relative soil moisture works along with parameters mainly crop type, crop period and soil class, machine learning algorithms are used. Before moving on to analysis, the need for analysis has to defined and organized in a manner to feed into machine learning algorithms. The compiled data used in these algorithms are derived from different sources and has been compiled into an excel sheet.

The software utilized for extracting and organizing data are (i) Quantum GIS, commonly called as QGIS – an open source software, extended its capability with “PyQGIS”, (ii) Geospatial Data Abstraction Library, in short “GDAL”, for making pixel level calculations on the satellite data obtained and the programming language used is Python – second most popular language after Java.

Crop Type:

The type of crop that is grown in farmer’s field is available along with date. Frequency of data available is approximately twice a month for year 2016 and 2017. Total number of types of crop available from all 112 sites is 51.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Type | No | Type | No | Type |
| 1 | No\_Crop | **18** | Sunflower | **35** | Potato |
| 2 | Maize | **19** | Sugarcane | **36** | Maize+CountryBean |
| 3 | Chickpea | **20** | Garlic | **37** | CountryBean+Maize |
| 4 | Mariegold | **21** | Beetroot+Banana | **38** | Turmeric+Onion |
| 5 | Ragi | **22** | Banana | **39** | Pumkin |
| 6 | Groundnut+Mariegold | **23** | Watermelon+Banana+PolythynPlastic | **40** | HorseGram+CountryBean |
| 7 | Cotton | **24** | Banana+Watermelon | **41** | CountryBean+Sunflower |
| 8 | FieldBean | **25** | Ginger | **42** | Onion+Turmeric+Banana |
| 9 | CountryBean | **26** | Weed | **43** | Watermelon |
| 10 | Sunflower+Maize | **27** | Cabbage | **44** | Banana+Beetroot |
| 11 | Sorghum | **28** | Garlic+Cabbage | **45** | Sunflower+Sorghum |
| 12 | HorseGram | **29** | Beetroot | **46** | Maize+Beetroot |
| 13 | Onion+Turmeric | **30** | Onion | **47** | Onion+Beans |
| 14 | Turmeric | **31** | Maize+Mariegold | **48** | Chilly+Beans |
| 15 | Groundnut | **32** | Toor+Chilly | **49** | Sorghum+Pulses |
| 16 | Beans | **33** | Chilly+Turmeric | **50** | Onion+Toor |
| 17 | Tomato | **34** | Chilly | **51** | Toor |

Since the number of classes are way too many, it is decided to reduce the number of soil classes based on plant leaf coverage area and its height of growth. Keeping this in mind, the plants are divided into 9 classes.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Less coverage | Medium coverage | Dense coverage |
| Short | 1 | 2 | 3 |
| Medium | 4 | 5 | 6 |
| Tall | 7 | 8 | 9 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Crop\_Type | Crop Class | Crop\_Type | Crop Class | Crop\_Type | Crop Class |
| No\_Crop | 1 | Onion | 2 | CountryBean | 6 |
| Weed | 1 | Toor+Chilly | 2 | HorseGram+CountryBean | 6 |
| Pumkin | 1 | Chilly+Turmeric | 2 | CountryBean+Sunflower | 6 |
| Watermelon | 1 | Chilly | 2 | Sunflower+Maize | 8 |
| Ragi | 2 | Potato | 2 | Sorghum | 8 |
| Cotton | 2 | Turmeric+Onion | 2 | Sunflower | 8 |
| FieldBean | 2 | Onion+Beans | 2 | Maize+CountryBean | 8 |
| HorseGram | 2 | Chilly+Beans | 2 | CountryBean+Maize | 8 |
| Onion+Turmeric | 2 | Onion+Toor | 2 | Sunflower+Sorghum | 8 |
| Turmeric | 2 | Toor | 2 | Sorghum+Pulses | 8 |
| Groundnut | 2 | Beetroot | 3 | Sugarcane | 9 |
| Beans | 2 | Mariegold | 4 | Beetroot+Banana | 9 |
| Tomato | 2 | Maize+Mariegold | 4 | Banana | 9 |
| Garlic | 2 | Maize | 5 | Watermelon+Banana+PolythynPlastic | 9 |
| Ginger | 2 | Groundnut+Mariegold | 5 | Banana+Watermelon | 9 |
| Cabbage | 2 | Maize+Beetroot | 5 | Onion+Turmeric+Banana | 9 |
| Garlic+Cabbage | 2 | Chickpea | 6 | Banana+Beetroot | 9 |

Crop period:

Crop period is computed using python code snippets. Here, the crop period is defined in percentage that when crop is sown, it is considered as 0% and during the harvest, it is considered as 100% and interpolated linearly in between days. The code snippet used is as follows:

Code Snippet

from openpyxl import load\_workbook

import numpy as np

basepath = 'D:\MASTERS-PROJECT\MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = basepath + 'Soil\_Moisture\_Berambadi\_1617.xlsx'

out\_file = basepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_out.xlsx'

wb = load\_workbook(in\_file)

sheet = wb['SM\_ML\_values']

Crop\_Type = list(sheet.columns)[5]

crop\_array\_count = 0

instance = 0

appending = 0

crops\_array = []

previous\_crop = ''

crop\_array\_temp = []

for crop in Crop\_Type:

row\_num = crop.row

if(row\_num > 1 and crop.value is not None):

# print (crop.value)

current\_crop = crop.value

if (crop.value.strip() != 'No\_Crop'):

if(previous\_crop != current\_crop):

if(previous\_crop != 'No\_Crop'):

crops\_array.append(crop\_array\_temp)

crop\_array\_temp = []

instance = 0

appending = 1

instance += 1

crop\_array\_temp.append([crop.row, instance])

if ((appending and crop.value.strip() == 'No\_Crop')):

crops\_array.append(crop\_array\_temp)

crop\_array\_temp = []

appending = 0

instance = 0

if (row\_num > 5265):

break

previous\_crop = crop.value

for crop\_set in crops\_array:

if (len(crop\_set) != 0):

max\_instance = int(crop\_set[len(crop\_set)-1][1])

for row in crop\_set:

sheet.cell(row=row[0], column=8).value = row[1]/max\_instance

wb.save(out\_file)

print ("done totally")

Relative Soil Moisture:

The relative soil moisture is obtained with respect to GPS coordinates of sites from the moisture data files which are present in ‘GeoTiff’ format. The tool used for extracting relative soil moisture from ‘GeoTiff’ files is ‘Geospatial Data Abstraction Library’ available in the PyQGIS console. The code snippet used is as follows:

Code Snippet

import pickle

import numpy as np

import pandas as pd

import os, os.path

from datetime import datetime, timedelta

from pandas import ExcelWriter

from openpyxl import load\_workbook

from osgeo import gdal, osr

from qgis.core import QgsRasterLayer

basepath = 'D:\MASTERS-PROJECT\\'

subBasepath = basepath + 'MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_out.xlsx'

out\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_out.xlsx'

sm\_folder = 'D:\MASTERS-PROJECT\sm\_VATI\\'

SM\_ML\_df = pd.read\_excel(in\_file, sheet\_name='SM\_ML\_values')

SM\_ML\_Dates\_df = SM\_ML\_df[['Date']]

SM\_ML\_Dates\_df.drop\_duplicates(subset=['Date'], keep='first', inplace=True)

SM\_ML\_Dates\_df.reset\_index(drop=True, inplace=True)

# print (SM\_ML\_Dates\_df)

sm\_VATI\_LatLon\_DF = SM\_ML\_df[['Plot\_Id', 'Latitude', 'Longitude']]

sm\_VATI\_LatLon\_DF.drop\_duplicates(subset=['Plot\_Id'], keep='first', inplace=True)

sm\_VATI\_LatLon\_DF.reset\_index(drop=True, inplace=True)

for index, SM\_ML\_Dates\_df\_row in SM\_ML\_Dates\_df.iterrows():

date\_dfval = SM\_ML\_Dates\_df\_row['Date']

date = str(date\_dfval)

date = datetime.strptime(date, '%Y-%m-%d %H:%M:%S')

date = date.strftime('%Y%m%d')

file\_fullPath = sm\_folder + date + '.tif'

if(os.path.isfile(file\_fullPath)):

# print (date)

raster\_ds = gdal.Open(file\_fullPath)

rlayer = QgsRasterLayer(file\_fullPath)

geoTrans = raster\_ds.GetGeoTransform()

ulX = geoTrans[0]

ulY = geoTrans[3]

pxUnitX = geoTrans[1]

pxUnitY = geoTrans[5] # this value is -ve

prj = raster\_ds.GetProjectionRef()

srs = osr.SpatialReference(wkt=prj)

EPSG\_num = srs.GetAttrValue('authority', 1)

crsSrc = QgsCoordinateReferenceSystem(4326) # WGS 84

crsDest = QgsCoordinateReferenceSystem(int(EPSG\_num)) # WGS 84 / UTM zone 43N

xform = QgsCoordinateTransform(crsSrc, crsDest)

sm\_VATI\_nparray = np.array([])

for index, sm\_VATI\_LatLon\_DF\_row in sm\_VATI\_LatLon\_DF.iterrows():

pt\_gps = QgsPoint(float(sm\_VATI\_LatLon\_DF\_row['Longitude']), float(sm\_VATI\_LatLon\_DF\_row['Latitude']))

pt\_meter = xform.transform(pt\_gps)

# print (pt\_meter)

sm = rlayer.dataProvider().identify(pt\_meter, QgsRaster.IdentifyFormatValue)

sm\_val = sm.results()[1]

# print (sm\_val)

sm\_VATI\_nparray = np.append(sm\_VATI\_nparray, sm\_val)

sm\_VATI\_LatLon\_DF['sm\_VATI'] = sm\_VATI\_nparray

# print (sm\_VATI\_LatLon\_DF.head())

for index, sm\_VATI\_LatLon\_DF\_row in sm\_VATI\_LatLon\_DF.iterrows():

Plot\_Id\_val = sm\_VATI\_LatLon\_DF\_row['Plot\_Id']

SM\_ML\_df.loc[(SM\_ML\_df['Plot\_Id'] == Plot\_Id\_val) & (SM\_ML\_df['Date'] == date\_dfval), 'sm\_VATI'] = sm\_VATI\_LatLon\_DF\_row['sm\_VATI']

writer = ExcelWriter(out\_file)

SM\_ML\_df.to\_excel(writer,'SM\_ML\_values')

writer.save()

print ("done totally")

Maximum relative soil moisture:

As saturation capacity is a property of soil at a site, maximum relative soil moisture can be defined as one property of site where at a site, the value of relative soil moisture in any time does not exceed the maximum saturation capacity. Code Snippet used is as follows:

Code Snippet

import numpy as np

import pandas as pd

import os

from datetime import datetime, timedelta

from osgeo import gdal, osr

from qgis.core import QgsRasterLayer

from pandas import ExcelWriter

basepath = 'D:\MASTERS-PROJECT\\'

subBasepath = basepath + 'MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_out.xlsx'

out\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_withMinMaxRSM\_out.xlsx'

sm\_folder = 'D:\MASTERS-PROJECT\sm\_VATI\\'

SM\_ML\_df = pd.read\_excel(in\_file, sheet\_name='SM\_ML\_values')

RSM\_minmax\_DF = SM\_ML\_df[['Plot\_Id', 'Latitude', 'Longitude']]

RSM\_minmax\_DF.drop\_duplicates(subset=['Plot\_Id'], keep='first', inplace=True)

RSM\_minmax\_DF.reset\_index(drop=True, inplace=True)

# print (RSM\_max\_DF.shape[0])

rsm\_Max\_npArray = np.zeros(RSM\_minmax\_DF.shape[0])

rsm\_Min\_npArray = np.zeros(RSM\_minmax\_DF.shape[0])

folder = os.listdir(sm\_folder)

fldr\_len = len(folder)

for ij, each\_file in enumerate(folder):

if (ij % 30 ==0):

print (ij, "completed")

if '.tif' in each\_file and '.xml' not in each\_file:

file\_fullPath = sm\_folder + each\_file

# print (file\_fullPath)

for index, row in RSM\_minmax\_DF.iterrows():

# print (index)

rlayer\_rsm = QgsRasterLayer(file\_fullPath)

rsm\_ds = gdal.Open(file\_fullPath)

prj = rsm\_ds.GetProjectionRef()

srs = osr.SpatialReference(wkt=prj)

EPSG\_num = srs.GetAttrValue('authority', 1)

crsSrc = QgsCoordinateReferenceSystem(4326) # WGS 84

crsDest = QgsCoordinateReferenceSystem(int(EPSG\_num)) # WGS 84 / UTM zone 43N

xform = QgsCoordinateTransform(crsSrc, crsDest)

pt\_gps = QgsPoint(float(row['Longitude']), float(row['Latitude']))

pt\_meter = xform.transform(pt\_gps)

RSM = rlayer\_rsm.dataProvider().identify(pt\_meter, QgsRaster.IdentifyFormatValue)

if (RSM.isValid()):

if not RSM.results()[1] == None:

rsm\_val = RSM.results()[1]

if(rsm\_Max\_npArray[index] < rsm\_val):

rsm\_Max\_npArray[index] = rsm\_val

if(rsm\_Min\_npArray[index] > rsm\_val):

rsm\_Min\_npArray[index] = rsm\_val

RSM\_minmax\_DF['min\_RSM'] = rsm\_Min\_npArray

RSM\_minmax\_DF['max\_RSM'] = rsm\_Max\_npArray

SM\_ML\_df['min\_RSM'] = SM\_ML\_df['Plot\_Id'].map(RSM\_minmax\_DF.set\_index('Plot\_Id')['min\_RSM'])

SM\_ML\_df['max\_RSM'] = SM\_ML\_df['Plot\_Id'].map(RSM\_minmax\_DF.set\_index('Plot\_Id')['max\_RSM'])

print ("writing to excel")

writer = ExcelWriter(out\_file)

SM\_ML\_df.to\_excel(writer,'SM\_ML\_values')

writer.save()

print ("done totally")

Soil Class:

The soil classes are available as vector file, ie., dbase file. Hence, the easiest approach that is followed is to get the soil class of all sites based on nearest neighbor algorithm. The major classes of the soil has been defined in terms of locations and the description and soil type is mentioned according. To feed into machine learning algorithms, The soil types are mapped to integer types accordingly.

|  |  |
| --- | --- |
| Soil\_Type | Int\_Type |
| Clay | 1 |
| Gravelly\_Sandy\_Clay | 2 |
| Gravelly\_Sandy\_Clay\_Loam | 3 |
| Sandy\_Clay | 4 |
| Sandy\_Clay\_Loam | 5 |

Table showing soil types and corresponding mapped integer types for ML algorithms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | SoilName | Description | Soil\_Type | Int\_Type |
| ARK | Annurkeri soils | well drained, have dark reddish brown to very dusky red sandy clay to clay soils | Sandy\_Clay | 4 |
| BMB | Beemanabeedu soils | moderately well drained, have very dark greyish brown to dark grey and very dark brown clayey soils | Clay | 1 |
| BMD | Berambadi soils | well drained, dark brown to dark greyish brown clayey soils | Clay | 1 |
| BRG | Bargi soils | well drained, have very dark brown to very dark grayish brown clay soils | Clay | 1 |
| DRH | Devarahalli soils | well drained, have dark red to reddish brown and dusky red gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| GPR | Gopalapura soils | well drained, have dark brown to dark reddish brown and reddish brown gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| HDR | Hindupur soils | well drained, have dark reddish brown to dusky red sandy clay loam to sandy clay soils | Sandy\_Clay\_Loam | 5 |
| HGH | Honnegaudanahalli soils | well drained, have very dark brown to brown and dark reddish brown sandy clay loam soils | Sandy\_Clay\_Loam | 5 |
| HPR | Hullipura soils | well drained, have dark brown to very dark brown gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| KDH | Kalligaudanahalli soils | well drained, have dark red to dark reddish brown and dark brown sandy clay to clay soils | Sandy\_Clay | 4 |
| KLP | Kallipura soils | well drained, have dark reddish brown to dark red gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| KNG | Kannigala soils | well drained, have dark reddish brown to dark red gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| MDH | Maddinahundi soils | well drained, have dark reddish brown gravelly sandy clay soils | Gravelly\_Sandy\_Clay | 2 |
| MGH | Magoonahalli soils | well drained, have very dark brown to dark brown gravelly sandy clay loam soils | Gravelly\_Sandy\_Clay\_Loam | 3 |
| SPR | Shivapura soils | well drained, have dark reddish brown gravelly sandy clay loam to sandy clay soils | Gravelly\_Sandy\_Clay\_Loam | 3 |

Table showing classes and corresponding classification and description

The code snippet used for extracting soil class is as follows:

Code Snippet

import numpy as np

import pandas as pd

from pandas import ExcelWriter

basepath = 'D:\MASTERS-PROJECT\\'

subBasepath = basepath + 'MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_withMinMaxRSM\_out.xlsx'

out\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_withMinMaxRSM\_SoilClass\_out.xlsx'

soil\_class\_file = subBasepath + 'Soil\_Classes.xlsx'

soil\_class\_df = pd.read\_excel(soil\_class\_file, sheet\_name='soil\_classes')

soil\_class\_df = soil\_class\_df[['longitude','latitude','Int\_Type']]

# print (soil\_class\_df)

SM\_ML\_df = pd.read\_excel(in\_file, sheet\_name='SM\_ML\_values')

sm\_Soil\_Class\_LatLon\_DF = SM\_ML\_df[['Plot\_Id', 'Latitude', 'Longitude']]

sm\_Soil\_Class\_LatLon\_DF = sm\_Soil\_Class\_LatLon\_DF.drop\_duplicates(subset=['Plot\_Id'], keep='first')

sm\_Soil\_Class\_LatLon\_DF.reset\_index(drop=True, inplace=True)

# print (sm\_Soil\_Class\_LatLon\_DF.shape)

soil\_found\_class\_nparray = np.empty(shape=sm\_Soil\_Class\_LatLon\_DF.shape[0])

soil\_found\_class\_nparray[:] = np.nan

# print (soil\_found\_class\_nparray.shape)

count = 1

for i, SM\_row in sm\_Soil\_Class\_LatLon\_DF.iterrows():

sm\_longitude = SM\_row['Longitude']

sm\_latitude = SM\_row['Latitude']

left\_extent\_arry = np.array([])

right\_extent\_arry = np.array([])

for j, Class\_row in soil\_class\_df.iterrows():

if sm\_longitude >= Class\_row['longitude']:

left\_extent\_arry = np.append(left\_extent\_arry, j)

if sm\_longitude <= Class\_row['longitude']:

right\_extent\_arry = np.append(right\_extent\_arry, j)

if (left\_extent\_arry.shape[0] != 0):

left\_extent\_j = left\_extent\_arry.max()

else:

left\_extent\_j = None

if (right\_extent\_arry.shape[0] != 0):

right\_extent\_j = right\_extent\_arry.min()

else:

right\_extent\_j = None

# print (sm\_longitude, left\_extent\_j, right\_extent\_j)

if (left\_extent\_j != None and right\_extent\_j != None):

top\_extent\_arry = np.array([])

bottom\_extent\_arry = np.array([])

for jk, Class\_row in soil\_class\_df.iterrows():

if (right\_extent\_j >= jk >= left\_extent\_j):

if sm\_latitude >= Class\_row['latitude']:

bottom\_extent\_arry = np.append(bottom\_extent\_arry, jk)

if sm\_latitude <= Class\_row['latitude']:

top\_extent\_arry = np.append(top\_extent\_arry, jk)

if (top\_extent\_arry.shape[0] != 0):

LT\_extent\_jk = top\_extent\_arry.min()

else:

LT\_extent\_jk = None

if (bottom\_extent\_arry.shape[0] != 0):

RT\_extent\_jk = bottom\_extent\_arry.max()

else:

RT\_extent\_jk = None

# taking left top value for soil

if (LT\_extent\_jk != None and RT\_extent\_jk != None):

soil\_found\_class\_nparray[i] = int(soil\_class\_df['Int\_Type'].iloc[int(LT\_extent\_jk)])

else:

soil\_found\_class\_nparray[i] = np.nan

sm\_Soil\_Class\_LatLon\_DF['Int\_Type'] = soil\_found\_class\_nparray

SM\_ML\_df['Soil\_Class'] = SM\_ML\_df['Plot\_Id'].map(sm\_Soil\_Class\_LatLon\_DF.set\_index('Plot\_Id')['Int\_Type'])

print ("writing to excel")

writer = ExcelWriter(out\_file)

SM\_ML\_df.to\_excel(writer,'SM\_ML\_values')

writer.save()

print ("done totally")

Thus, the different features that can be used in machine learning are (i) crop type, (ii) crop period, (iii) maximum relative soil moisture and (iv) soil class.

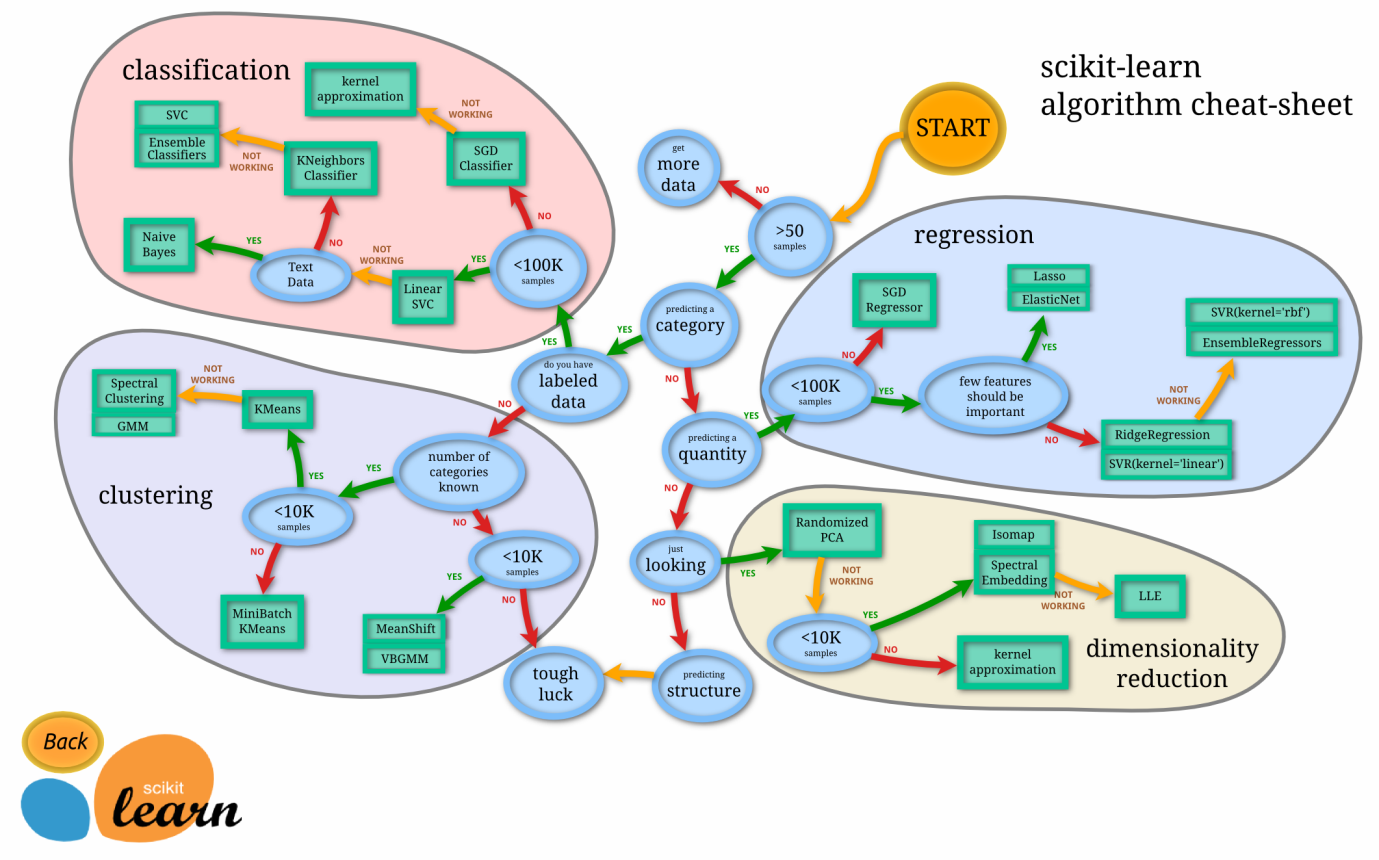
**Machine Learning Methodology:**

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.

We can separate learning problems in a few large categories:

1. supervised learning, in which the data comes with additional attributes that is to be predicted. This problem can be either:
   1. classification: samples belong to two or more classes and learning is done from already labeled data how to predict the class of unlabeled data. To think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided, it is to tried to label them with the correct category or class.
   2. regression: if the desired output consists of one or more continuous variables, then the task is called regression.
2. unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values. The goal in such problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

Hence, the problem studied has name features and we want to classify them into classes. Thus, the nature of problem we have falls into supervised learning and it is a classification problem.



Scikit-Learn:

Scikit-Learn is a famous machine learning module available in python programming language. Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

There are many classifiers available in scikit. But not all classifiers work in all the problem cases. Hence, after analyzing with most of them, it is decided to use three classifiers which works well with our data. They are basically ensemble classifiers – (i) Bagging classifier, (ii) ExtraTrees classifier, (iii) GradientBoosting classifier.

Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

This algorithm encompasses several works from the literature. When random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting. If samples are drawn with replacement, then the method is known as Bagging. When random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces [R156]. Finally, when base estimators are built on subsets of both samples and features, then the method is known as Random Patches. The class and its arguments for bagging classifier are:

class sklearn.ensemble.BaggingClassifier(base\_estimator=None, n\_estimators=10, max\_samples=1.0, max\_features=1.0, bootstrap=True, bootstrap\_features=False, oob\_score=False, warm\_start=False, n\_jobs=1, random\_state=None, verbose=0)

ExtraTrees Classifier:

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The class and its arguments for ExtraTrees classifier are:

class sklearn.ensemble.ExtraTreesClassifier(n\_estimators=10, criterion=’gini’, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_features=’auto’, max\_leaf\_nodes=None, min\_impurity\_decrease=0.0, min\_impurity\_split=None, bootstrap=False, oob\_score=False, n\_jobs=1, random\_state=None, verbose=0, warm\_start=False, class\_weight=None)

GradientBoosting Classifier:

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage n\_classes\_ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced. The class and its arguments for GradientBoosting classifier are:

class sklearn.ensemble.GradientBoostingClassifier(loss=’deviance’, learning\_rate=0.1, n\_estimators=100, subsample=1.0, criterion=’friedman\_mse’, min\_samples\_split=2, min\_samples\_leaf=1, min\_weight\_fraction\_leaf=0.0, max\_depth=3, min\_impurity\_decrease=0.0, min\_impurity\_split=None, init=None, random\_state=None, max\_features=None, verbose=0, max\_leaf\_nodes=None, warm\_start=False, presort=’auto’)

Using these classifiers, two types of problem are tried:

1. Finding Crop Type Class

In finding crop type class problem, the mandatory features are Relative Soil Moisture and Field Soil Moisture and the Crop Type is kept as ‘to be predicted’ feature. Thus, other features namely crop period, maximum relative soil moisture and soil class are kept as optional features, and hence they are fed into algorithms as combinations.

1. Finding Field Soil Moisture Class

In finding field soil moisture class problem, the mandatory feature is Relative Soil Moisture only and Field Soil Moisture is kept as ‘to be predicted’ feature. Thus, other features namely crop type, crop period, maximum relative soil moisture and soil class are kept as optional features, and hence they are fed into algorithms as combinations.

Code Snippet - Finding Crop Type Class

import pickle

import numpy as np

import pandas as pd

from pandas import ExcelWriter

import matplotlib.pyplot as plt

import itertools

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import BaggingClassifier,ExtraTreesClassifier,GradientBoostingClassifier

basepath = 'D:\MASTERS-PROJECT\\'

subBasepath = basepath + 'MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_withMinMaxRSM\_SoilClass\_out.xlsx'

# Reading from Excel file

SM\_ML\_df = pd.read\_excel(in\_file, sheet\_name='SM\_ML\_values')

SM\_ML\_df.replace(0, np.nan, inplace=True)

SM\_ML\_df.replace(-np.inf, np.nan, inplace=True)

# dropping rows with no crop type

SM\_ML\_df = SM\_ML\_df.dropna(axis=0, subset=['Crop\_Type'])

# Forming Dictionary for crop Type and replacing with integer in Crop Type Column

Crop\_Type\_nparray = SM\_ML\_df['Crop\_Type'].str.strip().unique()

int\_toCrop\_type\_dict = dict(enumerate(Crop\_Type\_nparray))

crop\_type\_toInt\_dict = {y:x for x,y in int\_toCrop\_type\_dict.items()}

SM\_ML\_df["Crop\_Type"] = SM\_ML\_df["Crop\_Type"].str.strip().map(crop\_type\_toInt\_dict)

# column\_optional\_list = ['Plot\_Id','max\_RSM','Crop\_Period','Crop\_Type','Soil\_Class']

column\_optional\_list = ['max\_RSM','Crop\_Period','Soil\_Class']

# column\_optional\_list = ['max\_RSM']

column\_mandatory\_list = ['rsm\_VATI','Field\_SM','Crop\_Type']

combinations = []

combinations.append([[]])

for i in range(1, len(column\_optional\_list)+1):

els = [list(x) for x in itertools.combinations(column\_optional\_list, i)]

combinations.append(els)

# Re-Setting Combination Manually

# combinations = []

# combinations.append([['max\_RSM','Crop\_Period','Soil\_Class']])

# print (combinations)

writer = ExcelWriter(subBasepath + '11\_performing\_Analsis\_getting\_Crop\_Type.xlsx', engine='xlsxwriter')

index = ['Class'+str(i) for i in range(1, len(Crop\_Type\_nparray)+1)]

Crop\_Type\_DF = pd.DataFrame(Crop\_Type\_nparray, index=index)

Crop\_Type\_DF.to\_excel(writer,"Classes",startrow=2, startcol=1)

sheet = writer.sheets["Classes"]

sheet.write('A1', "Classification Algorithm - Comparing Field Soil Moisture and Relative Soil Moisture")

sheet.write('C3', "All Classes")

combo\_list = []

# class\_groups\_df = pd.DataFrame()

comb\_count = 1

for temp1 in combinations:

for comb in temp1:

columns\_loop = list(column\_mandatory\_list)

sheetname = 'Combo\_' + str(comb\_count)

comb\_count += 1

columns\_loop.extend(comb)

print (sheetname, columns\_loop)

Field\_SM\_VATI = SM\_ML\_df[columns\_loop]

Field\_SM\_VATI = Field\_SM\_VATI.dropna(axis=0, how='any')

X = Field\_SM\_VATI.drop(['Crop\_Type'], axis=1).values

y = Field\_SM\_VATI['Crop\_Type'].values

# print (X.shape, y.shape)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1)

clf = []

clf.extend([AdaBoostClassifier(),BaggingClassifier(),ExtraTreesClassifier(),GradientBoostingClassifier(),RandomForestClassifier()])

scores = []

for i, each\_clf in enumerate(clf):

each\_clf.fit(X\_train, y\_train)

scores.append(each\_clf.score(X\_test, y\_test))

scores\_np = np.array(scores)

clf\_np = np.array(clf)

# print (scores\_np)

clf\_best = clf\_np[np.argmax(scores\_np)]

# print("Max Accuracy: ", scores\_np.max())

# print(clf\_best)

category = clf\_best.predict(X\_test)

result\_DF = pd.DataFrame()

result\_DF['Crop\_Type'] = y\_test

result\_DF['Class'] = category

result\_DF['Class\_int'] = category

result\_DF["Crop\_Type"] = result\_DF["Crop\_Type"].map(int\_toCrop\_type\_dict)

result\_DF["Class"] = result\_DF["Class"].map(int\_toCrop\_type\_dict)

# frames = [class\_groups\_df, result\_DF]

# class\_groups\_df = pd.concat(frames, ignore\_index=True)

result\_DF.to\_excel(writer,sheetname)

sheet = writer.sheets[sheetname]

sheet.write('F2', "Combinations:")

sheet.write('F3', "Best Classifying Algorithm:")

sheet.write('F4', "Parameters:")

sheet.write('F5', "Max Accuracy:")

# sheet.write('I2', str(column\_mandatory\_list) + " with " + str(comb))

sheet.write('I2', str(columns\_loop))

sheet.write('I3', str(clf\_best))

sheet.write('I4', str(clf\_best.get\_params))

sheet.write('I5', scores\_np.max())

combo\_list.append([sheetname, scores\_np.max(), str(columns\_loop), str(clf\_best)])

index = ['Sheet'+str(i+1) for i in range(1, len(combo\_list)+1)]

combo\_nparray\_DF = pd.DataFrame(combo\_list, index=index, columns=['Combo\_No', 'R-Squared','Combination','Best Classifying Algorithm'])

combo\_nparray\_DF.sort\_values(' R-Squared ', ascending=False, inplace=True)

combo\_nparray\_DF.to\_excel(writer,"Classes",startrow=2, startcol=5)

writer.save()

Code Snippet - Finding Field Soil Moisture Class

import pickle

import numpy as np

import pandas as pd

from pandas import ExcelWriter

import matplotlib.pyplot as plt

import itertools

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble importBaggingClassifier,ExtraTreesClassifier,GradientBoostingClassifier

basepath = 'D:\MASTERS-PROJECT\\'

subBasepath = basepath + 'MTECH-PROJ-Prob\_1\_VATI\_vs\_Field\_Moisture\\'

in\_file = subBasepath + 'Soil\_Moisture\_Berambadi\_1617\_CropAge\_VATI\_withMinMaxRSM\_SoilClass\_out.xlsx'

# Reading from Excel file

SM\_ML\_df = pd.read\_excel(in\_file, sheet\_name='SM\_ML\_values')

SM\_ML\_df.replace(0, np.nan, inplace=True)

SM\_ML\_df.replace(-np.inf, np.nan, inplace=True)

# Replacing with crop classes

crop\_class\_file = subBasepath + 'Crop\_Classes.xlsx'

crop\_class\_df = pd.read\_excel(crop\_class\_file, sheet\_name='crop\_classes')

# print (crop\_class\_df)

replacing\_dict = {}

for index, each\_row in crop\_class\_df.iterrows():

# print (each\_row, type(each\_row))

replacing\_dict[each\_row['Crop\_Type']] = 'class' + str(each\_row['Crop\_Class'])

# print (replacing\_dict)

SM\_ML\_df["Crop\_Type"] = SM\_ML\_df["Crop\_Type"].str.strip().replace(replacing\_dict)

# print (SM\_ML\_df)

# dropping rows with no crop type

SM\_ML\_df = SM\_ML\_df.dropna(axis=0, subset=['Crop\_Type'])

# Forming Dictionary for crop Type and replacing with integer in Crop Type Column

Crop\_Type\_nparray = SM\_ML\_df['Crop\_Type'].str.strip().unique()

# print (Crop\_Type\_nparray)

int\_toCrop\_type\_dict = dict(enumerate(Crop\_Type\_nparray))

crop\_type\_toInt\_dict = {y:x for x,y in int\_toCrop\_type\_dict.items()}

# print (int\_toCrop\_type\_dict)

SM\_ML\_df["Crop\_Type"] = SM\_ML\_df["Crop\_Type"].str.strip().map(crop\_type\_toInt\_dict)

SM\_ML\_df['Mapped\_Field\_SM'] = SM\_ML\_df['Field\_SM']

# SM\_ML\_df = SM\_ML\_df.dropna(axis=0, how='any')

# Mapping Function for Field Soil Moisture into Integer Class

max\_val = 60

interval = 3

no\_of\_intervals = int(max\_val/interval)

def func(x):

if x <= max\_val:

return int(x/interval)

elif x > max\_val:

return no\_of\_intervals

else:

return np.nan

int\_Class\_toFieldMoistureClass\_dict = {}

last\_key = 0

for i in range(no\_of\_intervals):

int\_Class\_toFieldMoistureClass\_dict[i] = str(i\*interval) + 't' + str((i+1)\*interval)

last\_key = i

int\_Class\_toFieldMoistureClass\_dict[last\_key+1] = 'gt60'

int\_Class\_toFieldMoistureClass\_dict[last\_key+2] = 'NA'

# print (int\_Class\_toFieldMoistureClass\_dict)

SM\_ML\_df['Mapped\_Field\_SM'] = SM\_ML\_df['Mapped\_Field\_SM'].map(func)

# print (SM\_ML\_df.head())

# column\_optional\_list = ['Plot\_Id','max\_RSM','Crop\_Period','Crop\_Type','Soil\_Class']

column\_optional\_list = ['max\_RSM','Crop\_Period','Crop\_Type','Soil\_Class']

# column\_optional\_list = ['max\_RSM']

column\_mandatory\_list = ['rsm\_VATI','Field\_SM','Mapped\_Field\_SM']

combinations = []

combinations.append([[]])

for i in range(1, len(column\_optional\_list)+1):

els = [list(x) for x in itertools.combinations(column\_optional\_list, i)]

combinations.append(els)

# Re-Setting Combination Manually

# combinations = []

# combinations.append([['max\_RSM','Crop\_Type','Crop\_Period','Soil\_Class']])

writer = ExcelWriter(subBasepath + '11\_performing\_Analsis\_getting\_Field\_Soil\_Moisture\_Class.xlsx', engine='xlsxwriter')

# print (int\_Class\_toFieldMoistureClass\_dict)

Field\_Moisture\_Class\_DF = pd.DataFrame.from\_dict(int\_Class\_toFieldMoistureClass\_dict, orient='index')

Field\_Moisture\_Class\_DF.to\_excel(writer,"Classes",startrow=2,startcol=1)

sheet = writer.sheets["Classes"]

sheet.write('A1', "Classification Algorithm - Finding Field Soil Moisture Class from Relative Soil Moisture (from SWIR bands)")

sheet.write('C3', "All Classes")

combo\_list = []

class\_groups\_df = pd.DataFrame()

comb\_count = 1

for temp1 in combinations:

for comb in temp1:

columns\_loop = list(column\_mandatory\_list)

sheetname = 'Combo\_' + str(comb\_count)

columns\_loop.extend(comb)

print (sheetname, columns\_loop)

Field\_SM\_VATI = SM\_ML\_df[columns\_loop]

Field\_SM\_VATI = Field\_SM\_VATI.dropna(axis=0, how='any')

X = Field\_SM\_VATI.copy()

y = Field\_SM\_VATI['Mapped\_Field\_SM']

# print (X.shape, y.shape)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1)

X\_test\_Field\_SM = X\_test['Field\_SM']

X\_train = X\_train.drop(['Field\_SM','Mapped\_Field\_SM'], axis=1)

X\_test = X\_test.drop(['Field\_SM','Mapped\_Field\_SM'], axis=1)

# print (X\_train)

# break

# break

X\_train = X\_train.values

y\_train = y\_train.values

X\_test = X\_test.values

clf = []

clf.extend([BaggingClassifier(),ExtraTreesClassifier(),GradientBoostingClassifier()])

scores = []

for i, each\_clf in enumerate(clf):

each\_clf.fit(X\_train, y\_train)

scores.append(each\_clf.score(X\_test, y\_test))

# print (scores)

scores\_np = np.array(scores)

clf\_np = np.array(clf)

# print (scores\_np)

clf\_best = clf\_np[np.argmax(scores\_np)]

# print("Max Accuracy: ", scores\_np.max())

# print(clf\_best)

category = clf\_best.predict(X\_test)

result\_DF = pd.DataFrame()

result\_DF['Field\_SM'] = X\_test\_Field\_SM

result\_DF['Field\_SM'] = SM\_ML\_df['Field\_SM'].map(func)

result\_DF['Field\_SM'] = result\_DF['Field\_SM'].map(int\_Class\_toFieldMoistureClass\_dict)

category = np.rint(category)

# print (category,type(category))

result\_DF['SM\_Class'] = category

result\_DF['Class\_int'] = category

result\_DF['SM\_Class'] = result\_DF['SM\_Class'].map(int\_Class\_toFieldMoistureClass\_dict)

result\_DF.to\_excel(writer,sheetname)

sheet = writer.sheets[sheetname]

sheet.write('F2', "Combinations:")

sheet.write('F3', "Best Classifying Algorithm:")

sheet.write('F4', "Parameters:")

sheet.write('F5', "Max Accuracy:")

sheet.write('I2', str(column\_mandatory\_list) + " with " + str(comb))

sheet.write('I2', str(columns\_loop))

sheet.write('I3', str(clf\_best))

sheet.write('I4', str(clf\_best.get\_params))

sheet.write('I5', scores\_np.max())

combo\_list.append([sheetname, scores\_np.max(), str(columns\_loop), str(clf\_best)])

comb\_count += 1

# if (comb\_count > 1):

# break

# break

index = ['Sheet'+str(i+1) for i in range(1, len(combo\_list)+1)]

combo\_nparray\_DF = pd.DataFrame(combo\_list, index=index, columns=['Combo\_No', ' R-Squared ','Combination','Best Classifying Algorithm'])

combo\_nparray\_DF.sort\_values(' R-Squared ', ascending=False, inplace=True)

combo\_nparray\_DF.to\_excel(writer,"Classes",startrow=2, startcol=5)

writer.save()

print ("done totally")

In all the two problems, 92 sites that fall on the Berambadi region of Gundlupet Taluk have been used for training the machine learning algorithms and 20 sites that fall in Vaddagare region of Koratagere Taluk have been kept as test datasets for computing testing purpose.

Pickle:

One awesome tool available in python is ‘pickle’. As soon as the classifier is trained with the training dataset, the classifier variable can be dumped into local storage for further use and hence reduces the computation time. The dumped file can be loaded again easily and can be used for prediction without the need for training datasets into picture. The Code Snippet for using pickle is as follows:

Code Snippet

# dump classifier to a file

clf = [BaggingClassifier()]

clf.fit(X\_train, y\_train)

file = open('trained\_classifier.pickle', 'wb')

pickle.dump(clf, file)

file.close()

# for retrieving back the trained classifer

file = open('trained\_classifier.pickle', 'rb')

Clf = pickle.load(file)

file.close()

**Chapter 4**

**Results and Discussions**

The code is programmed in a way to automatically use the best classifier based on accuracy for each classifier. And also, the best classifying algorithm for each combination is identified and sorted out as a table based on accuracy obtained from the test datasets.

Problem 1: Finding Crop Type Class

|  |  |  |
| --- | --- | --- |
| Combination | R-Squared | Best Classifying Algorithm |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'Crop\_Period', 'Soil\_Class'] | 0.565217 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'max\_RSM', 'Soil\_Class'] | 0.486842 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'max\_RSM'] | 0.469799 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'Soil\_Class'] | 0.447368 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'max\_RSM', 'Crop\_Period', 'Soil\_Class'] | 0.434783 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type'] | 0.389262 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'max\_RSM', 'Crop\_Period'] | 0.376471 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'Crop\_Period'] | 0.294118 | ExtraTreesClassifier |

In this problem, the combinations column shows the features that has been used in performing the classification in machine learning. The features that helped classify this problem best are crop period and soil class. The best classifying algorithm that has been found out is ExtraTrees Classifier. It could seen that after using machine learning algorithms the R-squared value which was earlier less than 0.1 has been improved to more than or equal to 0.56.

Problem 2: Finding Field Soil Moisture Class

|  |  |  |
| --- | --- | --- |
| Combination | R-Squared | Best Classifying Algorithm |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Soil\_Class'] | 0.381579 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', | 0.369128 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type', 'Soil\_Class'] | 0.368421 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Soil\_Class'] | 0.355263 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM'] | 0.348993 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Period', 'Soil\_Class'] | 0.326087 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Period', 'Crop\_Type', 'Soil\_Class'] | 0.326087 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Period'] | 0.317647 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Type', 'Soil\_Class'] | 0.315789 | BaggingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Type'] | 0.288591 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Period'] | 0.282353 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Period', 'Crop\_Type'] | 0.282353 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'Crop\_Period', 'Crop\_Type'] | 0.270588 | GradientBoostingClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Period', 'Soil\_Class'] | 0.26087 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Type'] | 0.255034 | ExtraTreesClassifier |
| ['rsm\_VATI', 'Field\_SM', 'max\_RSM', 'Crop\_Period', 'Crop\_Type', 'Soil\_Class'] | 0.23913 | GradientBoostingClassifier |

In this problem, the combinations column shows the features that has been used in performing the classification in machine learning. The features that helped classify this problem best are maximum relative soil moisture and soil class. The best classifying algorithm that has been found out is Bagging Classifier. It could seen that after using machine learning algorithms the R-squared value which was earlier less than 0.1 has been improved to more than or equal to 0.38.

**Chapter 5**

**Conclusions and Future Scope of Study**

1. Thus we are able achieve good correlation between relative soil moisture and field soil moisture to a greater extent which did not make any sense before machine learning algorithms were applied and the classifier could be used for prediction of future values of field moisture content from satellite data.
2. Using machine learning, we could extract many interesting features from the datasets rather than just correlating. Thus, problem 1 – identifying the crop type from the variances shown between relative soil moisture and field soil moisture.

Future Scope:

1. Though we have compared values corresponding to GPS coordinates accurately, still the field moisture datasets which we used are points and the relative soil moisture is averaged over a larger grid. An attempt can be made to cluster two field moisture data points in a single grid as an average or some mapping can be done and check if the accuracy is improving. More accuracy could be obtained if the resolution of the satellite products is improved.
2. Still the problem can be better understood using deep learning algorithms (a subset of machine learning tools) using ANN (Artificial Neural Networks), TensorFlow is one such tool which is an open-source machine learning ANN framework and will help achieve it in a better way.

**References**

1. H. McNairna,\*, C. Duguayb, B. Briscoc, T.J. Pultza (2002). The effect of soil and crop residue characteristics on polarimetric radar response. Remote Sensing of Environment, Volume 80, Issue 2, Pages 308-320.
2. Jun Wen, Zhongbo Su (2003). A time series based method for estimating relative soil moisture with ERS wind scatterometer data. Geophysical Research Letters, Volume 30, Issue 7.
3. B.J. Choudury, T.J. Schmugge, R.W. Newton and A.Chang (1978). Effect of Surface Roughness on the Microwave Emission from Soils. Journal of Geophysical Research, Volume 84, Issue C9, Pages 5699–5706
4. J.R. Wang, P.E. O'Neill, T.J. Jackson, E.T. Engman (1982). A Multi-Frequency Measurement of Thermal Microwave Emission From Soils: The Effects of Soil Texture and Surface Roughness. International Geoscience and Remote Sensing Symposium, Munich.
5. L. Breiman, “Pasting small votes for classification in large databases and on-line”, Machine Learning, 36(1), 85-103, 1999.
6. L. Breiman, “Bagging predictors”, Machine Learning, 24(2), 123-140, 1996.
7. T. Ho, “The random subspace method for constructing decision forests”, Pattern Analysis and Machine Intelligence, 20(8), 832-844, 1998.
8. G. Louppe and P. Geurts, “Ensembles on Random Patches”, Machine Learning and Knowledge Discovery in Databases, 346-361, 2012.
9. P. Geurts, D. Ernst., and L. Wehenkel, “Extremely randomized trees”, Machine Learning, 63(1), 3-42, 2006.
10. J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, Vol. 29, No. 5, 2001.
11. Friedman, Stochastic Gradient Boosting, 1999
12. T. Hastie, R. Tibshirani and J. Friedman. Elements of Statistical Learning Ed. 2, Springer, 2009.