A Project Report On

IRRIGATION CLOUD SYSTEMS USING MACHINE LANGUAGE

Submitted by

K.Swathi 15311A1242

K.Shreya 15311A1249

B.Shivani 15311A1260

Under the guidance of

Praneeth Kulkarni, Path Creators

In partial fulfillment of the requirement for the award of the Degree of

B. TECH

In

Information Technology

SREENIDHI INSTITUTE OF SCIENCE AND TECHNOLOGY

(An Autonomous institution)

Yamnampet, Ghatkesar Mandal, Medchal Dist.., Hyderabad-501301

Affiliated to

Jawaharlal Nehru Technology University

Hyderabad - 500085

2018

Sreenidhi Institute of Science and Technology

(An Autonomous institution)

Department of Information Technology

DECLARATION BY THE CANDIDATE

I hereby declare that the Internship report entitled "**IRRIGATION CLOUD SYSTEMS**" submitted by me to **Jawaharlal Nehru Technology University, Hyderabad** in partial fulfillment of the requirements for the award of the degree of **B.Tech in Information Technology** is a record of bonafide work carried out by me under the supervision of Coordinators ______________________________. I further declare that the work reported has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or university.

Sreenidhi Institute of Science And Technology

(An autonomous institution)

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the Project report on **“IRRIGATION CLOUD SYSTEMS”** submitted by **B.Shivani(15311A1260),K.Swathi(15311A1242),K.Shreya(15311A1249)** in the partial fulfillment for the award of B.Tech degree in **Information Technology** affiliated to **Jawaharlal Nehru Technology University, Hyderabad** is a record of bonafide work carried out by her under our guidance. The report fulfills the requirements as per the regulations of this University and in our opinion meets the necessary standards for submission. The results embodied in the project work have and not been submitted to any other University or institute for the award of any degree or diploma.

Head of the Department

(Dr. V. V. S. S. S. Balaram)

Acknowledgements

Foremost, I would like to express our deepest appreciation to the **Sreenidhi Institute of Science and Technology, Hyderabad** management who provided us the possibility to complete this report.

We would like to express our sincere gratitude to **Dr.T.CH. Shiva Reddy, Principal of Sreenidhi Institute of Science and Technlogy, Hyderabad** for the continuous support in completion of my report.

We own the great gratitude to **Dr.Balaram Head of the Department of Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad f**or approving the work with great interest and encouragement during the completion of work in the institution.

I would like to express our special appreciation and thanks to our coordinator **Praneeth Kalluri of Path Creators** who has been a tremendous mentor for us. I would like to thank him for encouraging our work and for allowing us to grow as a graduate engineer. Your advice on both work as well as on our career has been priceless.

We also thank all the faculty members of **Sreenidhi Institute of Science and Technology, Hyderabad** for giving me the courage and strength we needed to complete our goals. This acknowledgement would be incomplete without expressing our whole hearted thanks to my family and friends who motivated me during the course of work.

Table of Contents

ABSTRACT: An automatic irrigation control system has been designed to

facilitate the automatic supply of adequate of water from a reservoir to field or domestic crops in all agricultural seasons. One of the objectives of this work is to see how human control could be removed from irrigation and also to optimize the use of water in the process. The method employed is to continuously monitor the soil moisture level to decide whether irrigation is needed, and how much water is needed in the soil.

A pumping mechanism is used to deliver the needed amount of water to the soil. The work can be grouped into four subsystems namely; power supply, sensing unit, control unit and pumping subsystems which make up the automatic irrigation control system.

A moisture sensor was constructed to model the electrical resistance of the soil; a regulated 12 volts power supply unit was constructed to power the system; the control circuit was implemented using operational amplifier and timer; and the pumping subsystem consisting of a submersible low-noise micro water pump was constructed using a small dc-operated motor.

System response tests were carried out to determine the time taken for the system to irrigate potted samples of different soil types having different levels of dryness. The results obtained showed that sandy soils require less water than loamy soils and clay soils require the most water for irrigation.

Chapter 1

Introduction

In this chapter introductions on Irrigation Cloud Systems, Python and Machine Learning.

1.1 Introduction to Irrigation Cloud Systems

The rise in energy demand has outpaced power generation capacity due to the high increase in population and industries. This calls for management of demand to optimize the usage of the limited generated power. One of the areas where power is so essential is irrigation. There is always need to pump water to the water tanks and operate the irrigation system such as sprinklers. However, two scarce and valuable resources of irrigation, i.e. water and energy, are not efficiently utilized by the current irrigation systems. They do not have the means to determine where and when irrigation is required. Consequently, irrigation is sometimes performed when it is not necessary or delayed when required. This leads to water/energy waste and low-crop yield, respectively. These challenges can be mitigated if the irrigation system was able to determine precisely when and where to irrigate. A lot of research has been done to address this. In an approach for integrating precision agriculture and smart grid technologies is presented. This aims at balancing consumption and generation in the farmland, which increases the sustainability of water supply.

The coordination with the Smart Grid operator enables farmers to save on energy costs and support grid at peak hours. However, there is need for minimizing the amount of energy and water that is used in the farm. Furthermore, the tools and equipment used in the implementation of this approach make it rather costly which compromises its feasibility.

1.2 Introduction to Machine Learning

A branch of **artificial intelligence**, concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data.
As intelligence requires knowledge, it is necessary for the computers to acquire knowledge.
According to Tom M. Mitchell, a computer is said to learn from experience E with respect to some class of tasks T and performance measure P,if its

performance at tasks in T, as measured by P improves with experience

- A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data).
- Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases.
- Some machine learning systems attempt to eliminate the need for human interaction in data analysis, while others adopt a collaborative approach between human and machine.
- Human intuition cannot, however, be entirely eliminated, since the system's designer must specify how the data is to be represented and what mechanisms will be used to search for a characterization of the data.
- **Supervised learning**
    - Prediction
    - Classification (discrete labels), Regression (real values)
- **Unsupervised learning**
    - Clustering
    - Probability distribution estimation
    - Finding association (in features)
    - Dimension reduction
- **Semi-supervised learning**
- **Reinforcement learning**
    - Decision making (robot, chess machine)
- The success of machine learning system also depends on the algorithms.
- The algorithms control the search to find and build the knowledge structures.
- The learning algorithms should extract useful information from training examples.
- Supervised learning categories and techniques
    - **Linear classifier** (numerical functions)
    - **Parametric** (Probabilistic functions)
        - Naïve Bayes, Gaussian discriminant analysis (GDA), Hidden Markov models (HMM), Probabilistic graphical models
    - **Non-parametric** (Instance-based functions)
        - *K*-nearest neighbors, Kernel regression, Kernel density estimation, Local regression

- **Non-metric** (Symbolic functions)
    - Classification and regression tree (CART), decision tree
- **Aggregation** -Bagging (bootstrap + aggregation), Adaboost, Random forest

1.3 Introduction to Python

Python is a popular programming language. It was created in 1991 by Guido van Rossum.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.
- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.
- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Chapter 2

Implementation

Data collection is not a simple task, as it may seem. Various decisions have to be made for collecting data. For my project I used dataset for training, testing. In this chapter we are going to study how data is collected, stored, processed and classified. Before discussing these process there is a brief discussion on the platform, modules imported and algorithms.

2.1 Proposed Platform

Anaconda is a Python distribution that is particularly popular for data analysis and scientific computing Open source project developed by Continuum Analytics, Inc. Available for Windows, Mac OS X and Linux Includes many popular packages:
NumPy, SciPy, Matplotlib, Pandas, IPython, Cython Includes Spyder, a Python development environment Includes conda, a platform-independent package manager.
Anaconda is easy to

- Install Download installer from https://www.continuum.io/download
- Execute the installer and follow the instructions
- Anaconda is installed on Yale clusters Omega and Grace
- $ module load Langs/Python/2.7-anaconda

ABOUT JUPYTER

The Jupyter Notebook is an incredibly powerful tool for interactively developing and presenting data science projects. As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user’s code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

PYTHON MODULES

1.PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, **Series** (1-dimensional) and **DataFrame** (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

2.NUMPY:

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy:

Using NumPy, a developer can perform the following operations −

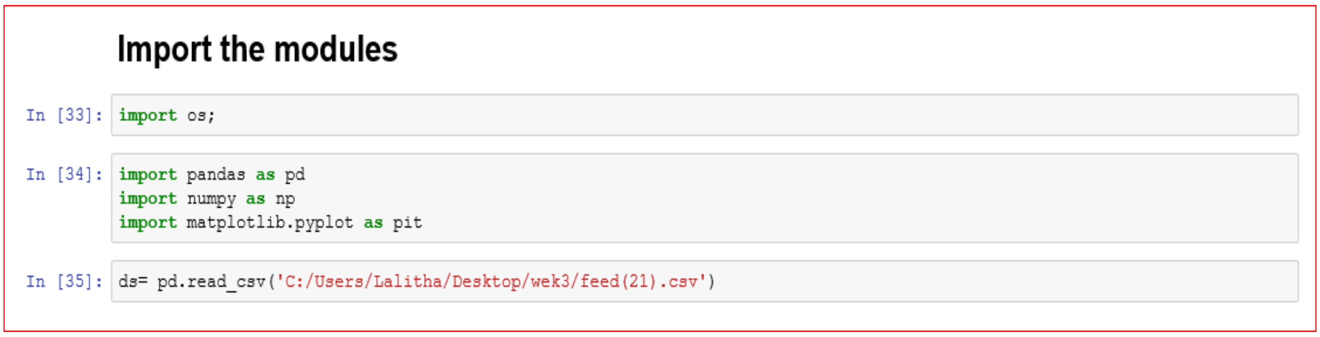
- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

3.MATPLOTLIB:

Matplotlib **is a plotting library for the** Python **programming language and its numerical mathematics extension NumPy**

- The library itself is huge, at something like 70,000 total lines of code.

- Matplotlib is home to several different *interfaces* (ways of constructing a figure) and capable of interacting with a handful of different *backend*. (Backend deal with the process of how charts are actually rendered, not just structured internally.



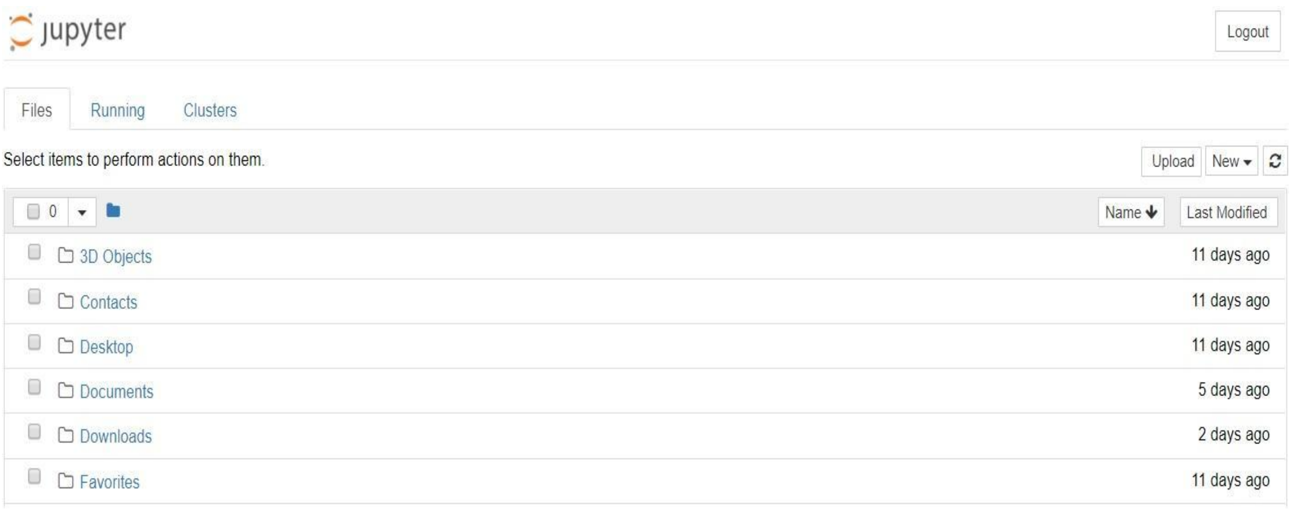
Import the modules

```
In [33]: import os;

In [34]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as pit

In [35]: ds= pd.read_csv('C:/Users/Lalitha/Desktop/wek3/feed(21).csv')
```

2.2Running Python

On Windows, you can run Jupyter via the shortcut Anaconda adds to your start menu, which will open a new tab in your default web browser that should look something like the following screenshot.

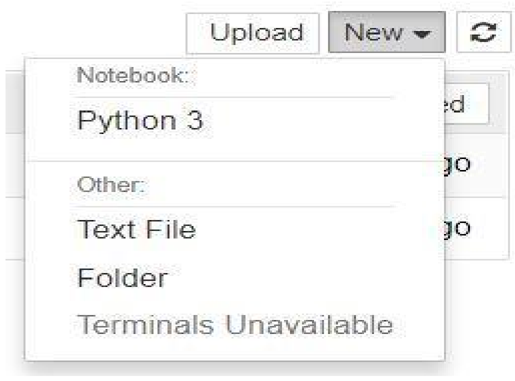
It is also possible to start the dashboard on any system via the command prompt (or terminal on Unix systems) by entering the command `jupyter notebook`; in this case, the current working directory will be the start-up directory.



Select items to perform actions on them.

3D Objects	11 days ago
Contacts	11 days ago
Desktop	11 days ago
Documents	5 days ago
Downloads	2 days ago
Favorites	11 days ago

Browse to the folder in which you would like to create your first notebook, click the "New" drop-down button in the top-right and select "Python 3" (or the version of your choice)



What is an ipynb File?

It will be useful to understand what this file really is. Each `.ipynb` file is a text file that describes the contents of your notebook in a format called JSON. Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some metadata. We can edit this — by selecting "Edit > Edit Notebook Metadata" from the menu bar in the notebook.

There are two fairly prominent terms that you should notice, which are probably new to you: *cells* and *kernels* are key both to understanding Jupyter and to what makes it more than just a word processor. Fortunately, these concepts are not difficult to understand.

- A kernel is a "computational engine" that executes the code contained in a notebook document.
- A cell is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel.

Cells

- A **code cell** contains code to be executed in the kernel and displays its output below.
- A **Markdown cell** contains text formatted using Markdown and displays its output in-place when it is run.

The first cell in a new notebook is always a code cell. Let's test it out with a classic hello world example. Type `print('Hello World!')` into the cell and click the run button Run in the toolbar above or press `Ctrl + Enter`. The result should look like this:

```
print('Hello World!')
Hello World!
```

When you ran the cell, its output will have been displayed below and the label to its left will have changed from `In [ ]` to `In [1]`. The output of a code cell also forms part of the document, which is why you can see it in this article. You can always tell the difference between code and Markdown cells because code cells have that label on the left and Markdown cells do not. The "In" part of the label is simply short for "Input," while the label number indicates when the cell was executed on the kernel — in this case the cell was executed first. Run the cell again and the label will change to `In [2]`because now the cell was the second to be run on the kernel. It will become clearer why this is so useful later on when we take a closer look at kernels.

```
import time
time.sleep(3)
```

This cell doesn't produce any output, but it does take three seconds to execute. Notice how Jupyter signifies that the cell is currently running by changing its label to `In [*]`. In general, the output of a cell comes from any text data specifically printed during the cells execution, as well as the value of the last line in the cell, be it a lone variable, a function call, or something else. For example:a

```
defsay_hello(recipient):
return'Hello, {}!'.format(recipient)

say_hello('Tim')
'Hello, Tim!'
```

Kernels

Behind every notebook runs a kernel. When you run a code cell, that code is executed within the kernel and any output is returned back to the cell to be displayed. The kernel's state persists over time and between cells — it pertains to the document as a whole and not individual cells.

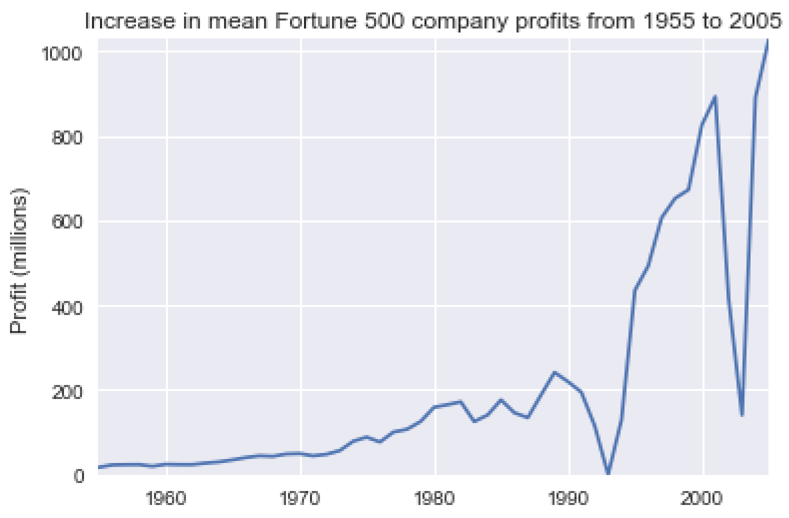
Plotting with matplotlib

Next, we can get to addressing the question at hand by plotting the average profit by year. We might as well plot the revenue as well, so first we can define some variables and a method to reduce our code.

```
group_by_year                                                          =
df.loc[:,['year','revenue','profit']].groupby('year')
avgs = group_by_year.mean()
x = avgs.index
y1 = avgs.profit

defplot(x, y, ax, title, y_label):
  ax.set_title(title)
  ax.set_ylabel(y_label)
  ax.plot(x, y)
  ax.margins(x=0, y=0)
```

After plotting, we get,

```
plot(x, y1, ax,'Increase in mean Fortune 500 company profits
from 1955 to 2005','Profit (millions)')
```



2.4 Algorithms

a. **Linear Regression**

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation Y= a *X + b.

The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

In this equation:

- Y – Dependent Variable

- a – Slope
- X – Independent variable
- b – Intercept

Linear Regression is of mainly two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression(as the name suggests) is characterized by multiple (more than 1) independent variables. While finding best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.

b. **Logistic Regression**

It is a classification not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variable(s). In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logistic regression**. Since, it predicts the probability, its output values lies between 0 and 1 (as expected).Again, let us try and understand this through a simple example.

Let’s say your friend gives you a puzzle to solve. There are only 2 outcome scenarios either you solve it or you don’t. Now imagine, that you are being given wide range of puzzles / quizzes in an attempt to understand which subjects you are good at. The outcome to this study would be something like this – if you are given a trigonometry based tenth grade problem, you are 70% likely to solve it. On the other hand, if it is grade fifth history question, the probability of getting an answer is only 30%. This is what Logistic Regression provides you.

c. SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.For example, if we only had two features like Height and Hair length of an individual, we’d first plot these two variables in two dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**)

d. **KNN (K-Nearest Neighbors)**

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modeling.

Things to consider before selecting kNN:

- KNN is computationally expensive
- Variables should be normalized else higher range variables can bias it
- Works on pre-processing stage more before going for kNN like outlier, noise removal

e. **Random Forest Classifier**

Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node

f. **Decision Tree Classifier**

Decision Tree Classifier is a simple and widely used classification technique. It applies a straitforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the calss label of the record is reached. Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too. The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

g. **K means Clustering**

K-means clustering is a type of unsupervised learning, which is used when you have unlabelled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable *K*. The algorithm works iteratively to assign each data point to one of *K* groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the *K*-means clustering algorithm are:

1. The centroids of the K clusters, which can be used to label new data

2. Labels for the training data (each data point is assigned to a single cluster)

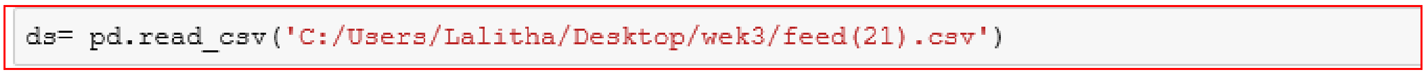
Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

The *K*-means clustering algorithm uses iterative refinement to produce a final result. The algorithm inputs are the number of clusters *K* and the data set. The data set is a collection of features for each data point. The algorithms starts with initial estimates for the *K* centroids, which can either be randomly generated or randomly selected from the data set.

Chapter 3

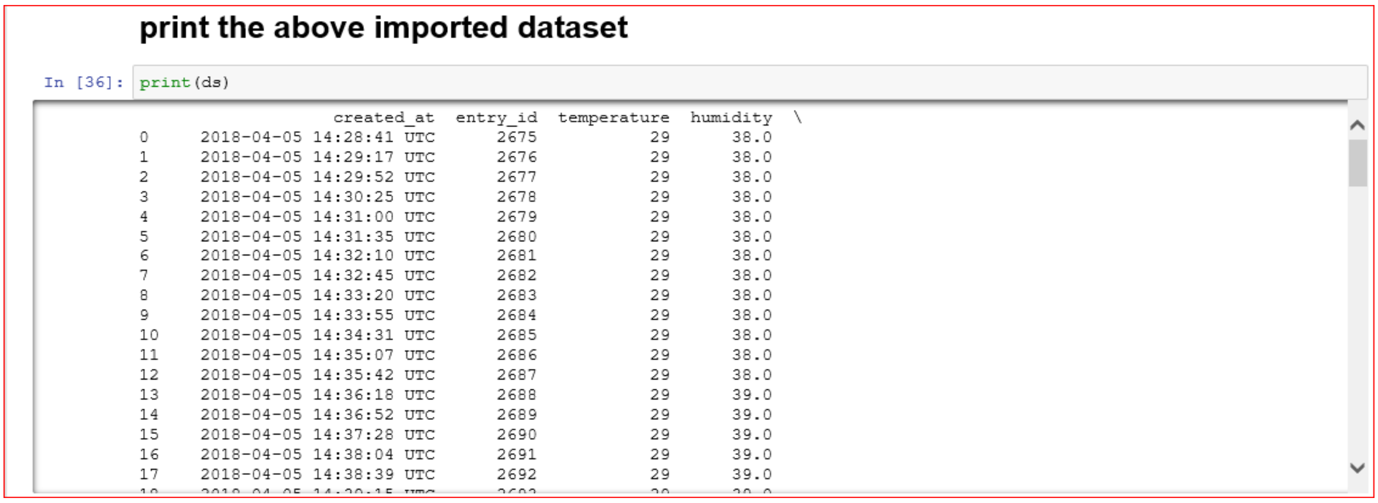
RESULT AND ANALYSIS

1. **Importing the dataset**



```
ds= pd.read_csv('C:/Users/Lalitha/Desktop/wek3/feed(21).csv')
```

2. **To print the data set**

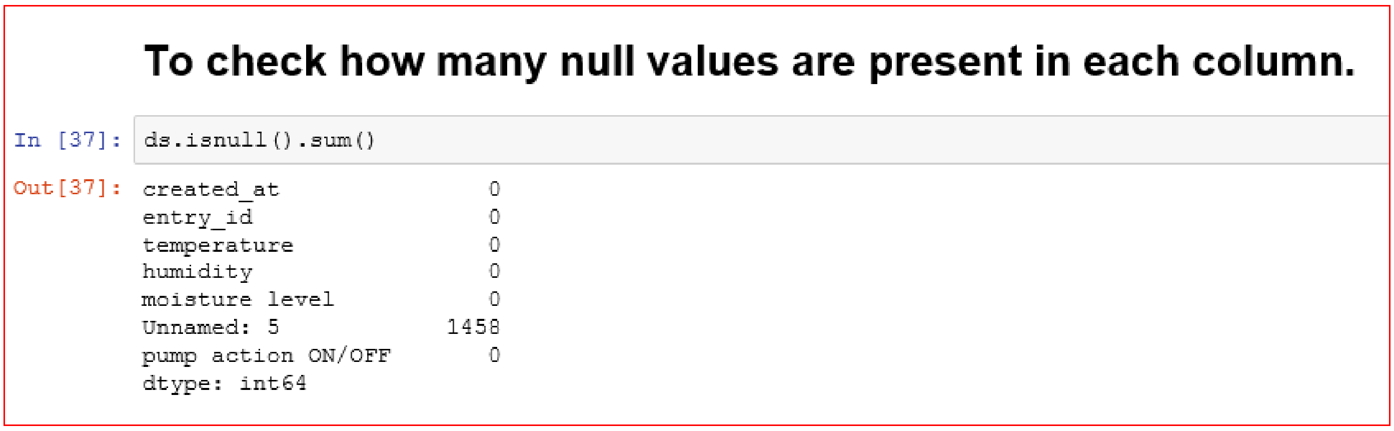


print the above imported dataset

```
In [36]: print(ds)
```

```
                     created_at  entry_id  temperature  humidity  \
0       2018-04-05 14:28:41 UTC      2675           29      38.0
1       2018-04-05 14:29:17 UTC      2676           29      38.0
2       2018-04-05 14:29:52 UTC      2677           29      38.0
3       2018-04-05 14:30:25 UTC      2678           29      38.0
4       2018-04-05 14:31:00 UTC      2679           29      38.0
5       2018-04-05 14:31:35 UTC      2680           29      38.0
6       2018-04-05 14:32:10 UTC      2681           29      38.0
7       2018-04-05 14:32:45 UTC      2682           29      38.0
8       2018-04-05 14:33:20 UTC      2683           29      38.0
9       2018-04-05 14:33:55 UTC      2684           29      38.0
10      2018-04-05 14:34:31 UTC      2685           29      38.0
11      2018-04-05 14:35:07 UTC      2686           29      38.0
12      2018-04-05 14:35:42 UTC      2687           29      38.0
13      2018-04-05 14:36:18 UTC      2688           29      39.0
14      2018-04-05 14:36:52 UTC      2689           29      39.0
15      2018-04-05 14:37:28 UTC      2690           29      39.0
16      2018-04-05 14:38:04 UTC      2691           29      39.0
17      2018-04-05 14:38:39 UTC      2692           29      39.0
```

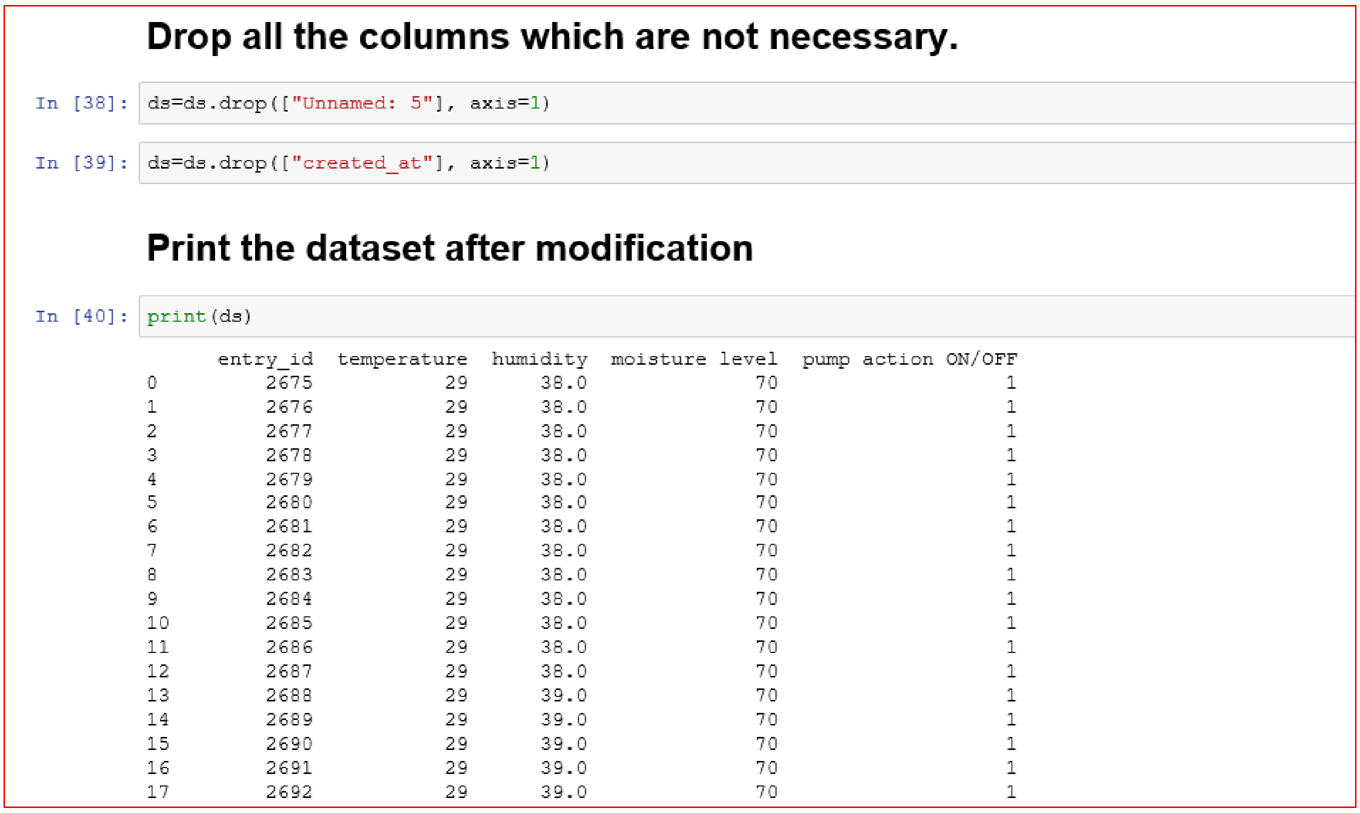
3. **Check null values present in the data set**



To check how many null values are present in each column.

```
In [37]: ds.isnull().sum()

Out[37]: created_at              0
         entry_id                0
         temperature             0
         humidity                0
         moisture level          0
         Unnamed: 5           1458
         pump action ON/OFF      0
         dtype: int64
```

4. To remove null values and print the dataset.



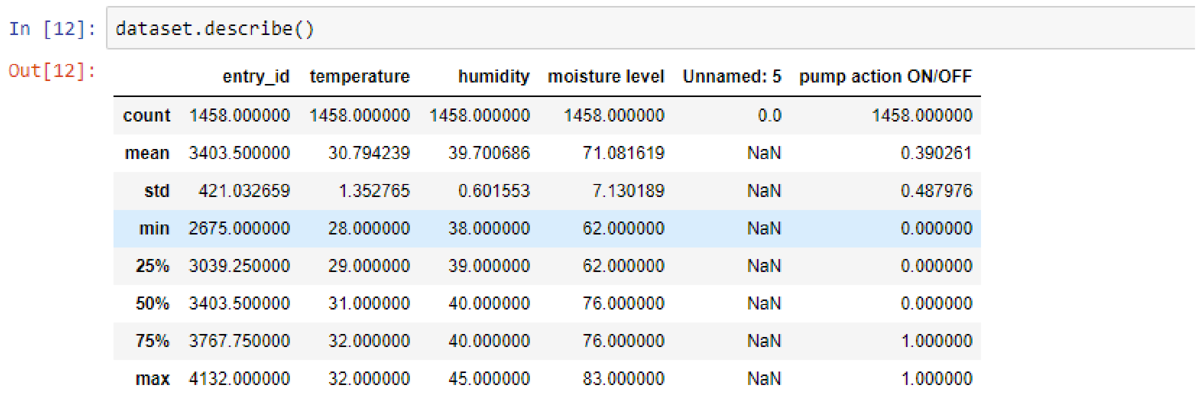
Drop all the columns which are not necessary.

```
In [38]: ds=ds.drop(["Unnamed: 5"], axis=1)

In [39]: ds=ds.drop(["created_at"], axis=1)
```

Print the dataset after modification

```
In [40]: print(ds)

       entry_id  temperature  humidity  moisture level  pump action ON/OFF
0          2675           29      38.0              70                   1
1          2676           29      38.0              70                   1
2          2677           29      38.0              70                   1
3          2678           29      38.0              70                   1
4          2679           29      38.0              70                   1
5          2680           29      38.0              70                   1
6          2681           29      38.0              70                   1
7          2682           29      38.0              70                   1
8          2683           29      38.0              70                   1
9          2684           29      38.0              70                   1
10         2685           29      38.0              70                   1
11         2686           29      38.0              70                   1
12         2687           29      38.0              70                   1
13         2688           29      39.0              70                   1
14         2689           29      39.0              70                   1
15         2690           29      39.0              70                   1
16         2691           29      39.0              70                   1
17         2692           29      39.0              70                   1
```

i. Describing the numerical data



```
In [12]: dataset.describe()
```

Out[12]:

	entry_id	temperature	humidity	moisture level	Unnamed: 5	pump action ON/OFF
count	1458.000000	1458.000000	1458.000000	1458.000000	0.0	1458.000000
mean	3403.500000	30.794239	39.700686	71.081619	NaN	0.390261
std	421.032659	1.352765	0.601553	7.130189	NaN	0.487976
min	2675.000000	28.000000	38.000000	62.000000	NaN	0.000000
25%	3039.250000	29.000000	39.000000	62.000000	NaN	0.000000
50%	3403.500000	31.000000	40.000000	76.000000	NaN	0.000000
75%	3767.750000	32.000000	40.000000	76.000000	NaN	1.000000
max	4132.000000	32.000000	45.000000	83.000000	NaN	1.000000

Since Unnamed column consist of NaN's we remove it from the dataset as:
dataset.drop('Unnamed:5',1)
Since last row does contain NaN drop the row as:

dataset=dataset.drop(dataset.index[1458])

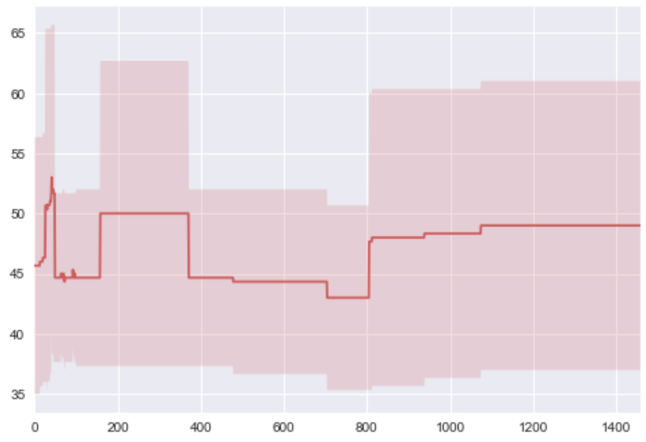
```
In [24]: dataset.tail()
```

Out[24]:

	created_at	entry_id	temperature	humidity	moisture level	pump action ON/OFF
1453	2018-04-05 16:00:04 UTC	4128.0	31.0	40.0	76.0	0.0
1454	2018-04-05 16:00:04 UTC	4129.0	31.0	40.0	76.0	0.0
1455	2018-04-05 16:00:04 UTC	4130.0	31.0	40.0	76.0	0.0
1456	2018-04-05 16:00:04 UTC	4131.0	31.0	40.0	76.0	0.0
1457	2018-04-05 16:00:04 UTC	4132.0	31.0	40.0	76.0	0.0

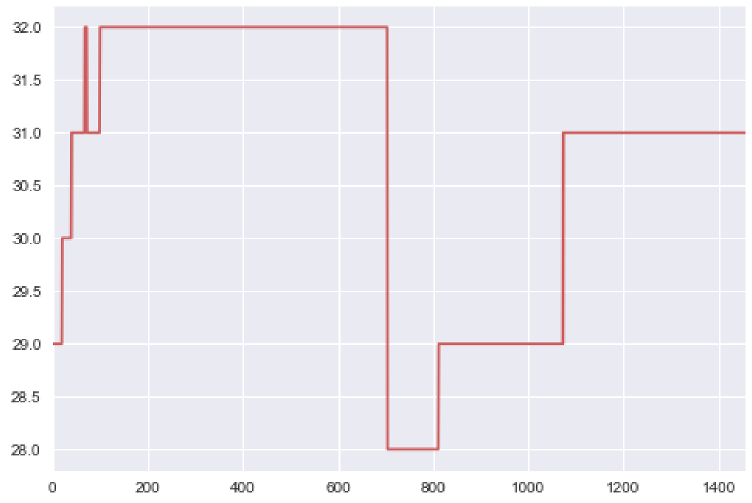
ii. Plotting the features

```
In [42]: dataset=dataset.set_index(dataset.created_at)
         sns.tsplot([dataset.temperature,dataset.humidity,dataset['moisture level']],color="indianred")

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0xc85d588>
```



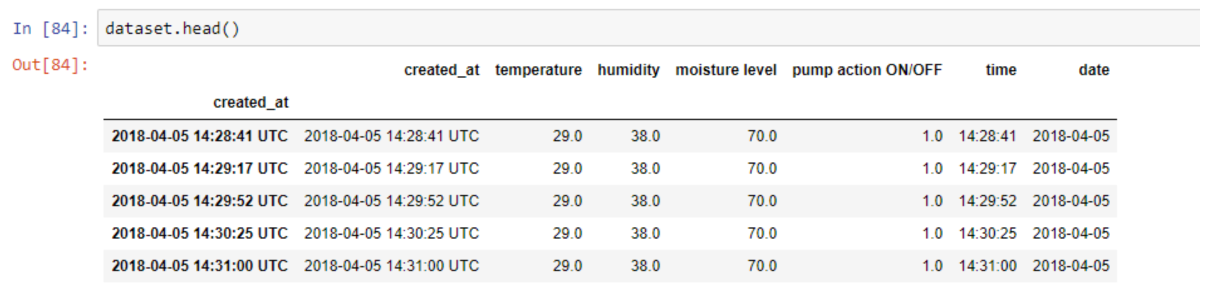
iii. Plotting the temperature values

```
In [66]: sns.tsplot(dataset.temperature,color="indianred")

Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x10cdcf60>
```



iv. Creating date and time columns

dataset['time']=pd.DatetimeIndex(dataset['created_at']).time dataset['date']=pd.DatetimeIndex(dataset['created_at']).date



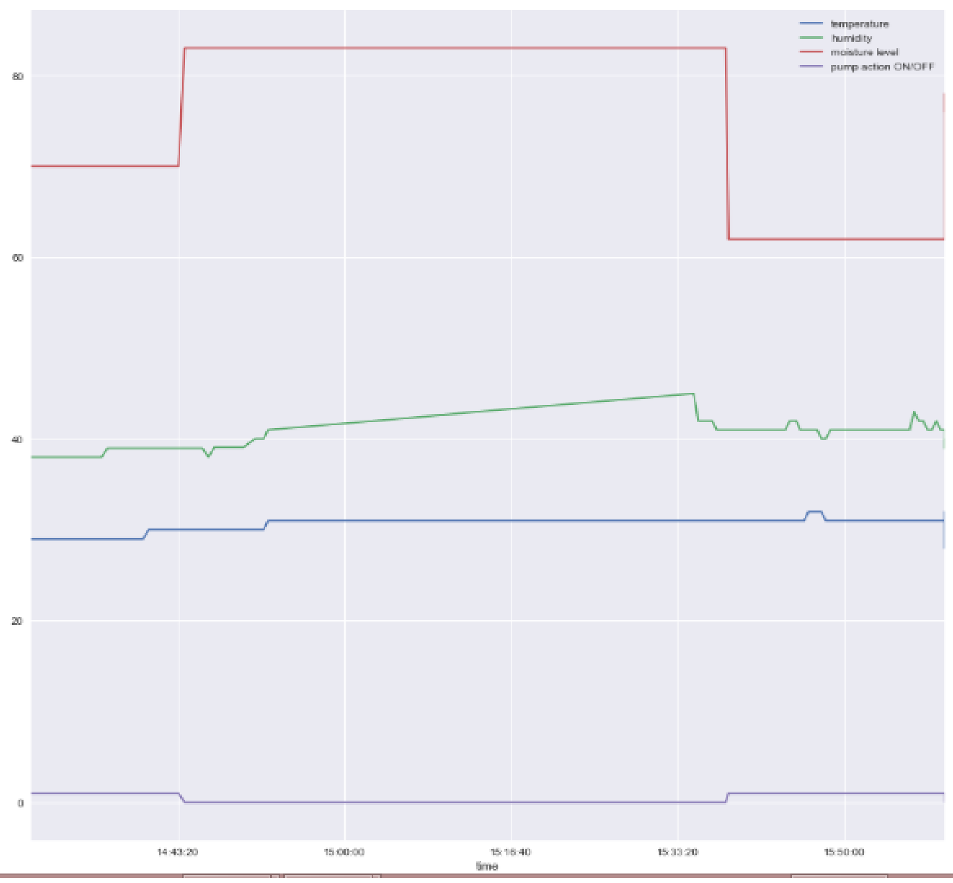
```
In [84]: dataset.head()
Out[84]:
```

created_at	created_at	temperature	humidity	moisture level	pump action ON/OFF	time	date
2018-04-05 14:28:41 UTC	2018-04-05 14:28:41 UTC	29.0	38.0	70.0	1.0	14:28:41	2018-04-05
2018-04-05 14:29:17 UTC	2018-04-05 14:29:17 UTC	29.0	38.0	70.0	1.0	14:29:17	2018-04-05
2018-04-05 14:29:52 UTC	2018-04-05 14:29:52 UTC	29.0	38.0	70.0	1.0	14:29:52	2018-04-05
2018-04-05 14:30:25 UTC	2018-04-05 14:30:25 UTC	29.0	38.0	70.0	1.0	14:30:25	2018-04-05
2018-04-05 14:31:00 UTC	2018-04-05 14:31:00 UTC	29.0	38.0	70.0	1.0	14:31:00	2018-04-05

v. Each feature against time

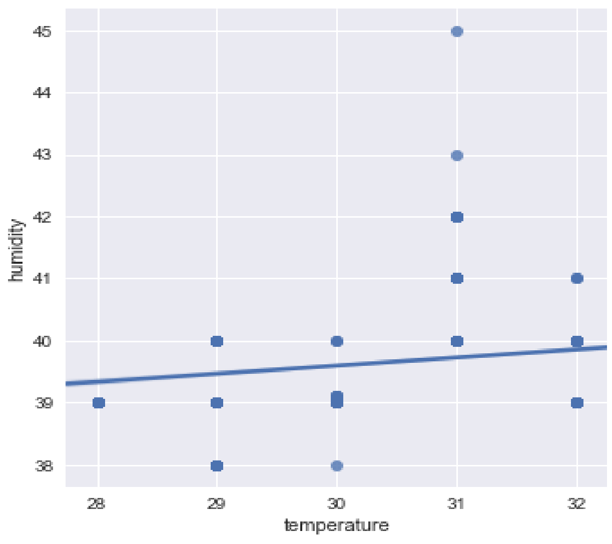
dataset.set_index('time').plot(figsize=(15,15))

```
In [74]: dataset.set_index('time').plot(figsize=(15,15))
Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0xeed0400>
```

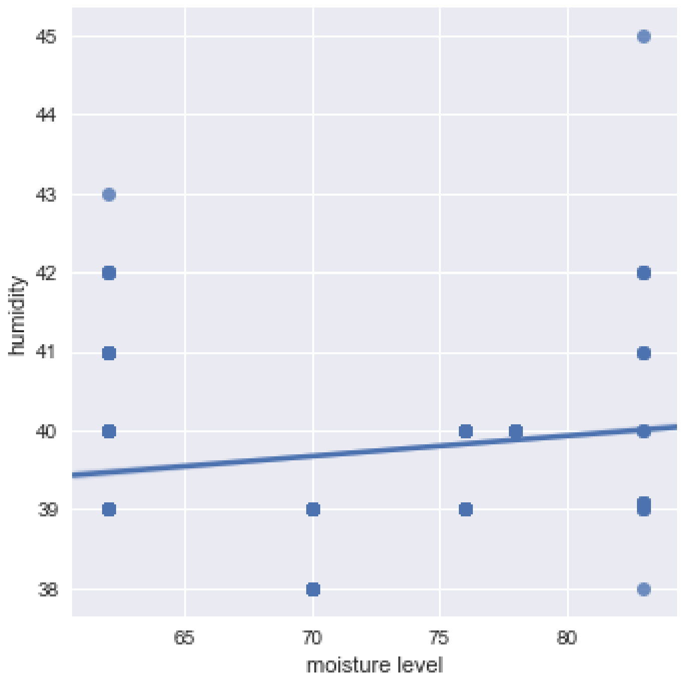


According to the the above plot we found that during the time span of 14:43:20 to 15:40:00 the humidity raised constantly and mositure remained constant and the pump was OFF.

```
In [79]: sns.lmplot('temperature','humidity',data=dataset)
Out[79]: <seaborn.axisgrid.FacetGrid at 0xfa92ef0>
```

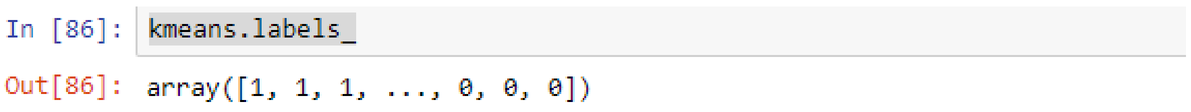


```
In [80]: sns.lmplot('moisture level','humidity',data=dataset)
Out[80]: <seaborn.axisgrid.FacetGrid at 0x114e5e80>
```



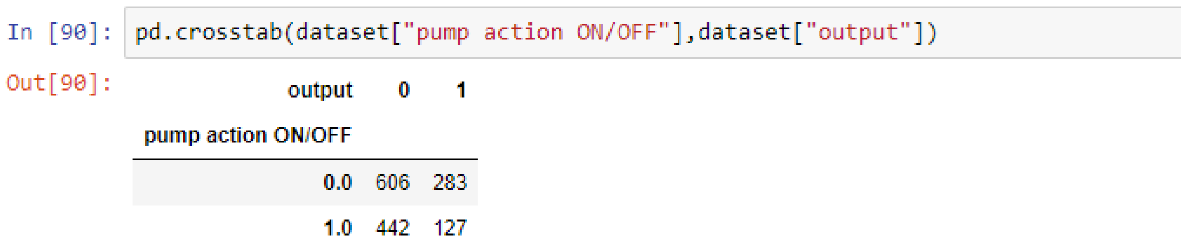
vi. Modeling

```
f1=dataset['temperature'].values
f2=dataset['humidity'].values
X=np.matrix(zip(f1,f2))
kmeans = KMeans(n_clusters=2).fit(X)
```



```
In [86]: kmeans.labels_
Out[86]: array([1, 1, 1, ..., 0, 0, 0])
```

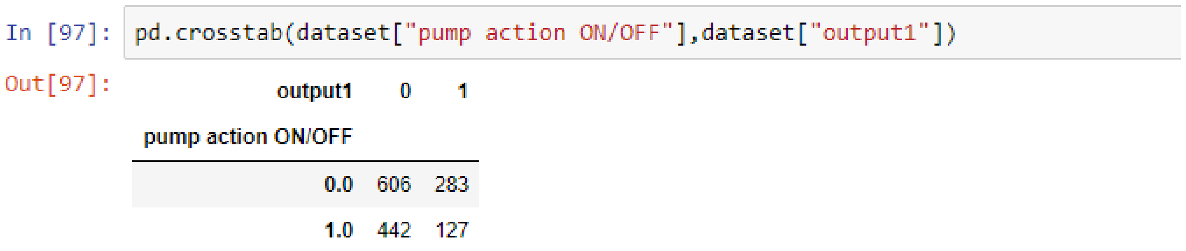
dataset['output']=kmeans.labels_



```
In [90]: pd.crosstab(dataset["pump action ON/OFF"],dataset["output"])
Out[90]:
```

output	0	1
pump action ON/OFF		
0.0	606	283
1.0	442	127

x1=dataset['humidity'].values
x2=dataset['moisture level'].values
X=np.matrix(zip(f1,f2))
kmeans = KMeans(n_clusters=2).fit(X)
dataset['output1']=kmeans.labels_



```
In [97]: pd.crosstab(dataset["pump action ON/OFF"],dataset["output1"])
Out[97]:
```

output1	0	1
pump action ON/OFF		
0.0	606	283
1.0	442	127

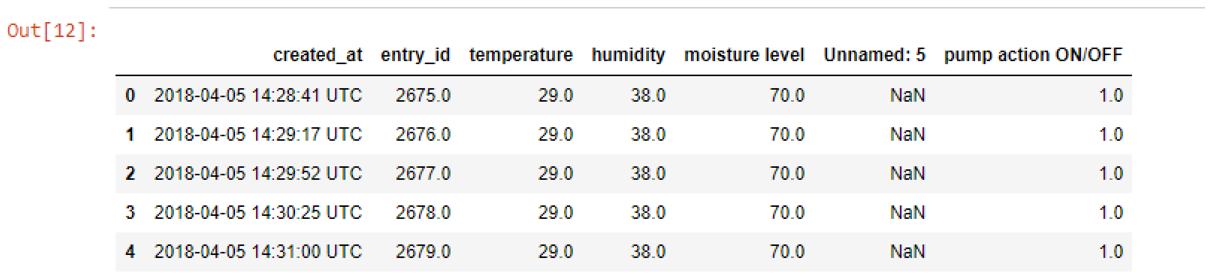
b. Modeling using Supervised algorithms

i. Import the required modules

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

ii. Reading the dataset

dataset=pd.read_csv("C:/Users/user/Desktop/data_set.csv")
dataset.head()



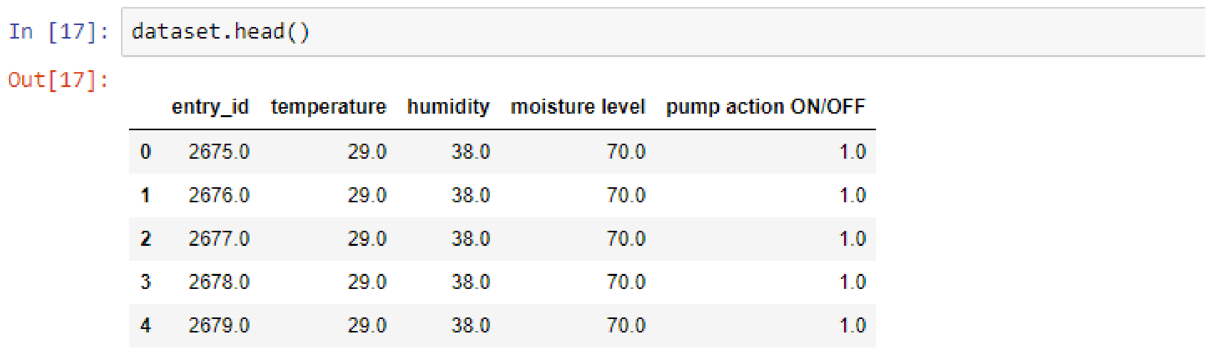
```
Out[12]:
```

	created_at	entry_id	temperature	humidity	moisture level	Unnamed: 5	pump action ON/OFF
0	2018-04-05 14:28:41 UTC	2675.0	29.0	38.0	70.0	NaN	1.0
1	2018-04-05 14:29:17 UTC	2676.0	29.0	38.0	70.0	NaN	1.0
2	2018-04-05 14:29:52 UTC	2677.0	29.0	38.0	70.0	NaN	1.0
3	2018-04-05 14:30:25 UTC	2678.0	29.0	38.0	70.0	NaN	1.0
4	2018-04-05 14:31:00 UTC	2679.0	29.0	38.0	70.0	NaN	1.0

iii. Data Preprocessing

Remove the unwanted rows and columns as there are empty values in the "Unnamed: 5" column,therefore we can eliminate it.Drop is a function available in pandas library which is used to remove the mentioned columns or rows.axis is an attribute of drop function.It's value 0 indicates rows and 1 indicates columns)

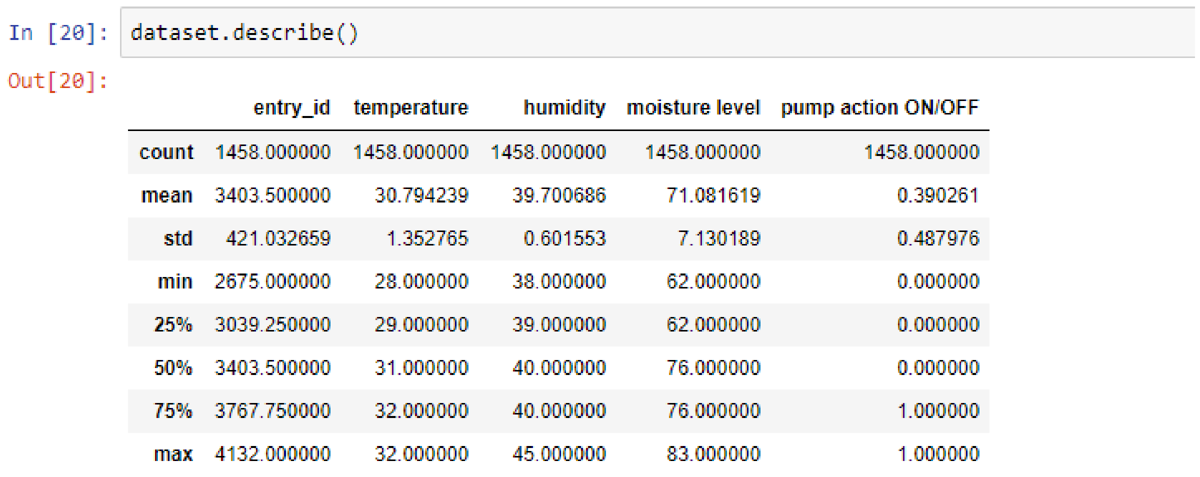
```
dataset=dataset.drop(['Unnamed: 5'],axis=1)
dataset=dataset.drop(['created_at'],axis=1)
dataset.drop(dataset.index[1458],axis=0,inplace=True)
```



```
In [17]: dataset.head()
Out[17]:
```

	entry_id	temperature	humidity	moisture level	pump action ON/OFF
0	2675.0	29.0	38.0	70.0	1.0
1	2676.0	29.0	38.0	70.0	1.0
2	2677.0	29.0	38.0	70.0	1.0
3	2678.0	29.0	38.0	70.0	1.0
4	2679.0	29.0	38.0	70.0	1.0

iv. Describing the dataset

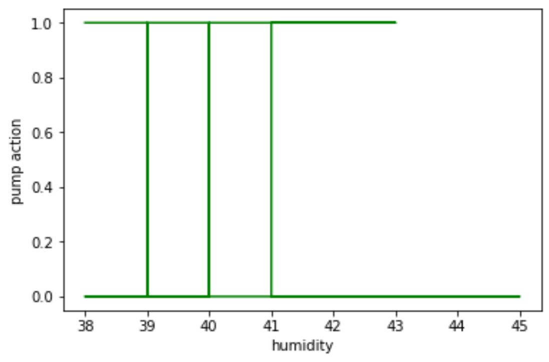


```
In [20]: dataset.describe()
Out[20]:
```

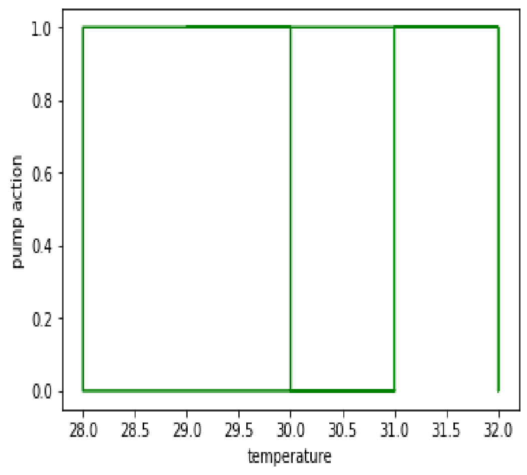
	entry_id	temperature	humidity	moisture level	pump action ON/OFF
count	1458.000000	1458.000000	1458.000000	1458.000000	1458.000000
mean	3403.500000	30.794239	39.700686	71.081619	0.390261
std	421.032659	1.352765	0.601553	7.130189	0.487976
min	2675.000000	28.000000	38.000000	62.000000	0.000000
25%	3039.250000	29.000000	39.000000	62.000000	0.000000
50%	3403.500000	31.000000	40.000000	76.000000	0.000000
75%	3767.750000	32.000000	40.000000	76.000000	1.000000
max	4132.000000	32.000000	45.000000	83.000000	1.000000

v. Feature Selection

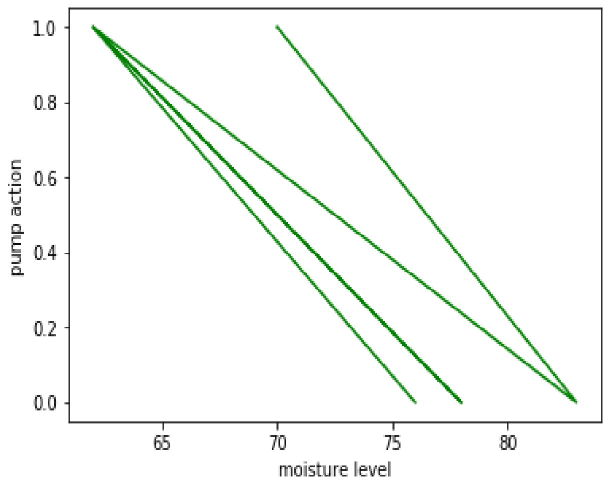
```
In [24]: plt.plot(dataset['humidity'],dataset['pump action ON/OFF'],color='g')
         plt.xlabel('humidity')
         plt.ylabel('pump action')

Out[24]: Text(0,0.5,'pump action')
```



```
In [23]: plt.plot(dataset['temperature'],dataset['pump action ON/OFF'],color='g')
         plt.xlabel('temperature')
         plt.ylabel('pump action')

Out[23]: Text(0,0.5,'pump action')
```



```
In [25]: plt.plot(dataset['moisture level'],dataset['pump action ON/OFF'],color='g')
         plt.xlabel('moisture level')
         plt.ylabel('pump action')

Out[25]: Text(0,0.5,'pump action')
```



vi. Splitting the Dataset

from sklearn.cross_validation import train_test_split

result=dataset['pump action ON/OFF']

X_train,X_test,y_train,y_test=train_test_split(dataset,result,train_size=0.75)

```
In [68]: X_train.count()

Out[68]: entry_id              1093
         temperature           1093
         humidity              1093
         moisture level        1093
         pump action ON/OFF    1093
         dtype: int64

In [69]: X_test.count()

Out[69]: entry_id              365
         temperature           365
         humidity              365
         moisture level        365
         pump action ON/OFF    365
         dtype: int64

In [70]: y_train.count()

Out[70]: 1093

In [71]: y_test.count()

Out[71]: 365
```

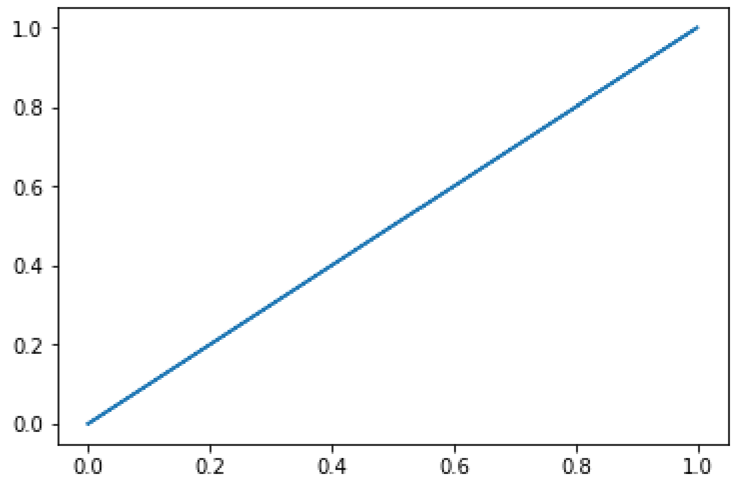
vii. Algorithms

a. Logistic Regression

```
In [72]: from sklearn.linear_model import LogisticRegression

In [73]: model=LogisticRegression()
         model.fit(X_train,y_train)

Out[73]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

```
In [74]: y_pred=model.predict(X_test)
         print(y_pred)

         [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
          1. 1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0.
          0. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 0. 1. 0. 1.
          0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0.
          0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1.
          0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0.
          0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1. 0. 1. 1.
          0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 0.
          0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0.
          1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 0. 1.
          0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1.
          0. 0. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 1. 0. 1.
          1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1.
          0. 0. 0. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1.
          0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1.
          1. 1. 1. 0. 0.]
```

```
In [75]: accuracy_score(y_test,y_pred)

Out[75]: 1.0

In [76]: plt.plot(y_test,y_pred)

Out[76]: [<matplotlib.lines.Line2D at 0xc9e8350>]
```

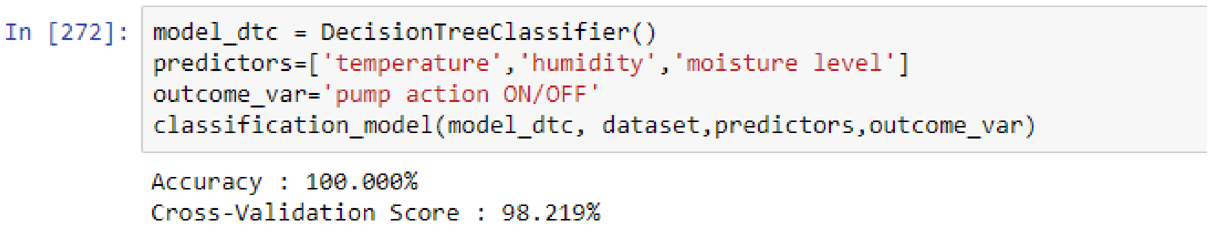


Defining a function for all the models

def classification_model(model, data, predictors, outcome):
 #Fit the model:

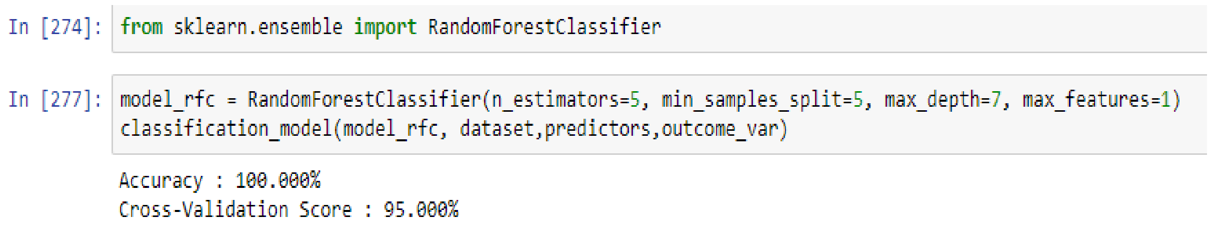
```
model.fit(data[predictors],data[outcome])

#Make predictions on training set:
predictions = model.predict(data[predictors])
#Print accuracy
accuracy = accuracy_score(predictions,data[outcome])
print("Accuracy : %s" % "{0:.3%}".format(accuracy))

#Perform k-fold cross-validation with 5 folds
kf = KFold(data.shape[0], n_folds=5)
error = []
for train, test in kf:
  # Filter training data
  train_predictors = (data[predictors].iloc[train,:])

  # The target we're using to train the algorithm.
  train_target = data[outcome].iloc[train]

  # Training the algorithm using the predictors and target.
  model.fit(train_predictors, train_target)

  #Record error from each cross-validation run
  error.append(model.score(data[predictors].iloc[test,:], data[outcome].iloc[test]))

print("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))

#Fit the model again so that it can be refered outside the function:
model.fit(data[predictors],data[outcome])
```

b. Decision Tree Classifier



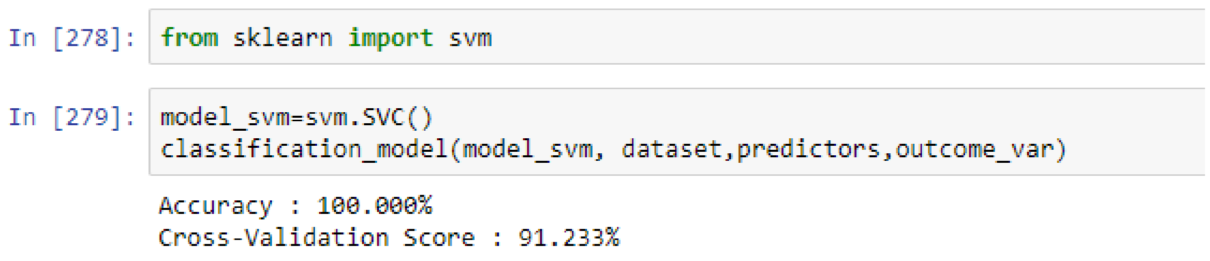
```
In [272]: model_dtc = DecisionTreeClassifier()
          predictors=['temperature','humidity','moisture level']
          outcome_var='pump action ON/OFF'
          classification_model(model_dtc, dataset,predictors,outcome_var)

          Accuracy : 100.000%
          Cross-Validation Score : 98.219%
```

c. Random Forest Classifier



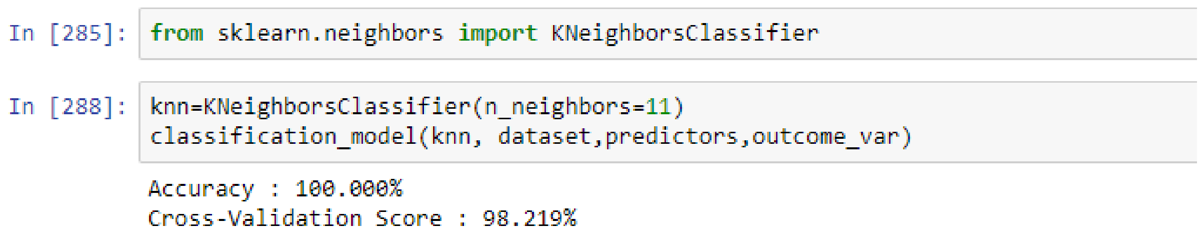
```
In [274]: from sklearn.ensemble import RandomForestClassifier

In [277]: model_rfc = RandomForestClassifier(n_estimators=5, min_samples_split=5, max_depth=7, max_features=1)
          classification_model(model_rfc, dataset,predictors,outcome_var)

          Accuracy : 100.000%
          Cross-Validation Score : 95.000%
```

d. SVM (Support Vector Machine)



```
In [278]: from sklearn import svm

In [279]: model_svm=svm.SVC()
          classification_model(model_svm, dataset,predictors,outcome_var)

          Accuracy : 100.000%
          Cross-Validation Score : 91.233%
```

e. KNN (K-Nearest Neighbors)



```
In [285]: from sklearn.neighbors import KNeighborsClassifier

In [288]: knn=KNeighborsClassifier(n_neighbors=11)
          classification_model(knn, dataset,predictors,outcome_var)

          Accuracy : 100.000%
          Cross-Validation Score : 98.219%
```

Chapter 4

Conclusion

A Smart Irrigation and Monitoring System has been proposed so as to reduce wastage of water and to automate the irrigation structure of large areas of crops. The system mainly monitors the behavior of soil moisture, air humidity, and air temperature and see how it contributes to evaluate the needs of water in a plant. The system uses machine learning and compares actual values obtained from sensors with a threshold value that has been fed to the machine learning for analysis. After this process, the machine learning cross checks the result obtained with weather forecast and then decides whether irrigation needs to be done or not. Introducing the cloud computing services reduce delivery time effectively and improves the cost in maintaining the sensors and IT Resources.

References

[1] K. Matsumoto, R. Katsuma, N. Shibata, K. Yasumoto and M. Ito, (2009), “Extended Abstract: Minimizing Localization Cost with Mobile Anchor in Underwater Sensor Networks,” The Fourth ACM International Workshop on Under Water Networks (WUWNet), 2009.

[2] Data Quest