CROP YIELD PREDICTION

SYSTEM

S. SONU PATRA

Regd.No: 1701289273

### Department of Information Technology

TRIDENT ACADEMY OF TECHNOLOGY

Bhubaneswar-751024, Odisha, India.

May 2021

*Project Report on*

CROP YIELD PREDICTION

SYSTEM

*Submitted in Partial Fulfillment of the Requirement for the Award of the Degree of*

### Bachelor of Technology in

Information Technology

*Submitted by*

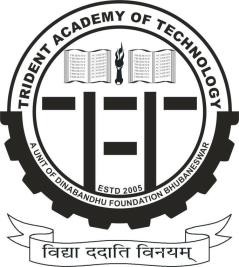
### S. SONU PATRA

Regd.No: 1701289273

*Under the Guidance of*

### Dr. Biswaranjan Nayak

Asst. Professor, Dept. of CSE



### Department of Information Technology

TRIDENT ACADEMY OF TECHNOLOGY

Bhubaneswar-751024, Odisha, India.

May 2021

CERTIFICATE OF APPROVAL

This B.Tech. Viva-Voce Examination of the Major Project work submitted by the candidate **S.Sonu Patra** bearing BPUT Regd.No.: **1701289273** is held during **7th May**, **2021** and is accepted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Information Technology** of **Biju Patnaik University of Technology, Odisha.**

Place: Bhubaneswar

Date: 07-05, 2021 (External Signature)

Place: Bhubaneswar

**Dr. Biswaranjan Nayak**

Date: 07-05, 2021 Dept. of CSE, Project Guide

Place: Bhubaneswar

Date: 07-05, 2021 HOD, Dept. of CSE

i

# DECLARATION

I, **S.Sonu Patra** declare that the Major Project Work presented through this report was carried out by me in accordance with the requirements and in compliance of the Academic Regulations of the Biju Patnaik University of Technology (BAR) for the Bachelor of Technology (B.Tech.) Degree Programed in Information Technology and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is solely my own work. Work done in collaboration with, or with the assistance of, others, has been acknowledged and is indicated as such. Any views expressed in the report are those of the author.

Place: Bhubaneswar **S.Sonu Patra**

Date: 07-05, 2021 Regd.No: **1701289273**

ii

# CERTIFICATE

This is to certify that the report of the Major Project Work on the topic entitled **“CROP YIELD PREDICTION SYSTEM”** which is submitted by **S.Sonu Patra** in partial fulfillment of the requirement for the award of the of **Bachelor of Technology in Information Technology of Biju Patnaik University of Technology, Odisha**, is a bona fide record of the candidate's own work carried out by him under my supervision.

|  |  |
| --- | --- |
| Supervisor | Head of the Department |
| (**Dr. Biswaranjan Nayak**) |  |
| Asst. Professor, Dept. of CSE | Professor & Head, Dept. of IT |
| Trident Academy of  Technology | Trident Academy of Technology |
| Bhubaneswar, India. | Bhubaneswar-751024, Odisha,  India. |

iii

# ABSTRACT

In crop management system we will develop a model along with GUI interface where different inputs will be given as month, year, rainfall and it will predict the WPI. This model will be using Decision tree and some other algorithms.

Place: Bhubaneswar **S.Sonu Patra**

Date: 07-05, 2021 Regd.No: **1701289273**

iv

# ACKNOWLEDGMENTS

I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I am, in the first place, obliged and grateful to my parents without whose support and care I could not have completed this project. I express my deep gratitude towards my guide, Dr. Biswaranjan Nayak, Assistant Professor, Dept. of CSE, Trident Academy of Technology, Bhubaneswar, for his tremendous support, encouragement and help.

I convey my sincere thanks to our HOD, Department of Information Technology and the principal of Trident Academy of Technology, Bhubaneswar, for their permission and cooperation in the completion of the project without experiencing any hurdles. I would like to extend my gratitude to the Department of Information Technology, Trident Academy of Technology, Bhubaneswar, for their support and cooperation.

Finally, I extend my appreciation to all my friends, teaching and non-teaching staffs, who directly or indirectly helped me in this endeavor.

Place: Bhubaneswar S.Sonu Patra

Date: 07-05, 2021

Regd.No: **1701289273**

v

# CONTENTS

[**Approval**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark0) **i**

[**Declaration**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark1) **ii**

[**Certificate**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark2) **iii**

[**Abstract**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark3) **iv**

[**Acknowledgements**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark4) **v**

[**Contents**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark5) **vi**

1. [**The Problem Statement**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark6) **1**
   1. [**About the Problem Statement**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark6)  **1**
2. [**Introduction**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark7) **2**

[2.1](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark7) Definition of Regression 2

**2.1.1 Simple Linear Regression 3**

2.1.2 Multiple linear Regression 4

**2.1.3 Non-Linear Regression 6**

**2.1.4 Overfitting in Regression 7**

2.1.5 Support vector regression 8

2.1.6 Decision Tree regression 12

1. **Python & Related Libraries 16**

[**3.1**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark7)[**About Python**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark7)  **16**

[**3.1.1 Numpy**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_TOC_250001) **17**

[3.1.2 Pandas](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_TOC_250000) 19

**3.1.3 SK Learn 20**

**3.1.4 Matplotlib 21**

**vi**

|  |  |  |  |
| --- | --- | --- | --- |
| **4** | [**EDA for Preprocessing**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark9) | | **22** |
|  | **4.1.1** | [**EDA**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark9) | **22** |
|  | **4.1.2** | **Data Preprocessing** | **24** |
|  | **4.1.3** | **Coding & Results** | **26** |
| [**5 Conclusions & Future Scope**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark10) | | | **33** |
| [**5.1**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark10) | [**Conclusion**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark10) | | **33** |
| [**5.2**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark10) | [**Future Scope**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark10) | | **33** |
| [**References**](file:///C:\Users\LENOVO\Downloads\DEBASISH%20MOHAPATRA(1601289055)-MajorProjectReport-PDF%20(1)-converted.docx#_bookmark11) | |  | **34** |

vii

# CHAPTER 1

The Problem Statement

* As day by day everything is getting automated, but till now farmers are not able to predict how

Much wholesale price for crops they care producing after farming. In India weather condition is not in regular pattern. So how to predict the price on the basis of past date, month and year?

##### 1.1 About the Problem Statement

Here we will use decision tree model along with some other algorithms. A decision tree regression is a non-parametric supervised learning method used for both classification and regression tasks. Many different models can be used, the best is the Decision Tree. It tries to fit data with the best hyperplane which goes through the points and gives less error.

1

# CHAPTER 2

Introduction

* Regression analysis is used when you want to predict a continuous dependent variable from a number of independent variables. If the dependent variable is dichotomous, then logistic regression should be used. (If the split between the two levels of the dependent variable is close to 50-50, then both logistic and linear regression will end up giving you similar results.) The independent variables used in regression can be either continuous or dichotomous. Independent variables with more than two levels can also be used in regression analyses, but they first must be converted into variables that have only two levels. This is called dummy coding and will be discussed later. Usually, regression analysis is used with naturally-occurring variables, as opposed to experimentally manipulated variables, although you can use regression with experimentally manipulated variables.
  1. **Definition Of Regression**
* Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).
* Regression helps investment and financial managers to value assets and understand the relationships between variables, such as [commodity prices](https://www.investopedia.com/terms/c/commodity.asp) and the stocks of businesses dealing in those commodities.
* **2**
* Basically Regression divided into three types, they are :-

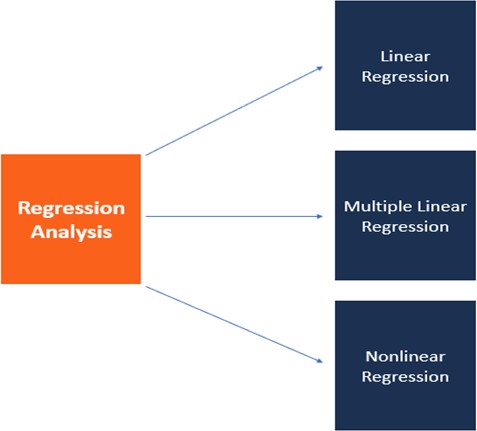


Fig.2.1: Regression Analysis

**2.1.1 Simple Linear Regression**

* Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.
* For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.
* The general form of simple regression is:

 Y = a + bX + u

Where:

* Y = the variable that you are trying to predict (dependent variable).
* X = the variable that you are using to predict Y (independent variable).
* a = the intercept.
* b = the slope.
* u = the regression residual.

**3**

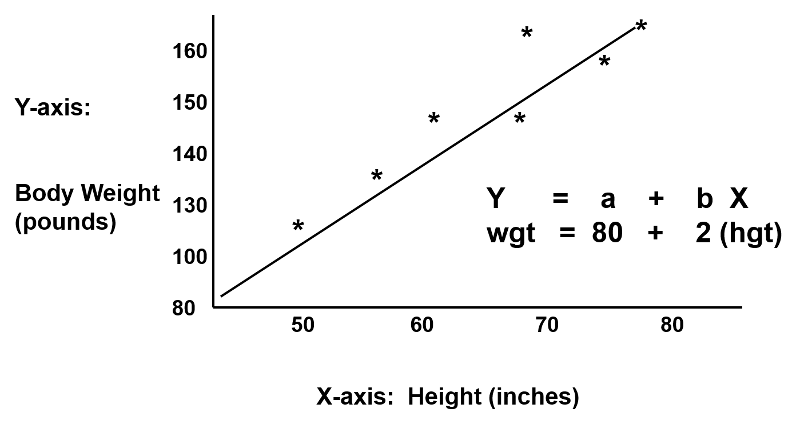


Fig:- 2.1.1 Simple linear regression

**2.1.2 Multi Linear Regression**

* Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
* The multiple regression model is based on the following assumptions:
* There is a [linear relationship](https://www.investopedia.com/terms/l/linearrelationship.asp) between the dependent variables and the independent variables.
* The independent variables are not too highly [correlated](https://www.investopedia.com/terms/c/correlation.asp) with each other.
* yi observations are selected independently and randomly from the population.
* Residuals should be [normally distributed](https://www.investopedia.com/terms/n/normaldistribution.asp) with a mean of 0 and [variance](https://www.investopedia.com/terms/v/variance.asp) *σ.*
* The general form of simple regression is:

Y = a + b1X1+ b2X2 + b3X3 + ... + btXt + u

Where:

* Y = the variable that you are trying to predict (dependent variable).
* X = the variable that you are using to predict Y (independent variable).
* a = the intercept.
* b = the slope.
* u = the regression residual.

**4**

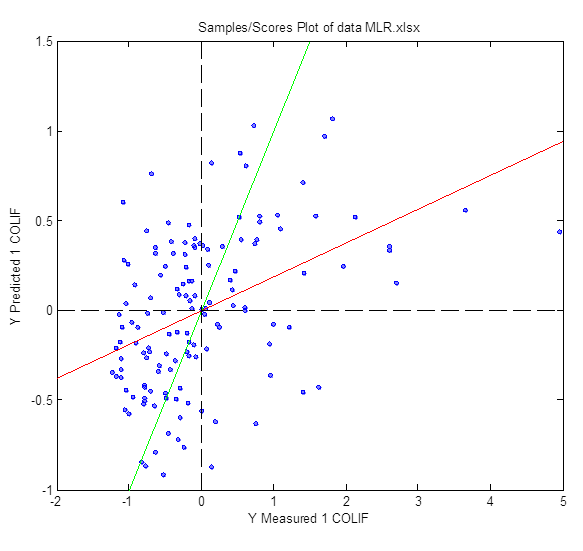


Fig:- 2.1.2 Multi Linear regression

## Example How to Use Multiple Linear Regression (MLR)

* As an example, an analyst may want to know how the movement of the market affects the price of ExxonMobil (XOM). In this case, their linear equation will have the value of the S&P 500 index as the independent variable, or predictor, and the price of XOM as the dependent variable.
* In reality, there are multiple factors that predict the outcome of an event. The price movement of ExxonMobil, for example, depends on more than just the performance of the overall market. Other predictors such as the price of oil, interest rates, and the price movement of oil [futures](https://www.investopedia.com/terms/f/futures.asp) can affect the price of XOM and stock prices of other oil companies. To understand a relationship in which more than two variables are present, multiple linear regression is used.
* Multiple linear regression (MLR) is used to determine a mathematical relationship among a number of random variables. In other terms, MLR examines how multiple independent variables are related to one dependent variable. Once each of the independent factors has been determined to predict the dependent variable, the information on the multiple variables can be used to create an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.
* Referring to the MLR equation above, in our example:
* yi = dependent variable—the price of XOM
* xi1 = interest rates
* xi2= oil price
* xi3= value of S&P 500 index
* xi4= price of oil futures

**5**

* B0 = y-intercept at time zero
* B1 = [regression coefficient](https://www.investopedia.com/terms/r/regression.asp) that measures a unit change in the dependent variable when xi1 changes - the change in XOM price when interest rates change
* B2 = coefficient value that measures a unit change in the dependent variable when xi2 changes—the change in XOM price when oil prices change
* The least-squares estimates, B0, B1, B2…Bp, are usually computed by statistical software. As many variables can be included in the regression model in which each independent variable is differentiated with a number—1,2, 3, 4...p. The multiple regression model allows an analyst to predict an outcome based on information provided on multiple explanatory variables.
* Still, the model is not always perfectly accurate as each data point can differ slightly from the outcome predicted by the model. The residual value, E, which is the difference between the actual outcome and the predicted outcome, is included in the model to account for such slight variations.
* Assuming we run our XOM price regression model through a statistics computation software, that returns this output:

XOM price = 75 – 1.5 interest rates + 7.8 oil price + 3.2 S&P 500 + 5.7 oil futures

R – Sq = 86.5%

* An analyst would interpret this output to mean if other variables are held constant, the price of XOM will increase by 7.8% if the price of oil in the markets increases by 1%. The model also shows that the price of XOM will decrease by 1.5% following a 1% rise in interest rates. R2 indicates that 86.5% of the variations in the stock price of Exxon Mobil can be explained by changes in the interest rate, oil price, oil futures, and S&P 500 index.

**2.1.3 Non-Linear Regression**

* Nonlinear regression is a form of regression analysis in which data is fit to a model and then expressed as a mathematical function.
* The goal of the model is to make the [sum of the squares](https://www.investopedia.com/terms/s/sum-of-squares.asp) as small as possible.  The sum of squares is a measure that tracks how far the Y observations vary from the nonlinear (curved) function that is used to predict Y.
* Nonlinear regression is a curved function of an X variable (or variables) that is used to predict a Y variable
* Nonlinear regression can show a prediction of population growth over time.
* **6**

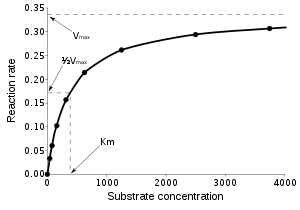


Fig:- 2.1.3

**2.1.4 Overfitting Regression**

* **Overfitting**is a modeling error that occurs when a function or model is too closely fit the training set and getting a drastic difference of fitting in test set. **Overfitting** the model generally takes the form of making an overly complex model to explain Model behavior in the data under study.
* In **regression** analysis, **overfitting** can produce misleading R-squared values, **regression** coefficients, and p-values
* **How to prevent Overfitting?**
* Training with more data
* Data Augmentation
* Cross-Validation
* Feature Selection
* Regularization

**7**

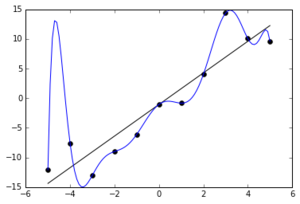


Fig:- 2.1.4 Overfitting Regression

**2.1.5 SupportVector Regression**

## Introduction to Support Vector Regression (SVR)

## Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. Let’s spend a few minutes understanding the idea behind SVR.

* The Idea Behind Support Vector Regression
* The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample. So let’s now dive deep and understand how SVR works actually.

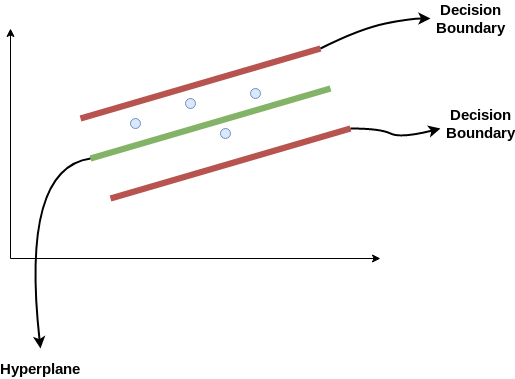
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/SVR1.png)

Fig:- 2.1.5

**8**

* Consider these two red lines as the decision boundary and the green line as the hyperplane. **Our objective, when we are moving on with SVR, is to basically consider the points that are within the decision boundary line.** Our best fit line is the hyperplane that has a maximum number of points.
* The first thing that we’ll understand is what is the decision boundary (the danger red line above!). Consider these lines as being at any distance, say ‘a’, from the hyperplane. So, these are the lines that we draw at distance ‘+a’ and ‘-a’ from the hyperplane. This ‘a’ in the text is basically referred to as epsilon.

Assuming that the equation of the hyperplane is as follows:

Y = wx+b (equation of hyperplane)

Then the equations of decision boundary become:

wx+b= +a

wx+b= -a

Thus, any hyperplane that satisfies our SVR should satisfy:

**-a < Y- wx+b < +a**

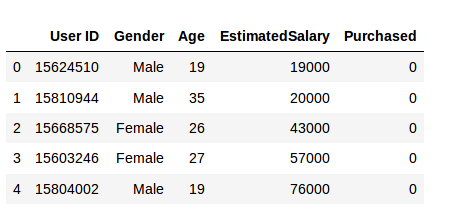
* Our main aim here is to decide a decision boundary at ‘a’ distance from the original hyperplane such that data points closest to the hyperplane or the support vectors are within that boundary line.
* Hence, we are going to take only those points that are within the decision boundary and have the least error rate, or are within the Margin of Tolerance. This gives us a better fitting model.

## Implementing Support Vector Regression (SVR) in Python

## 9

* Time to put on our coding hats! In this section, we’ll understand the use of Support Vector Regression with the help of a dataset. Here, we have to predict the salary of an employee given a few independent variables. A classic HR analytics project!

HR analytics Table:-

Table:- 2.1 [](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/data.png)

* **Step 1: Importing the libraries**

|  |  |
| --- | --- |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import pandas as pd |

* **Step 2: Reading the dataset**

|  |  |
| --- | --- |
|  | dataset = pd.read\_csv('Position\_Salaries.csv') |
|  | X = dataset.iloc[:, 1:2].values |
|  | y = dataset.iloc[:, 2].values |

* Step 3: Feature Scaling
* A real-world dataset contains features that vary in magnitudes, units, and range. I would suggest performing normalization when the scale of a feature is irrelevant or misleading.
* Feature Scaling basically helps to normalize the data within a particular range. Normally several common class types contain the feature scaling function so that they make feature scaling automatically. However, the SVR class is not a commonly used class type so we should perform feature scaling using Python.

|  |
| --- |
|  |
|  |
|  |

1**0**

|  |  |
| --- | --- |
|  | from sklearn.preprocessing import StandardScaler |
|  | sc\_X = StandardScaler() |
|  | sc\_y = StandardScaler() |
|  | y = sc\_y.fit\_transform(y) |

* Step 4: Fitting SVR to the dataset

|  |  |
| --- | --- |
|  | from sklearn.svm import SVR |
|  | regressor = SVR(kernel = 'rbf') |
|  | regressor.fit(X, y) |

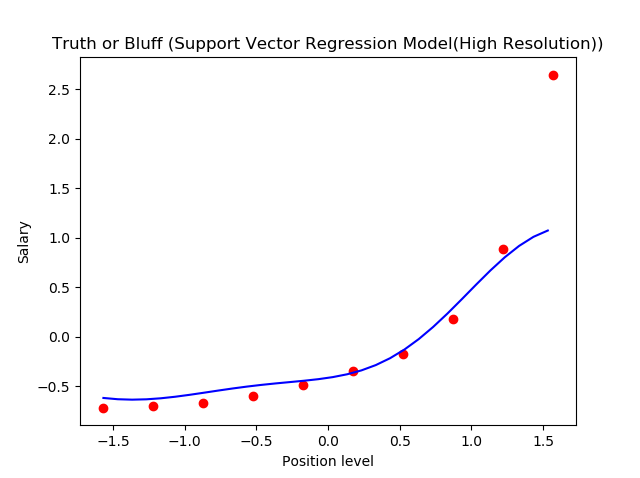
* Kernel is the most important feature. There are many types of kernels – linear, Gaussian, etc. Each is used depending on the dataset.
* Step 5. Predicting a new result

|  |  |
| --- | --- |
|  | y\_pred = regressor.predict(6.5) |
|  | y\_pred = sc\_y.inverse\_transform(y\_pred) |

So, the prediction for y\_pred(6, 5) will be 170,370.

* Step 6. Visualizing the SVR results (for higher resolution and smoother curve)

|  |  |
| --- | --- |
|  | X\_grid = np.arange(min(X), max(X), 0.01) #this step required because data is feature scaled. |
|  | X\_grid = X\_grid.reshape((len(X\_grid), 1)) |
|  | plt.scatter(X, y, color = 'red') |
|  | plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue') |
|  | plt.title('Truth or Bluff (SVR)') |
|  | plt.xlabel('Position level') |
|  | plt.ylabel('Salary') |
|  | plt.show() |

Fig:- 2.1.6[](https://cdn.analyticsvidhya.com/wp-content/uploads/2020/03/graph.png)

**11**

* This is what we get as output- the best fit line that has a maximum number of points. Quite accurate!

**2.1.6 Decision Tree Regression**

## Overview of Decision Tree Algorithm

* Decision Tree is one of the most commonly used, practical approaches for supervised learning. It can be used to solve both Regression and Classification tasks with the latter being put more into practical application.
* It is a tree-structured classifier with three types of nodes. The **Root Node**is the initial node which represents the entire sample and may get split further into further nodes. The **Interior Nodes**represent the features of a data set and the branches represent the decision rules. Finally, the **Leaf Nodes** represent the outcome. This algorithm is very useful for solving decision-related problems.

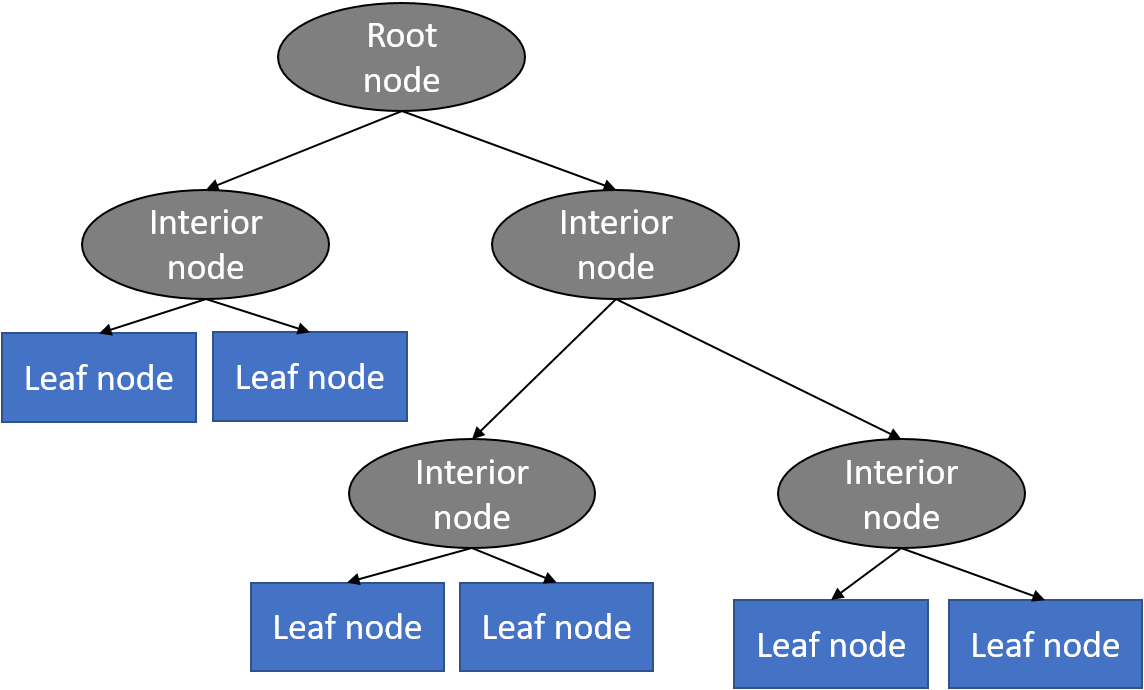


Fig:- 2.1.7

**12**

* With a particular data point, it is run completely through the entirely tree by answering True/False questions till it reaches the leaf node. The final prediction is the average of the value of the dependent variable in that particular leaf node. Through multiple iterations, the Tree is able to predict a proper value for the data point.

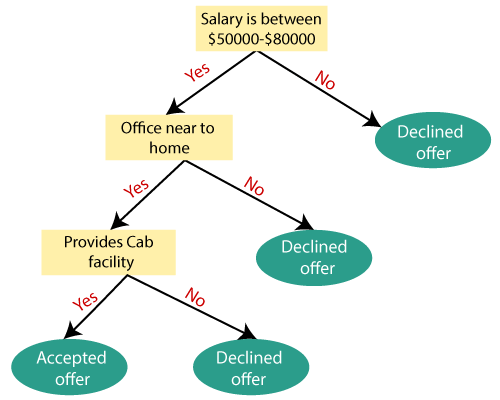


Fig:- 2.1.8 An example for Decision Tree Model

* The above diagram is a representation for the implementation of a Decision Tree algorithm. Decision trees have an advantage that it is easy to understand, lesser data cleaning is required, non-linearity does not affect the model’s performance and the number of hyper-parameters to be tuned is almost null. However, it may have an over-fitting problem, which can be resolved using the **Random Forest**algorithm which will be explained in the next article.
* In this example, we will go through the implementation of **Decision Tree Regression**, in which we will predict the revenue of an ice cream shop based on the temperature in an area for 500 days.

## Problem Analysis

* In this data, we have one independent variable Temperature and one independent variable Revenue which we have to predict. In this problem, we have to build a Decision Tree Regression Model which will study the correlation between the Temperature and Revenue of the Ice

**13**

* Cream Shop and predict the revenue for the ice cream shop based on the temperature on a particular day.

## Step 1: Importing the libraries

* The first step will always consist of importing the libraries that are needed to develop the ML model. The **NumPy**, **matplotlib**and the**Pandas libraries**are imported.

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd

## Step 2: Importing the dataset

* In this step, we shall use pandas to store the data obtained from my github repository and store it as a Pandas DataFrame using the function ‘pd.read\_csv’.In this, we assign the independent variable (X) to the ‘Temperature’ column and the dependent variable (y) to the ‘Revenue’ column.

dataset = pd.read\_csv('<https://raw.githubusercontent.com/mk-gurucharan/Regression/master/IceCreamData.csv'>)

X = dataset['Temperature'].values  
y = dataset['Revenue'].valuesdataset.head(5)>>Temperature Revenue  
24.566884 534.799028  
26.005191 625.190122  
27.790554 660.632289  
20.595335 487.706960  
11.503498 316.240194

## Step 3: Splitting the dataset into the Training set and Test set

* In the next step, we have to split the dataset as usual into the training set and the test set. For this we use test\_size=0.05 which means that 5% of 500 data rows (25 rows) will only be used as test set and the remaining 475 rows will be used as training set for building the model.

from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.05)

## 14

## Step 4: Training the Decision Tree Regression model on the training set

* We import the DecisionTreeRegressor class from sklearn.tree and assign it to the variable ‘**regressor’**. Then we fit the X\_train and the y\_train to the model by using theregressor.fit function. We use the reshape(-1,1) to reshape our variables to a single column vector.

# Fitting Decision Tree Regression to the dataset  
from sklearn.tree import DecisionTreeRegressor  
regressor = DecisionTreeRegressor()  
regressor.fit(X\_train.reshape(-1,1), y\_train.reshape(-1,1))

## Step 5: Predicting the Results

* In this step, we predict the results of the test set with the model trained on the training set values using the regressor.predict function and assign it to ‘**y\_pred’.**

y\_pred = regressor.predict(X\_test.reshape(-1,1))

## Step 6: Comparing the Real Values with Predicted Values

* In this step, we shall compare and display the values of y\_test as ‘**Real Values**’ and y\_pred as ‘**Predicted Values**’ in a Pandas dataframe.

df = pd.DataFrame({'Real Values':y\_test.reshape(-1), 'Predicted Values':y\_pred.reshape(-1)})  
df>>  
Real Values Predicted Values  
448.325981 425.265596  
535.866729 500.065779  
264.123914 237.763911  
691.855484 698.971806  
587.221246 571.434257  
653.986736 633.504009  
538.179684 530.748225  
643.944327 660.632289  
771.789537 797.566536  
644.488633 654.197406  
192.341996 223.435016  
491.430500 477.295054  
781.983795 807.541287  
432.819795 420.966453  
623.598861 612.803770  
599.364914 534.799028  
856.303304 850.246982  
583.084449 596.236690  
521.775445 503.084268  
228.901030 258.286810  
453.785607 473.568112  
406.516091 450.473207  
562.792463 634.121978  
642.349814 621.189730  
737.800824 733.215828

* From the above values, we infer that the model is able to predict the values of the y\_test with a good accuracy.

## Step 7: Visualising the Decision Tree Regression Results

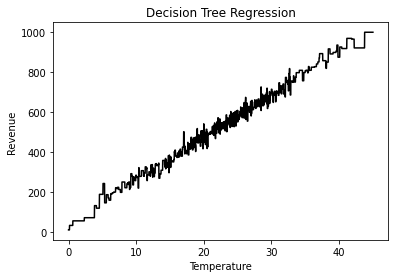
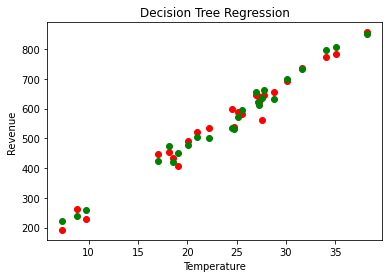


Fig:-2.1.9

**15**

# CHAPTER 3

Python and Related Libraries

Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Machine Learning are:

* NumPy
* Pandas
* Scikit-learn
* Matplotlib

## About Python

Python is one of the most popular and widely used programming languages and has replaced many programming languages in the industry.

There are a lot of reasons why Python is popular among developers and one of them is that it has an amazingly large collection of libraries that users can work with.

Here are a few important reasons as to why Python is popular:

* Python has a huge collection of libraries.
* Python is known as the beginner’s level programming language because of it simplicity and easiness.
* From developing to deploying and maintaining Python wants their developers to be more productive.
* Portability is another reason for huge popularity of Python.
* Python’s programming syntax is simple to learn and is of high level compared to C, Java, and C++.Hence, new applications can be developed by writing fewer lines of codes

****

The simplicity of Python has attracted many developers to create new libraries for machine learning. Because of the huge collection of libraries Python is becoming hugely popular among machine learning experts.

## NumPy

* NumPy is considered as one of the most popular machine learning library in Python.
* TensorFlow and other libraries uses NumPy internally for performing multiple operations on Tensors. Array interface is the best and the most important feature of NumPy.
* NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
* It is the fundamental package for scientific computing with Python. It contains various features including these important ones:
* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

#### Features Of NumPy

1. **Interactive:** NumPy is very interactive and easy to use.
2. **Mathematics:** Makes complex mathematical implementations very simple.
3. **Intuitive:** Makes coding real easy and grasping the concepts is easy.
4. **Lot of Interaction:** Widely used, hence a lot of open source contribution.

#### Where Is Numpy Used?

* This interface can be utilized for expressing images, sound waves, and other binary raw streams as an array of real numbers in N-dimensional.
* For implementing this library for machine learning having knowledge of Numpy is important for full stack developers.

**17**

* NumPy is an open-source numerical Python library.
* NumPy contains a multi-dimensional array and matrix data structures.
* It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines. Therefore, the library contains a large number of mathematical, algebraic, and transformation functions.
* NumPy is an extension of Numeric and Numarray.
* Numpy also contains random number generators.
* NumPy is a wrapper around a library implemented in C.

# **What Are The Most Important Numpy Data Types?**

* Let’s start by understanding the most important Numpy data types. There are a large number of NumPy objects available:

## One Dimensional Array

* One of the most important objects is an N-dimensional array type known as ndarray.
* We can think of a one-dimensional array as a column or a row of a table with one or more elements:

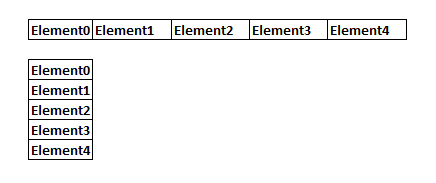


Fig:-3.1

* All of the items that are stored in ndarray are required to be of the same type. This implies that the
* **18**

ndarray is a block of homogeneous data. ndarray has striding information. This numerical value is the number of bytes of the next element in a dimension.

* This helps the array to navigate through memory and does not require copying the data.
* Each ndarray contains a pointer that points to its memory location in the computer. It also contains its dtype, its shape, and tuples of strides. The strides are integers indicating the number of bytes it has to move to reach the next element in a dimension.

## 3.1.2 Pandas

* Pandas is a machine learning library in Python that provides data structures of high-level and a wide variety of tools for analysis. One of the great feature of this library is the ability to translate complex operations with data using one or two commands. Pandas have so many inbuilt methods for grouping, combining data, and filtering, as well as time-series functionality.

#### Features Of Pandas

* Pandas make sure that the entire process of manipulating data will be easier. Support for operations such as Re-indexing, Iteration, Sorting, Aggregations, Concatenations and Visualizations are among the feature highlights of Pandas.

#### Where Is Pandas Used?

* Currently, there are fewer releases of pandas library which includes hundred of new features, bug fixes, enhancements, and changes in API. The improvements in pandas regards its ability to group and sort data, select best suited output for the apply method, and provides support for performing custom types operations.
* Data Analysis among everything else takes the highlight when it comes to usage of Pandas. But, Pandas when used with other libraries and tools ensure high functionality and good amount of flexibility.

**19**

#### Scikit-Learn

* It is a Python library is associated with NumPy and SciPy. It is considered as one of the best libraries for working with complex data.
* There are a lot of changes being made in this library. One modification is the cross-validation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression and nearest neighbours have received some little improvements.

#### Features Of Scikit-Learn

1. **Cross-validation:** There are various methods to check the accuracy of supervised models on unseen data.

1. **Unsupervised learning algorithms**: Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.
2. **Feature extraction:** Useful for extracting features from images and text (e.g. Bag of words)

#### Where Is Scikit-Learn Used

* It contains a numerous number of algorithms for implementing standard machine learning and data mining tasks like reducing dimensionality, classification, regression, clustering, and model selection.

**20**

#### Matplotlib

* matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.
* There are several toolkits which are available that extend python matplotlib functionality. Some of them are separate downloads, others can be shipped with the matplotlib source code but have external dependencies.
* Basemap: It is a map plotting toolkit with various map projections, coastlines and political boundaries.
* Cartopy: It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.
* Excel tools: Matplotlib provides utilities for exchanging data with Microsoft Excel.
* Mplot3d: It is used for 3-D plots.
* Natgrid: It is an interface to the natgrid library for irregular gridding of the spaced data.
* Some features of Python Plot supports-

1. Font properties
2. Axes properties
3. Line styles

#### tkinter

* Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
* Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

**21**

* + Import the *Tkinter* module.
  + Create the GUI application main window.
  + Add one or more of the above-mentioned widgets to the GUI application.
  + Enter the main event loop to take action against each event triggered by the user.

## Example

## #!/usr/bin/python

## import Tkinter

## top = Tkinter.Tk()

## # Code to add widgets will go here...

## top.mainloop()

## Tkinter Widgets

* Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
* There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

|  |  |
| --- | --- |
| **Sr.No.** | **Operator & Description** |
| 1 | [Button](https://www.tutorialspoint.com/python/tk_button.htm): - The Button widget is used to display buttons in your application. |
| 2 | [Canvas](https://www.tutorialspoint.com/python/tk_canvas.htm): - The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application. |
| 3 | [Checkbutton](https://www.tutorialspoint.com/python/tk_checkbutton.htm): - The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time. |
| 4 | [Entry](https://www.tutorialspoint.com/python/tk_entry.htm): - The Entry widget is used to display a single-line text field for accepting values from a user. |
| 5 | [Frame](https://www.tutorialspoint.com/python/tk_frame.htm): - The Frame widget is used as a container widget to organize other widgets. |
| 6 | [Label](https://www.tutorialspoint.com/python/tk_label.htm): - The Label widget is used to provide a single-line caption for other widgets. It can also contain images. |
| 7 | [Listbox](https://www.tutorialspoint.com/python/tk_listbox.htm): - The Listbox widget is used to provide a list of options to a user. |
| 8 | [Menubutton](https://www.tutorialspoint.com/python/tk_menubutton.htm): - The Menubutton widget is used to display menus in your application. |
| 9 | [Menu](https://www.tutorialspoint.com/python/tk_menu.htm): - The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. |
| 10 | [Message](https://www.tutorialspoint.com/python/tk_message.htm): - The Message widget is used to display multiline text fields for accepting values from a user. |

**22**

|  |  |
| --- | --- |
| 11 | [Radiobutton](https://www.tutorialspoint.com/python/tk_radiobutton.htm): - The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time. |
| 12 | [Scale](https://www.tutorialspoint.com/python/tk_scale.htm): - The Scale widget is used to provide a slider widget. |
| 13 | [Scrollbar](https://www.tutorialspoint.com/python/tk_scrollbar.htm): - The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes. |
| 14 | [Text](https://www.tutorialspoint.com/python/tk_text.htm): - The Text widget is used to display text in multiple lines. |
| 15 | [Toplevel](https://www.tutorialspoint.com/python/tk_toplevel.htm): - The Toplevel widget is used to provide a separate window container. |

## Standard attributes

They are as bellows: -

* [Dimensions](https://www.tutorialspoint.com/python/tk_dimensions.htm)
* [Colors](https://www.tutorialspoint.com/python/tk_colors.htm)
* [Fonts](https://www.tutorialspoint.com/python/tk_fonts.htm)
* [Anchors](https://www.tutorialspoint.com/python/tk_anchors.htm)
* [Relief styles](https://www.tutorialspoint.com/python/tk_relief.htm)
* [Bitmaps](https://www.tutorialspoint.com/python/tk_bitmaps.htm)
* [Cursors](https://www.tutorialspoint.com/python/tk_cursors.htm)

**23**

# CHAPTER 4

EDA and Data Preprocessing

#### EDA

* is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. Following are the different steps involved in EDA :
  + - * Data Collection
      * Data Cleaning
      * Data Preprocessing
      * Data Visualization

#### Data Collection

* Data collection is the process of gathering information in an established systematic way that enables one to test hypothesis and evaluate outcomes easily.

#### Data Cleaning

* Data cleaning is the process of ensuring that your data is correct and useable by identifying any errors in the data, or missing data by correcting or deleting them. Refer to this link for data cleaning.
* Once the data is clean we can go further for data preprocessing.

#### Data Preprocessing

* Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. It includes normalization and standardization, transformation, feature extraction and selection, etc. The product of data preprocessing is the final training dataset. Exploratory Data Analysis (EDA) is an

**24**

* approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) to maximize insight into a data set;
* uncover underlying structure
* extract important variables;
* detect outliers and anomalies;
* test underlying assumptions;
* develop parsimonious models; and
* determine optimal factor settings.
* Focus: The EDA approach is precisely that--an approach--not a set of techniques, but an attitude/philosophy about how a data analysis should be carried out.
* Philosophy: EDA is not identical to statistical graphics although the two terms are used almost interchangeably. Statistical graphics is a collection of techniques--all graphically based and all focusing on one data characterization aspect. EDA encompasses a larger venue; EDA is an approach to data analysis that postpones the usual assumptions about what kind of model the data follow with the more direct approach of allowing the data itself to reveal its underlying structure and model. EDA is not a mere collection of techniques; EDA is a philosophy as to how we dissect a data set; what we look for; how we look; and how we interpret. It is true that EDA heavily uses the collection of techniques that we call "statistical graphics", but it is not identical to statistical graphics.
* History: The seminal work in EDA is Exploratory Data Analysis, Tukey, (1977). Over the years it has benefitted from other noteworthy publications such as Data Analysis and Regression, Mosteller and Tukey (1977), Interactive Data Analysis, Hoaglin (1977), The ABC's of EDA, Velleman and Hoaglin (1981) and has gained a large following as "the" way to analyze a data set.
* Techniques: Most EDA techniques are graphical in nature with a few quantitative

**25**

* techniques. The reason for the heavy reliance on graphics is that by its very nature the main role of EDA is to open-mindedly explore, and graphics gives the analysts unparalleled power to do so, enticing the data to reveal its structural secrets, and being always ready to gain some new, often unsuspected, insight into the data. In combination with the natural pattern- recognition capabilities that we all possess, graphics provides, of course, unparalleled power to carry this out.
* The particular graphical techniques employed in EDA are often quite simple, consisting of various techniques of:
* Plotting the raw data (such as data traces, histograms, bihistograms, probability plots, lag plots, block plots, and Youden plots.
* Plotting simple statistics such as mean plots, standard deviation plots, box plots, and main effects plots of the raw data.
* Positioning such plots so as to maximize our natural pattern-recognition abilities, such as using multiple plots per page.

# Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data preprocessing is used database-driven applications such as customer relationship management and rule-based applications (like neural networks).

Data goes through a series of steps during preprocessing:

* Data Cleaning: Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data.
* Data Integration: Data with different representations are put together and conflicts within the data are resolved.
* Data Transformation: Data is normalized, aggregated and generalized.

**26**

* Data Reduction: This step aims to present a reduced representation of the data in a data warehouse.
* Data Discretization: Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.

**Code for Data Preprocessing**

#importing libraries

import numpy as np

import pandas as pd

#import datasets

dataset=pd.read\_csv('Arhar.csv')

print(dataset)

X = dataset.iloc[:, :3].values

    Y = dataset.iloc[:, 3].values

print(x)

#missing values

from sklearn.preprocessing import Imputer

imputer=Imputer(missing\_values='NaN',strategy='mean',axis=0)

imputer=imputer.fit(x[:,:3])

x[:,:3]=imputer.transform(x[:,:3])

print(x)

**27**

# Coding & Results:

## 5.1.1 Code:

**#Importing Libraries**

import tkinter

from tkinter import \*

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import random

from datetime import datetime

from sklearn.tree import DecisionTreeRegressor

from sklearn.svm import SVR

import tkinter.messagebox as m

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

from tkinter import ttk

from PIL import ImageTk,Image

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

**#decision tree regression function**

def DecisionTreeRegression():

data=n.get()

dataset = pd.read\_csv(data+".csv")

X = dataset.iloc[:,0:3].values

Y = dataset.iloc[:,3].values

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.1, random\_state=0)

a1=float(c1.get())

a2=float(c2.get())

a3=float(c3.get())

global DecisionTree

depth = random.randrange(7,18)

DecisionTree = DecisionTreeRegressor(max\_depth=depth)

DecisionTree.fit(X\_train,Y\_train)

y\_pred\_tree = DecisionTree.predict(X\_test)

val=DecisionPredictedValue([float(a1), a2, a3],X,Y)

m.showinfo(title="Whole Sale Price Index",message=val)

DecisionRegressorScore(X\_train,Y\_train)

**#support vector regression function**

def SupportVectorRegression():

data=n.get()

dataset = pd.read\_csv(data+".csv")

X = dataset.iloc[:,0:3].values

Y = dataset.iloc[:, 3].values

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.1, random\_state=0)

a1=float(c1.get())

a2=float(c2.get())

**28**

a3=float(c3.get())

global SupportVector

sc\_X=StandardScaler()

sc\_Y=StandardScaler()

Y\_train1=Y\_train.reshape(-1,1)

X\_train1=sc\_X.fit\_transform(X\_train)

Y\_train1=sc\_Y.fit\_transform(Y\_train1)

SupportVector=SVR(kernel='rbf')

SupportVector.fit(X\_train1,Y\_train1.ravel())

y\_pred=sc\_Y.inverse\_transform(SupportVector.predict(sc\_X.transform(np.array([[float(a1), a2, a3]]))))

support\_val=SupportPredictedValue([float(a1), a2, a3],X,Y)

m.showinfo(title="Whole Sale Price Index",message=support\_val)

SupportRegressorScore(X\_train,Y\_train)

**#multiple linear regression function**

def MultipleLinearRegression():

data=n.get()

dataset = pd.read\_csv(data+".csv")

X = dataset.iloc[:,0:3].values

Y = dataset.iloc[:, 3].values

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.1, random\_state=0)

a1=float(c1.get())

a2=float(c2.get())

a3=float(c3.get())

global MultipleLinear

MultipleLinear=LinearRegression()

MultipleLinear.fit(X\_train,Y\_train)

y\_pred=MultipleLinear.predict([[float(a1), a2, a3]])

multiple\_val=MultiplePredictedValue([float(a1), a2, a3],X,Y)

m.showinfo(title="Whole Sale Price Index",message=multiple\_val)

MultipleRegressorScore(X\_train,Y\_train)

**#these function are used to predict the output of different models**

def DecisionPredictedValue(value,X,Y):

if value[1]>=2019:

fsa = np.array(value).reshape(1, 3)

return DecisionTree.predict(fsa)[0]

else:

c=X[:,0:2]

x=[]

for i in c:

x.append(i.tolist())

fsa = [value[0], value[1]]

ind = 0

for i in range(0,len(x)):

if x[i]==fsa:

ind=i

break

return Y[i]

def SupportPredictedValue(value,X,Y):

if value[1]>=2019:

fsa = np.array(value).reshape(1, 3)

**29**

return SupportVector.predict(fsa)[0]

else:

c=X[:,0:2]

x=[]

for i in c:

x.append(i.tolist())

fsa = [value[0], value[1]]

ind = 0

for i in range(0,len(x)):

if x[i]==fsa:

ind=i

break

return Y[i]

def MultiplePredictedValue(value,X,Y):

if value[1]>=2019:

fsa = np.array(value).reshape(1, 3)

return MultipleLinear.predict(fsa)[0]

else:

c=X[:,0:2]

x=[]

for i in c:

x.append(i.tolist())

fsa = [value[0], value[1]]

ind = 0

for i in range(0,len(x)):

if x[i]==fsa:

ind=i

break

return Y[i]

**#this function used to check model and call that particular function**

def getModel():

value=model.get()

if(value=='DecisionTreeRegression'):

DecisionTreeRegression()

elif(value=='SupportVectorRegression'):

SupportVectorRegression()

elif(value=='MultipleLinearRegression'):

MultipleLinearRegression()

**#these function will calculate the regressor score of different models**

def DecisionRegressorScore(X\_train,Y\_train):

error\_val=DecisionTree.score(X\_train,Y\_train)

m.showinfo(title="Regressor score for Decision Tree",message=error\_val)

def MultipleRegressorScore(X\_train,Y\_train):

error\_val=MultipleLinear.score(X\_train,Y\_train)

m.showinfo(title="Regressor score for Multiple Regression",message=error\_val)

def SupportRegressorScore(X\_train,Y\_train):

error\_val=SupportVector.score(X\_train,Y\_train)

m.showinfo(title="Regressor score for Support Vector",message=error\_val)

**#this function will reset the entry field values**

def reset():

**30**

c1.set("")

c2.set("")

c3.set("")

n.set("")

models.set("")

**#these codes are for GUI part**

w=Tk()

# Create a photoimage object of the image in the path

image1 = Image.open('C:\\Users\\dell\\Desktop\\ML Project\\Agritech-Startup.jpg')

image1 = image1.resize((1532, 836), Image.ANTIALIAS)

test = ImageTk.PhotoImage(image1)

label1 = tkinter.Label(image=test)

label1.image = test

# Position image

label1.place(x=0, y=0)

c1=StringVar()

c2=StringVar()

c3=StringVar()

n=StringVar()

models=StringVar()

L1=Label(w,bg="lightgreen",fg="white",width=40,text="Crop Yield Prediction System",font=("times new roman",20,"bold"))

L2=Label(w,bg="black",fg="white",width=18,text="Output and Regressor",font=("times new roman",20,"bold"))

L3=Label(w,bg="black",fg="white",width=18,text="Enter Input To Predict",font=("times new roman",20,"bold"))

L4=Label(w, width = 25,text="Enter Month",font=("monospace",15,"italic"))

L5=Label(w, width = 25,text="Enter Year",font=("monospace",15,"italic"))

L6=Label(w, width = 25,text="Enter Rainfall",font=("monospace",15,"italic"))

E1= ttk.Combobox(w, width = 50,textvariable = c1)

E1['values'] =('1','2','3','4','5','6','7','8','9','10','11','12')

E1.current()

E2=ttk.Combobox(w,width=50,textvariable=c2)

E2['values'] =('1991','1992','1993','1994','1995','1996','1997','1998','1999','2000','2001','2002','2003','2004','2005','2006','2007','2008','2009','2010','2011','2012','2013','2014','2015','2016','2017','2018','2019','2020','2021','2022','2023','2024','2025','2026','2027','2028','2029','2030')

E2.current()

E3=ttk.Combobox(w,width=50,textvariable=c3)

E3['values'] =('47.5','31.7','117.8','250.2','262.4','193.5','58.7','30.7','11.7','11.3','40.1','15.7','30.4','57.8','219.8','310','254.7','152.7','129.4','14','6.7','19.2','27.4','36.1','22.2','72.9','95.4','261.2','237.5','188','60.2','14.4','10.7','17.4','21','62','69.4','53.8','192.8','242.4','205.2','131.8','42.9','39.4','15','19.3','22.8','27.6','37.4','62.6','168.3','289.1','256.3','171.9','76.1','29.7','14.9','19','22.6','27.8','37.9','62.4','168.1','289.3','256.2','171.5','76.3','29.9','15','19.2','20.1','27.1','40.2','62.7','165','286.3','254.3','171.9','77.2','28.1','14.1')

E3.current()

B1=Button(w,bd=3,width=30,bg="grey",fg="white",text="Output and Regressor",font=("times new roman",17),command=getModel)

**31**

B2=Button(w,width=15,bg="grey",fg="white",text="Reset",font=("times new roman",17),command=reset)

L7=Label(w, text = "Choose The Crop Name",width=18,background = 'black', foreground ="white",font = ("Times New Roman", 20,"bold"))

L8=Label(w, width = 25, text = "Select the Crop",font=("monospace",15,"italic"))

L9=Label(w, width = 25, text = "Select the Model",font=("monospace",15,"italic"))

crop = ttk.Combobox(w, width = 50, textvariable = n)

crop['values'] = ('Arhar','Bajra','Barley','Copra','Cotton','Gram','Groundnut','Jowar','Jute','Maize','Masoor','Moong','Niger','Paddy','Ragi','Rape','Safflower','Sesamum','Soyabean','Sugarcane','Sunflower','Urad','Wheat')

crop.grid(row=7,column=1,columnspan=1, pady=20)

crop.current()

model = ttk.Combobox(w, width = 50, textvariable = models)

model['values'] = ('DecisionTreeRegression','SupportVectorRegression','MultipleLinearRegression')

model.grid(row=9,column=1,columnspan=1, pady=20)

model.current()

L1.grid(row=1,column=2, padx=(100,10), pady=20)

L2.grid(row=10,column=1,columnspan=1, pady=20)

L3.grid(row=2,column=1,columnspan=1, pady=20)

L4.grid(row=3,column=1, padx=(15,10), pady=20)

L5.grid(row=3,column=2, padx=(70,10),pady=20)

L6.grid(row=3,column=3,padx=(70,10), pady=20)

E1.grid(row=4,column=1,padx=(15,10), pady=20)

E2.grid(row=4,column=2,padx=(70,10) ,pady=20)

E3.grid(row=4,column=3,padx=(70,10), pady=20)

B1.grid(row=11,column=2,padx=(100,10), pady=70)

B2.grid(row=11,column=3,padx=(100,10), pady=70)

L7.grid(row = 5,column=1,columnspan=1, pady=20)

L8.grid(row=6,column=1,columnspan=1, pady=20)

L9.grid(row=8,column=1,columnspan=1, pady=20)

w.mainloop()

**32**

# CHAPTER 5

Conclusions & Future Scope

## Conclusion

## We conclude that through this model we can predict that at how much price the farmer can sell the crops which will make him/her easy to get benefits. Also using this Decision tree these are one of the most widely used machine learning models, due to the fact that they work well with noisy or missing data and can easily be ensembled to form more robust predictors.

## Future Scope

## This has so many scope in the field of agriculture. Each and every farmers can predicts their different crops price so that they can prepare to produce crops on different months.

**33**

# REFERENCES

* <https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda>
* <https://www.researchgate.net/publication/26904238_Crop_Management_Factors_What_is_Important>
* <https://medium.com/pursuitnotes/decision-tree-regression-in-6-steps-with-python-1a1c5aa2ee16>

**34**