**PRACTICAL NO – 5**

**TITLE-** **Random Forest Regression**

**THEORY- Random** forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

**PROBLEM STATEMENT**- Building Random ForestModel.

**ALGORITHM-**

**1.Importing the libraries.**

**2.Import the datasets.**

**3. Fitting Random Forest regression to the dataset**

**4. Predicting new result with predictor.**

**5. Visualizing the Random Forest regression model.**

**6.End**

**CODE-**

# -\*- coding: utf-8 -\*-

"""

Created on Wed Feb 27 14:39:20 2019

@author: RoKu

"""

# Random Forest Regression

# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('Dataset\_flwr.csv')

X = dataset.iloc[:, 0:1].values

y = dataset.iloc[:, 1:].values

# Splitting the dataset into the Training set and Test set

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc\_X = StandardScaler()

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

X\_test = sc\_X.transform(X\_test)

# Fitting the Random Forest Regression to the dataset

from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators = 300, random\_state = 0)

regressor.fit(X, y)

# Predicting a new result

y\_pred = regressor.predict(y\_test)

# Visualising the Regression results (for higher resolution and smoother curve)

X\_grid = np.arange(min(X), max(X), 0.01)

X\_grid = X\_grid.reshape((len(X\_grid), 1))

plt.scatter(X, y, color = 'red')

plt.plot(X\_grid, regressor.predict(X\_grid), color = 'green')

plt.title('Sepal length Predictor')

plt.xlabel = ('Petal Length')

plt.ylabel = ('Species')

plt.show()

**RESULTS**-

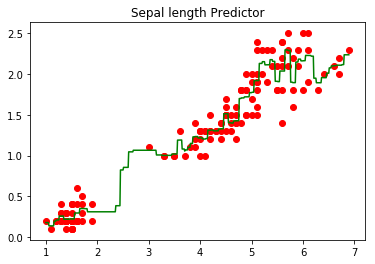


Fig.1- Random Forest Model of Sepal length predictor.

**DISCUSSIONS-**

**Standard Deviation:** A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

**Steps:**

1. Pick at random K data points from training set.
2. Build the decision tree associated with K data points.
3. Choose the number N tree of trees you want to build and repeat Steps 1 & 2.
4. For a new data point, make each one of your N trees predict the value of Y for the data point in question and assign the new point the average across all of the predicted Y values.

**CONCLUSIONS-**  So, in a way we are obtaining the prediction by not just a single tree but by a number of trees. This improves the accuracy of prediction, because you are taking the average of all the predictions of the individual trees from the forest (Ensemble Learning).

|  |  |
| --- | --- |
| **SUBMISSION DATE**-  15/03/2019 | **SIGN OF COURSE INSTRUCTOR**- |
| **ROLL NO OF THE STUDENT-**  **I-01** | **NAME OF THE STUDENT-**  **ROHIT KULKARNI** |