**MINOR-2**
**PROJECT**
**Report**
**on**

# COMPARISION OF MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEVELOPMENT TIME PREDICTION ON HISTORICAL SOFTWARE PROJECT MANAGEMENT RECORDS

Submitted By:

| Name | Roll No | Branch | Sap-id |
|---|---|---|---|
| Apaar Sharma | R110216032 | B.tech CSE CCVT | 500052272 |
| Devanshu Dwivedi | R110216059 | B.tech CSE CCVT | 500053470 |

**Under the guidance of**
**Amar Shukla**
**Assistant Professor (SS)**
**Department of Computer Science**

School of Computer Science
**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**
**Dehradun-248007, 2019**

**Approved By**

(Mr. Amar Shukla)
**Project Guide**

(Dr. Deepshikha Bhargava)
**Department Head**

# DECLARATION

I hereby declare that the project entitled "COMPARISION OF MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEVELOPMENT TIME PREDICTION ON HISTORICAL SOFTWARE PROJECT MANAGEMENT RECORDS" is a Bonafide and authentic record of work done by our team under the supervision of Mr. Amar Shukla during academic session 2018-2019. The work presented here is not copied from any external website or source and is also not registered earlier for any degree/diploma to any other educational institution. I understand that any such unfair means is liable to be punished in accordance with UPES rules and regulations.

Place: University of Petroleum and Energy Studies, Dehradun

**Department of Virtualization**

**University of Petroleum and Energy Studies**

# Certificate

It is to certify that the works contained in the project titled "COMPARISION OF MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEVELOPMENT TIME PREDICTION ON HISTORICAL SOFTWARE PROJECT MANAGEMENT RECORDS" by following students has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree or certification programme.

| Student name | Roll number | signature |
|---|---|---|
| Apaar Sharma | R110216032 | |
| Devanshu Dwivedi | R110216059 | |

***Signature***

**Dr. Rajeev Tiwari**

**Mentor**

Department of Virtualization,

School of Computer Sciences, U.P.E.S.

Dehradun, Uttarakhand

India-248007

***Signature***

**Dr. Deepshikha Bhargava**

**Head of the Department**

Department of Virtualization,

School of Computer Sciences, U.P.E.S.

Dehradun, Uttarakhand

India- 248007

# Abstract

Software Project Management (SPM) is majorly responsible in whether a software fails or succeeds. It is a key task to determine the time to be invested in the production of the software. Various algorithm can be used for this determination, but the accuracy and reliability of these algorithms play a major role. We are going to compare different machine learning techniques used to accurately predict the software production time. The datasets are made up of records of previously made software projects. Experimental result of this comparison will be presented.

For this experiment we are going to use a data set made up of 29 records. Out of the 29 records 20 will be used for training purpose while 9 will be used for the purpose of testing. After training and testing is done, we will compare all the machine learning algorithm under the same metrices. The metrices are namely:

1) Magnitude relative error
2) Mean magnitude relative error
3) Median of magnitude relative error

the machine learning algorithms that are under study are:

1) Artificial neural network
2) Linear regression
3) Multiple linear regression
4) Gaussian process

For each of the following algorithm a time prediction model was built which is mentioned in the report.

Finally, the result obtained and conclusion have been published in the report.

**Keywords:** machine learning techniques, software project management, software production duration

## Acknowledgement

It is giving us immense pleasure to share our sentiment of gratitude to each and every person and the one above all who have helped in achieving this feat. We express our heartful thanks and owe a profound sense of gratitude to our teacher and Project Mentor, **Mr. Amar Shukla**, University of Petroleum and Energy Studies for their sincere guidance and inspiration in completing this project.

We are exceedingly thankful to **Mr. Pravin Dagdee, Mr. GL Prakash, Mr. Shamik Tiwari and Mr. Harvinder Singh** and all the faculty members of University of Petroleum and Energy Studies for their immense guidance and support in the development of this project.

We would also like to thank all the companions as well as friends of ours who provides us a helping hand in tough times when the project was in a condition from where completing the project would have been a very tedious task.

The in-depth knowledge that we attained while developing this project is immense and will help us in future ventures and will help us in coming up with more similar yet more complex type of solutions.  This process of gaining knowledge was sublime and will help us In our upcoming challenges.

***Name of Students: -***

***Apaar Sharma***

***Devanshu Dwivedi***

# Contents

# List of Figures

## List of tables

UPES

**School of Computer Science**

University of Petroleum & Energy Studies, Dehradun

# Report (2019)

## 1. Project Title

COMPARISION OF MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEVELOPMENT TIME PREDICTION ON HISTORICAL SOFTWARE PROJECT MANAGEMENT RECORDS

## 2. Introduction

Software projects are much more difficult to visualize, control and monitor than the projects that create physical products like a bridge or laptop that can be seen by the eye. It is excessively difficult to meet deadlines with software projects or to deliver the project with all the expected features within a previously decided budget. Before a software project is started it is necessary that intensive planning is done. Without an effective plan the software development cannot be managed.

The data from previously managed software attempts can be used as datasets on which various machine learning techniques can be used to predict the duration of the project which makes it possible to come up with realistic deadlines and budget making software management a lot easier.

Accurate predictions for new projects can be achieved by methods like several methods using statistics such as correlation analysis and linear regression or techniques of machine learning like ANN (Artificial Neural Network).

Approaches like this require prediction function that according to the dataset available i.e. past record of earlier projects will be predicting the future effort and duration of software production.
In this project we’re going to compare various machine learning algorithm and present the result obtained during the process.

# 3. Literature review

In this section we're briefly going to talk about various machine learning techniques and approaches we will be comparing.

**3.1 neural networks:**

A neural network works pretty much like how signals are passed and processed by the neurons found in biological beings. [1]

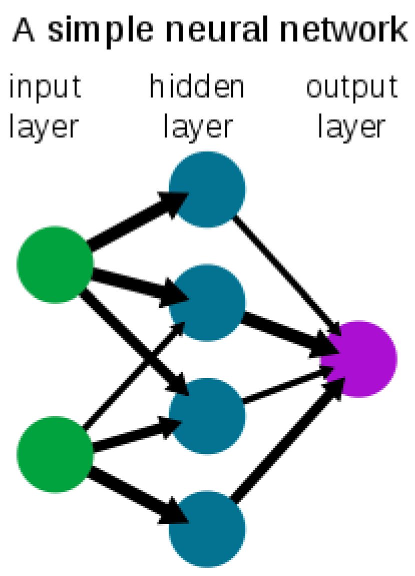


FIG 1) a simple ANN

The learning in the neural networks is done by adjusting parameters like bias and weight which is assigned to a neuron. It usually takes place during two periods which are named.

1) Training period
2) Application period

To train the neural network some learning algorithm is usually used. [2]

**3.2 Linear Regression**

Linear regression (LR) is used to find the linear relationship between two scaler variables. One is dependent variable y and one is independent variable x[3].

**3.3 Multiple linear regression**

Multiple Linear Regression is Simple Linear Regression, but with more Relationships.

The multiple linear regression will explain the relationship that is present between one continuous variable that is dependent (y) and more than one independent variables like (x1, x2, x3, x4… etc.). [4]

**3.4 Gaussian Process**

Gaussian process work on the theory of probability, it is made up of arbitrary values related to every point during a period or range time. The random or the arbitrary variable has its distribution propagated normally. And it is updated whenever new data appears. It is called a non-parametric approach. Basically, it is preferred when we want a curved line in place of a straight or linear line. It is called non parametric because infinite numbers of parameters are possible and can be used. But using all the functions is not possible as a prior so we need to put some constraints on it. We aim for a function that gives us a smooth graph. To achieve smoothness, we use covariance matrix which allows us to use values which are placed closely together in an input space. [5]

**various python libraries and tools used:**

1) **Numpy:** python programming language uses this library to add support for 2D and 3D arrays as well as complex and useful mathematical functions.

2) **Pandas:** it is an open source library used for the manipulation and analysis of data. It is also known as the python data analysis library. We are going to use it to automate our productive model building.

3) **Matplotlib:** as the name suggests this library is used to plot varioug graphs in python programming language.

4) **Sklearn:** scikit learn is the full form of sklearn. Machine learning library for python programming language. Provides various tools for the purpose of data mining and analyzing data. It is built on NumPy, SciPy and matplotbil. Can be used for various processes like classification, regression, clustering, dimension reduction, model selection and reprocessing etc.

## 4 Problem statement

Software project management is critical for the success of a software. For this accurate prediction of a feasible deadline is required. It’s not easy to predict the software development period for a software. For this existing record of previous software management lifecycles can be used as datasets for machine learning techniques to predict this software development duration.

There are plenty of machine learning techniques out there for prediction of the software development duration, so it is critical to use a technique which will provide the most accurate result. Thus, we are going to compare several machine learning techniques on the following data set.

## 5 Objectives

The primary objective is to come up with a conclusion about which prediction-based machine learning algorithm is most suitable for software development duration prediction during Software Project Management.

Techniques like ANN, Linear Regression, Least Median Square and Gaussian Process will be ran on a data set of previous Software Project Management records. Some data set will be used for teaching the algorithm, others will be used for testing the algorithm thus, a comparative study will be created.

## 6 System Requirements

**Hardware:**

1) 64-bit processor architecture that can support Microsoft windows or Linux
2) Suggested RAM specification capacity is 2GB.
3) Required input and output devices like keyboard and monitor

**Software:**

1) Windows 7,8, 10 or any flavor of Linux
2) Python
3) Anaconda IDE
4) SPYDER EDITOR
5) Sublime text editor

## 7 Methodology

The primary agenda or objective of this project is to compare how effective various vivid machine learning algorithms are two predict the software development time of a project.

We are going to use number of workers in person, time duration in months, effort in person month and lines of code in KLOC to predict the time for development of project in months.

The same data which was randomly selected was used for all the machine learning algorithms to be used in the comparative study.

Some refinement of data took place and finally these variables and records were used.

| Total number of projects used | Man power | Time taken by the projects in months | Effort calculated in person month | Line of code calculated in KLOC |
|---|---|---|---|---|
| 29 | 7-29 | 3-29 | 5-210 | 3-310 |

TABLE 1) INFORATION ABOUT THE DATASET

### Steps taken for the refinement of dataset:

**Step 1: descriptive analysis:** in this we try to find out any missing values or important features, just identify the columns with missing values.

**Step 2: treatment of the data:** in this the values that were missing are treated i.e. They are replaced by dummy values. The method we used was median.

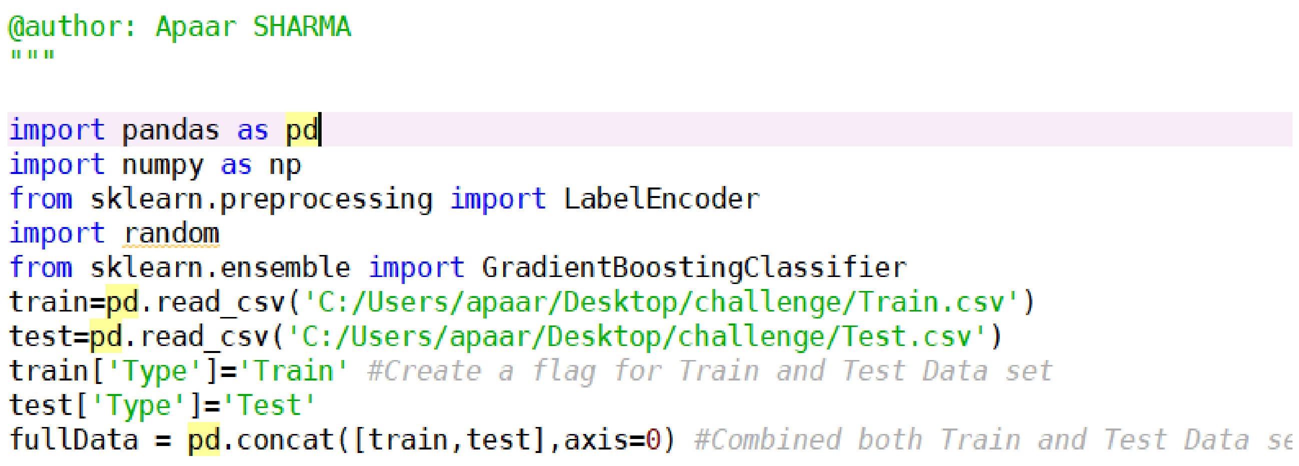
**Step 3: modeling the data:** we used GBM (gradient boosting algorithm) for this.

This was done in the way of bagging. In this we took random data build learning algorithms for it and relied on simple mean for the bagging probabilities.

**Step 4: performance was estimated:** in this step validation for the performance model has to be obtained. Works on train and test model.

## Implementation steps:

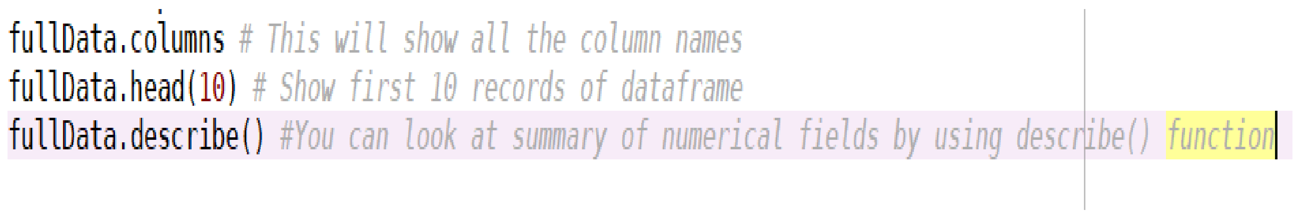
**Step 1 :** import the required libraries. Read the training and testing datasets



```
@author: Apaar SHARMA
"""

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import random
from sklearn.ensemble import GradientBoostingClassifier
train=pd.read_csv('C:/Users/apaar/Desktop/challenge/Train.csv')
test=pd.read_csv('C:/Users/apaar/Desktop/challenge/Test.csv')
train['Type']='Train' #Create a flag for Train and Test Data set
test['Type']='Test'
fullData = pd.concat([train,test],axis=0) #Combined both Train and Test Data se
```

Fig 2

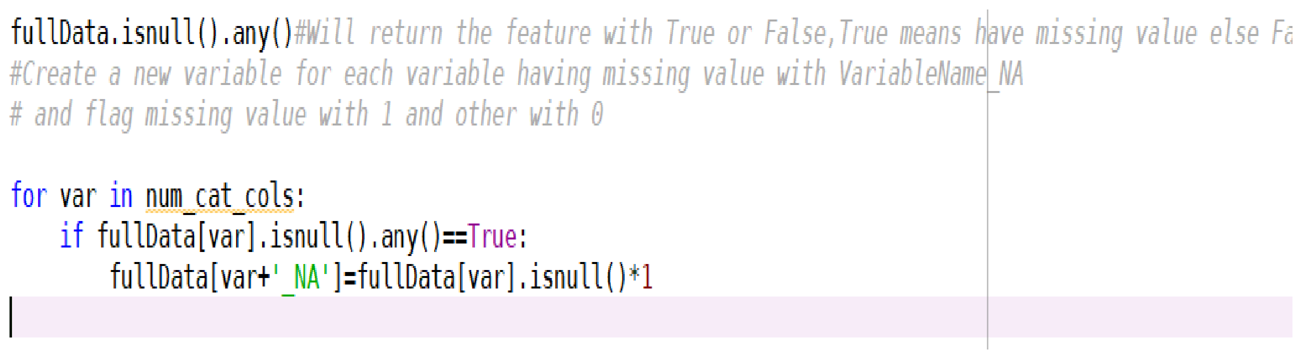
**Step 2:** view the column names



```
fullData.columns # This will show all the column names
fullData.head(10) # Show first 10 records of dataframe
fullData.describe() #You can look at summary of numerical fields by using describe() function
```

Fig 3

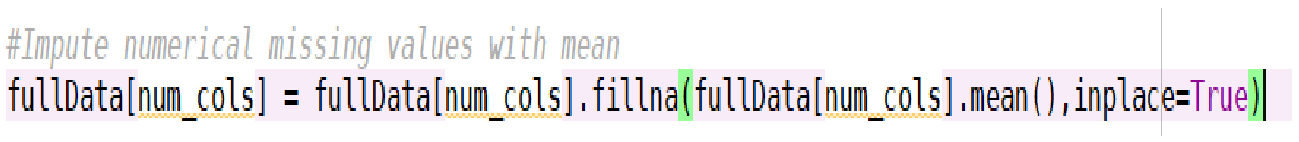
**Step 3:** identify the variables with missing values and create tags for it.



```
fullData.isnull().any()#Will return the feature with True or False,True means have missing value else Fa
#Create a new variable for each variable having missing value with VariableName_NA
# and flag missing value with 1 and other with 0

for var in num_cat_cols:
    if fullData[var].isnull().any()==True:
        fullData[var+'_NA']=fullData[var].isnull()*1
```

Fig 4

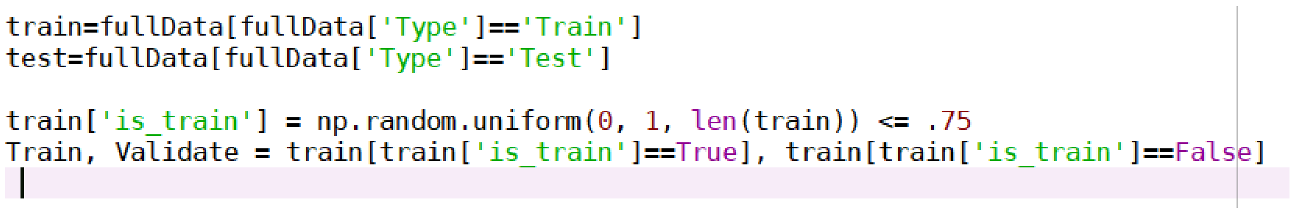
**Step 4:** put in the missing values.



```
#Impute numerical missing values with mean
fullData[num_cols] = fullData[num_cols].fillna(fullData[num_cols].mean(),inplace=True)
```

Fig 5

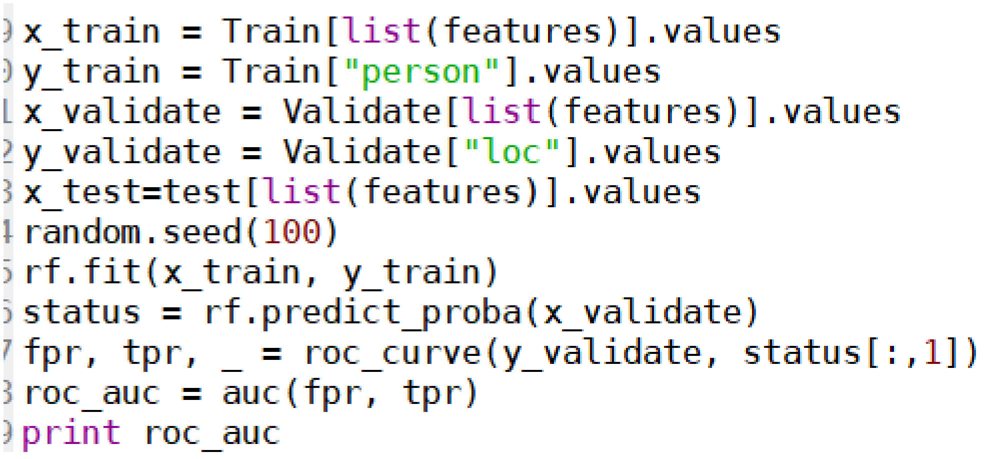
**Step 5:** split the data set into train and validate then again we will split the dataset used for train to train and validate.



```
train=fullData[fullData['Type']=='Train']
test=fullData[fullData['Type']=='Test']

train['is_train'] = np.random.uniform(0, 1, len(train)) <= .75
Train, Validate = train[train['is_train']==True], train[train['is_train']==False]
```

Fig 6

**Step 6:** pass the dummy values for missing values. And check performance



```
x_train = Train[list(features)].values
y_train = Train["person"].values
x_validate = Validate[list(features)].values
y_validate = Validate["loc"].values
x_test=test[list(features)].values
random.seed(100)
rf.fit(x_train, y_train)
status = rf.predict_proba(x_validate)
fpr, tpr, _ = roc_curve(y_validate, status[:,1])
roc_auc = auc(fpr, tpr)
print roc_auc
```

Fig 7

## The work flow is as follows:

**Step 1:** collect data set consisting of previous records available for software development.

**Step 2:** the found dataset is made up by selecting 29 records.

**Step 3:** use 20 records to train the machine learning technique

**Step 4:** use the remaining 9 records to test the results.

**Step 5:** time prediction model for each algorithm is built.

**Step 6:** compare the result of all algorithm

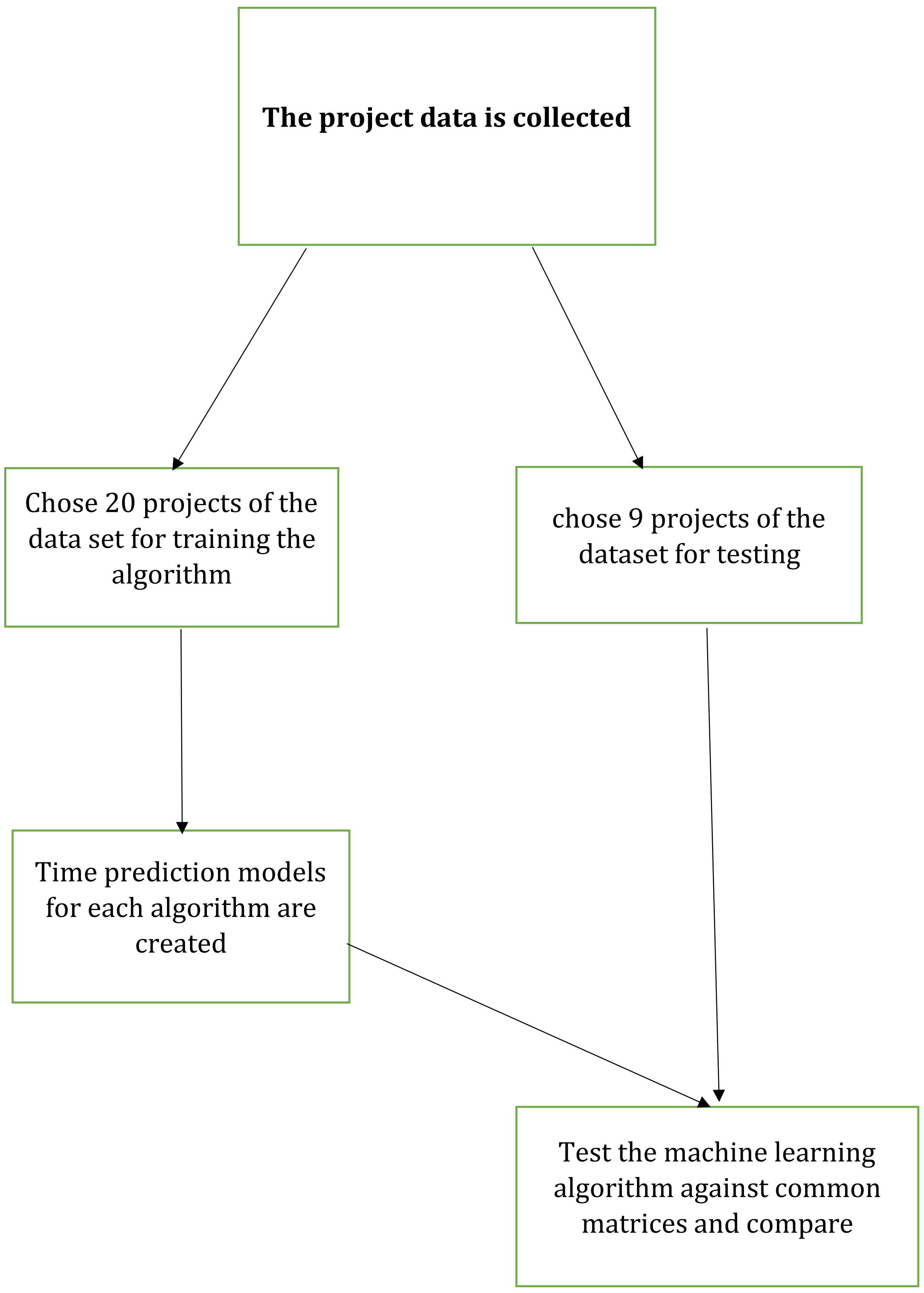

The project data is collected
Chose 20 projects of the data set for training the algorithm
chose 9 projects of the dataset for testing
Time prediction models for each algorithm are created
Test the machine learning algorithm against common matrices and compare

Fig 8: flow chart of building model

## Test and comparison

In order to find out how accurately the models perform we use common evaluation criteria or matrices namely magnitude relative error (MRE), mean magnitude relative error (MMRE) and median of MRE.

1) **magnitude relative error (MRE):** it is used for the purpose of pinpoint calculation of the absolute percentage of error between the time we predicted using our algorithm to the actual value given in our data set

$$MRE = \frac{ACTUAL\ value - ESTIMATED\ value}{ACTUAL\ value}$$

2) **mean magnitude relative error (MMRE):** calculates the average of MRE over the given projects (all( which are 29 in our case.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE$$

3) **median of MRE (MdMRE):** as the name suggests it is the median of MRE. It is used because for extreme values MMRE can be very sensitive

$$MdMRE = median(i)\{MRE(i)\}$$

## 8 Result

Here is a tabular result obtained between the following

1) ANN
2) Linear regression
3) Multiple regression
4) Gaussian process

On the matrices used:

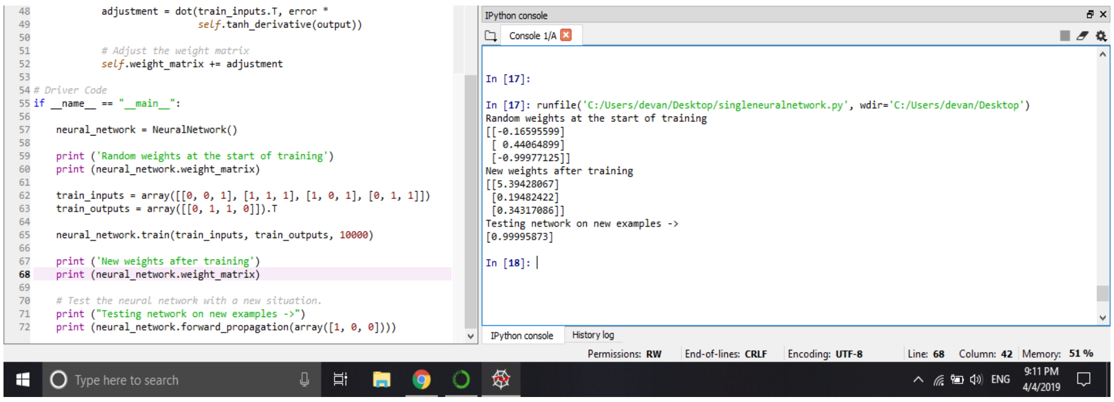
1) MRE
2) MMRE
3) MdMRE

| | Linear R | Multiple R | ANN | Gaussian P |
|---|---|---|---|---|
| MRE | 0.909 | 0.899 | 0.919 | 0.976 |
| MMRE | 0.874 | 0.922 | 0.936 | 0.951 |
| MdMRE | 0.914 | 0.933 | 0.900 | 0.960 |

TABLE 2 the evaluation

## 9 Output:

### NEURAL NETWORK

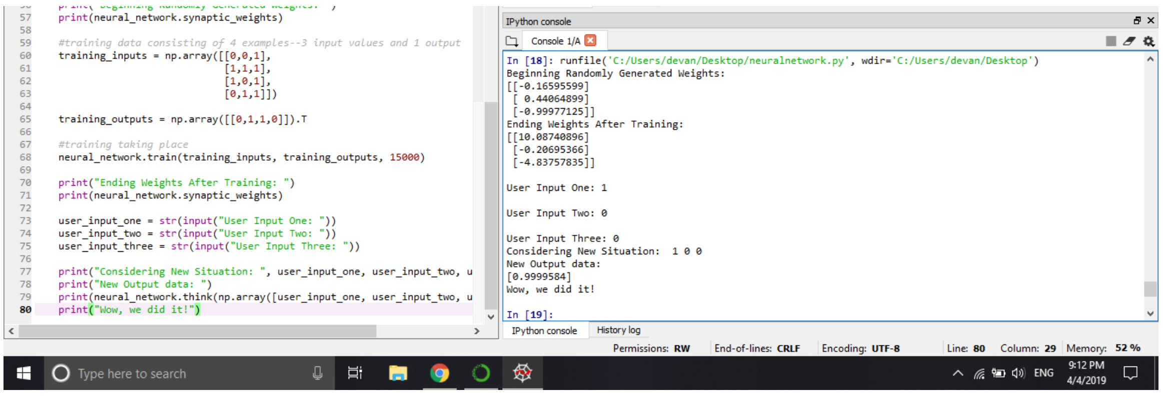


```
            adjustment = dot(train_inputs.T, error *
                             self.tanh_derivative(output))

            # Adjust the weight matrix
            self.weight_matrix += adjustment

# Driver Code
if __name__ == "__main__":

    neural_network = NeuralNetwork()

    print ('Random weights at the start of training')
    print (neural_network.weight_matrix)

    train_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1, 1]])
    train_outputs = array([[0, 1, 1, 0]]).T

    neural_network.train(train_inputs, train_outputs, 10000)

    print ('New weights after training')
    print (neural_network.weight_matrix)

    # Test the neural network with a new situation.
    print ("Testing network on new examples ->")
    print (neural_network.forward_propagation(array([1, 0, 0])))
```

```
In [17]:

In [17]: runfile('C:/Users/devan/Desktop/singleneuralnetwork.py', wdir='C:/Users/devan/Desktop')
Random weights at the start of training
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
New weights after training
[[5.39428067]
 [0.19482422]
 [0.34317086]]
Testing network on new examples ->
[0.99995873]

In [18]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 68 Column: 42 Memory: 51 %

Fig 9 neural network code output in previous run



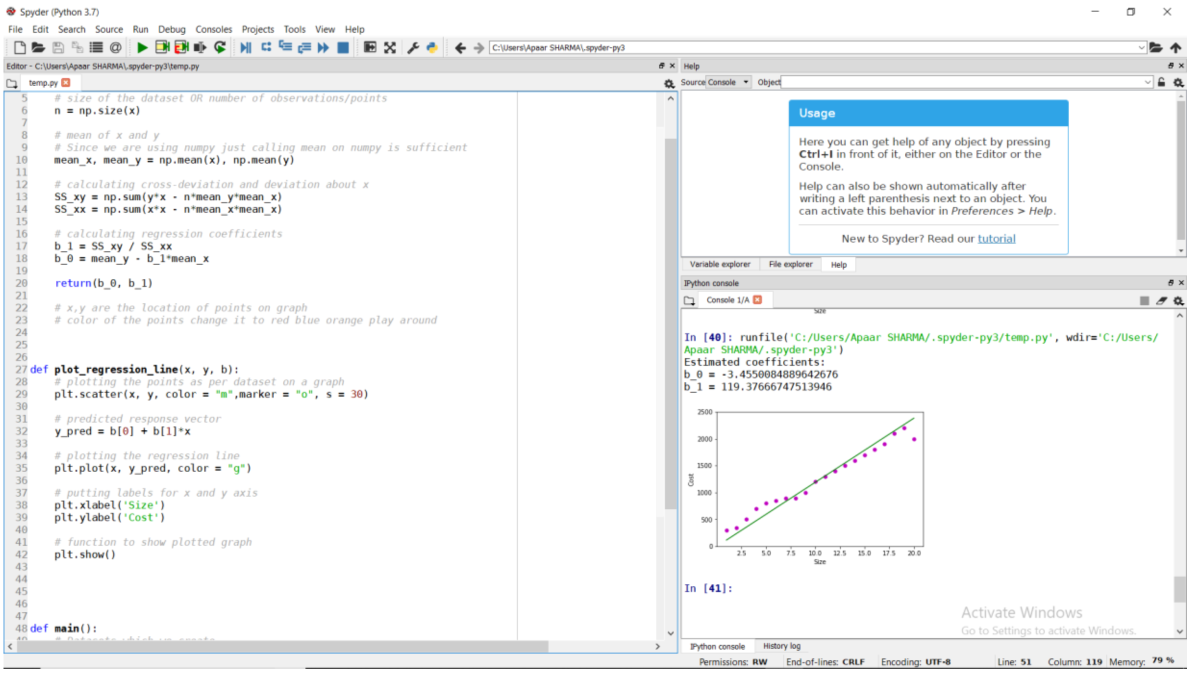
```
    print(neural_network.synaptic_weights)

    #training data consisting of 4 examples--3 input values and 1 output
    training_inputs = np.array([[0,0,1],
                                [1,1,1],
                                [1,0,1],
                                [0,1,1]])

    training_outputs = np.array([[0,1,1,0]]).T

    #training taking place
    neural_network.train(training_inputs, training_outputs, 15000)

    print("Ending Weights After Training: ")
    print(neural_network.synaptic_weights)

    user_input_one = str(input("User Input One: "))
    user_input_two = str(input("User Input Two: "))
    user_input_three = str(input("User Input Three: "))

    print("Considering New Situation: ", user_input_one, user_input_two, u
    print("New Output data: ")
    print(neural_network.think(np.array([user_input_one, user_input_two, u
    print("Wow, we did it!")
```

```
In [18]: runfile('C:/Users/devan/Desktop/neuralnetwork.py', wdir='C:/Users/devan/Desktop')
Beginning Randomly Generated Weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Ending Weights After Training:
[[10.08740896]
 [-0.20695366]
 [-4.83757835]]

User Input One: 1

User Input Two: 0

User Input Three: 0
Considering New Situation:  1 0 0
New Output data:
[0.9999584]
Wow, we did it!

In [19]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 80 Column: 29 Memory: 52 %

Fig 10 neural network code output after bias optimization

## LINEAR REGRESSION


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Apaar SHARMA\.spyder-py3
Editor - C:\Users\Apaar SHARMA\.spyder-py3\temp.py
temp.py
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log

```python
    # size of the dataset OR number of observations/points
    n = np.size(x)

    # mean of x and y
    # Since we are using numpy just calling mean on numpy is sufficient
    mean_x, mean_y = np.mean(x), np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y*x - n*mean_y*mean_x)
    SS_xx = np.sum(x*x - n*mean_x*mean_x)

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = mean_y - b_1*mean_x

    return(b_0, b_1)

    # x,y are the location of points on graph
    # color of the points change it to red blue orange play around



def plot_regression_line(x, y, b):
    # plotting the points as per dataset on a graph
    plt.scatter(x, y, color = "m",marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x

    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

    # putting labels for x and y axis
    plt.xlabel('Size')
    plt.ylabel('Cost')

    # function to show plotted graph
    plt.show()





def main():
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
In [40]: runfile('C:/Users/Apaar SHARMA/.spyder-py3/temp.py', wdir='C:/Users/
Apaar SHARMA/.spyder-py3')
Estimated coefficients:
b_0 = -3.4550084889642676
b_1 = 119.37666747513946
```

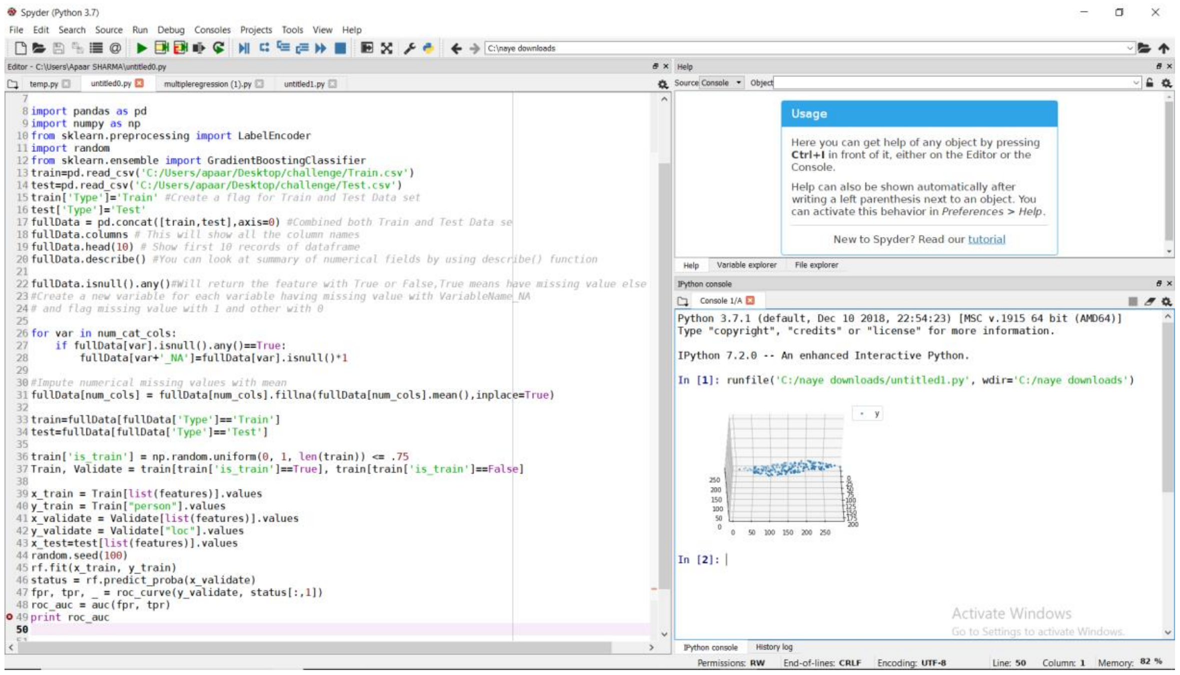
```
In [41]:
```

Activate Windows
Go to Settings to activate Windows.

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 51 Column: 119 Memory: 79 %

Fig 11 output for linear regression

## MULTIPLE REGRESSION


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\naye downloads
Editor - C:\Users\Apaar SHARMA\untitled0.py
temp.py
untitled0.py
multipleregression (1).py
untitled1.py
Help
Source Console Object
Help Variable explorer File explorer
IPython console
Console 1/A
IPython console History log

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
import random
from sklearn.ensemble import GradientBoostingClassifier
train=pd.read_csv('C:/Users/apaar/Desktop/challenge/Train.csv')
test=pd.read_csv('C:/Users/apaar/Desktop/challenge/Test.csv')
train['Type']='Train' #Create a flag for Train and Test Data set
test['Type']='Test'
fullData = pd.concat([train,test],axis=0) #Combined both Train and Test Data se
fullData.columns # This will show all the column names
fullData.head(10) # Show first 10 records of dataframe
fullData.describe() #You can look at summary of numerical fields by using describe() function

fullData.isnull().any()#Will return the feature with True or False,True means have missing value else
#Create a new variable for each variable having missing value with VariableName_NA
# and flag missing value with 1 and other with 0

for var in num_cat_cols:
    if fullData[var].isnull().any()==True:
        fullData[var+'_NA']=fullData[var].isnull()*1

#Impute numerical missing values with mean
fullData[num_cols] = fullData[num_cols].fillna(fullData[num_cols].mean(),inplace=True)

train=fullData[fullData['Type']=='Train']
test=fullData[fullData['Type']=='Test']

train['is_train'] = np.random.uniform(0, 1, len(train)) <= .75
Train, Validate = train[train['is_train']==True], train[train['is_train']==False]

x_train = Train[list(features)].values
y_train = Train["person"].values
x_validate = Validate[list(features)].values
y_validate = Validate["loc"].values
x_test=test[list(features)].values
random.seed(100)
rf.fit(x_train, y_train)
status = rf.predict_proba(x_validate)
fpr, tpr, _ = roc_curve(y_validate, status[:,1])
roc_auc = auc(fpr, tpr)
print roc_auc
```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/naye downloads/untitled1.py', wdir='C:/naye downloads')
```

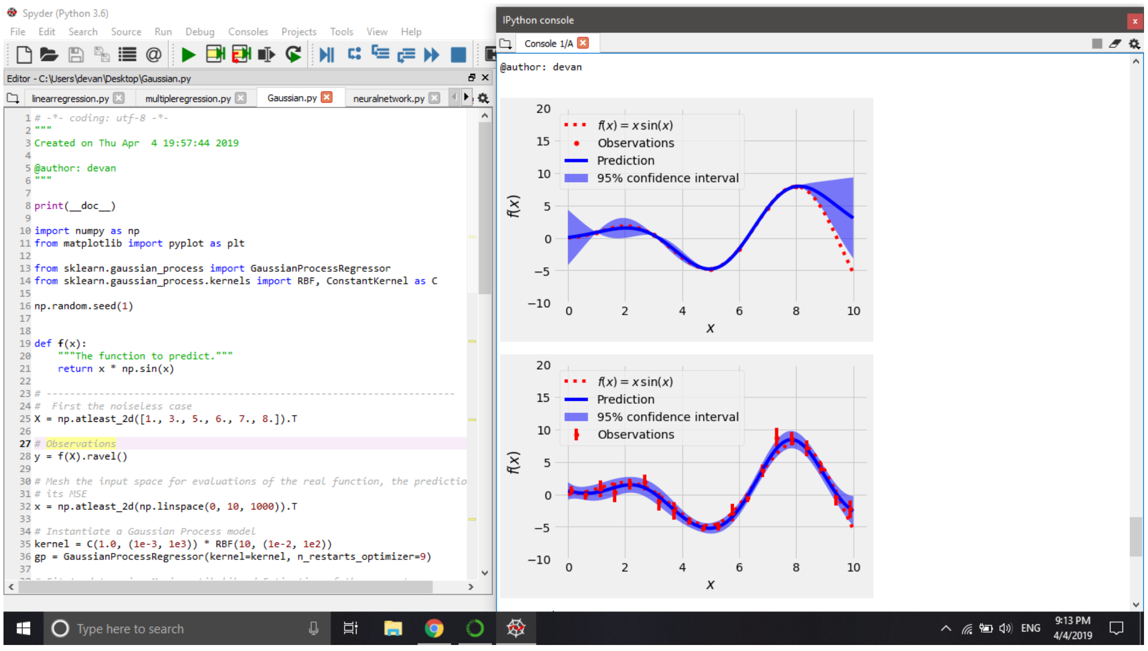
```
In [2]:
```

Activate Windows
Go to Settings to activate Windows.

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 50 Column: 1 Memory: 82 %

Fig 12 output for multiple regression

## GAUSSIAN PROCESS



```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  4 19:57:44 2019

@author: devan
"""


print(__doc__)

import numpy as np
from matplotlib import pyplot as plt

from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C

np.random.seed(1)


def f(x):
    """The function to predict."""
    return x * np.sin(x)

# ----------------------------------------------------------------------
#  First the noiseless case
X = np.atleast_2d([1., 3., 5., 6., 7., 8.]).T

# Observations
y = f(X).ravel()

# Mesh the input space for evaluations of the real function, the predictio
# its MSE
x = np.atleast_2d(np.linspace(0, 10, 1000)).T

# Instantiate a Gaussian Process model
kernel = C(1.0, (1e-3, 1e3)) * RBF(10, (1e-2, 1e2))
gp = GaussianProcessRegressor(kernel=kernel, n_restarts_optimizer=9)
```

```
@author: devan
```

Fig 13 output for gaussian process

## 10 Pert Chart

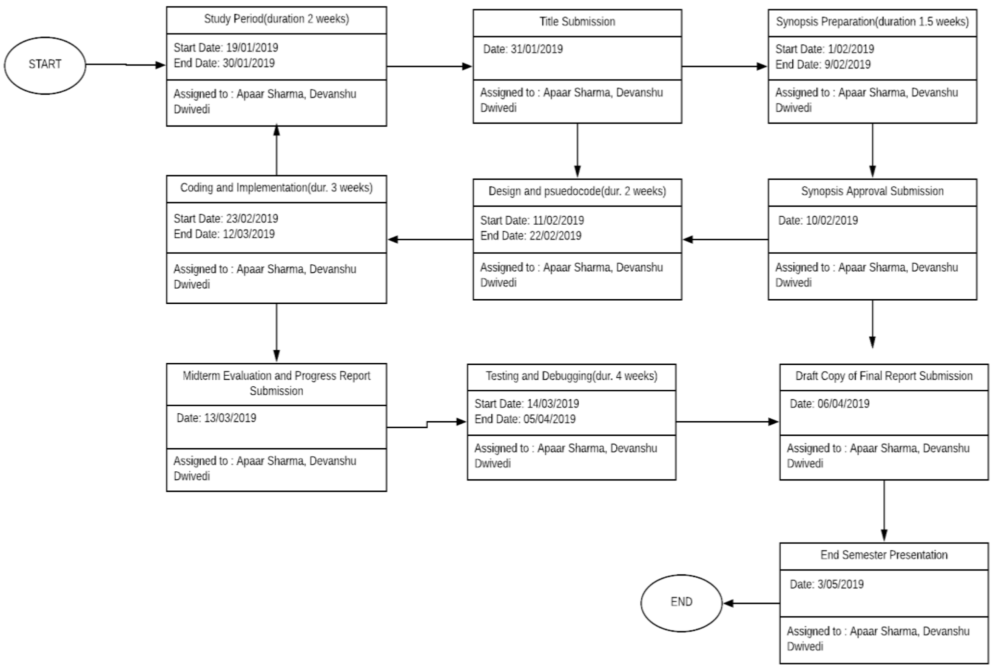

START
Study Period(duration 2 weeks)
Start Date: 19/01/2019
End Date: 30/01/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Title Submission
Date: 31/01/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Synopsis Preparation(duration 1.5 weeks)
Start Date: 1/02/2019
End Date: 9/02/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Synopsis Approval Submission
Date: 10/02/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Design and psuedocode(dur. 2 weeks)
Start Date: 11/02/2019
End Date: 22/02/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Coding and Implementation(dur. 3 weeks)
Start Date: 23/02/2019
End Date: 12/03/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Midterm Evaluation and Progress Report Submission
Date: 13/03/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Testing and Debugging(dur. 4 weeks)
Start Date: 14/03/2019
End Date: 05/04/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
Draft Copy of Final Report Submission
Date: 06/04/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
End Semester Presentation
Date: 3/05/2019
Assigned to : Apaar Sharma, Devanshu Dwivedi
END

Fig 14 PERT CHART

## 11 Conclusion

In our project we used different predictive machine learning algorithms namely: ANN, Linear regression, multiple regression and gaussian process to build software time prediction models which was based on the project data available in our data set which was plenty.

After this we deeply compared variety of prediction-based machine learning algorithms. Our overall experimental result portrays that **GAUSSIAN PROCESS** managed to show the best performance and accuracy out of all the algorithms under study.

We believe that is was the case because Gaussian process is a non-parametric algorithm which means that we can give it infinite number of parameters if we want.

Not only that, Gaussian process helps achieve a best fit curve rather than a straight line like linear regression. The curve can be made very smooth with a covariance matrix so that the input domain has the values which are near closed together.

In this project we can also witness how building predictive models have become so easy with the python libraries like:

Numpy: which provides support for huge number of math functions

Pandas: for the purpose and manipulation and analysis of data

Matplotlib: plotting library of python

Sklearn: known as scikit-learn provides various machine learning features like clustering and regression.

## 10 References

[1] Ms. Sonali. B. Maind, Research Paper on Basic of Artificial Neural Network

[2] S. K Shevade, S. S. Keerthi, C. Bhattacharyya and K. R. K. Murthy, "Improvements to the SMO Algorithm for SVM Regression", IEEE Transactions ON Neural Networks, vol. 13, no. 7, September (2000)

[3] A. Andrzejak and L. Silva, "Using Machine Learning for Non-Intrusive Modeling and Prediction of Software Aging", Network Operations and Management Symposium, IEEE, April 7-11, (2008), pp. 25-32

[4] J. A. Lopez, J. L. Berral, R. Gavalda and J. Torres, "Adaptive on-line software aging prediction based on machine learning", in Procs. 40th IEEE/IFIP Intl. Conf. on Dependable Systems and Networks, June 28, July 1, (2010), pp. 497-506

[5] D. J. C. Mackay, “Introduction to Gaussian Processes”, Dept. of Physics, Cambridge University, UK, (1998).

[6] I. H. Witten and E. Frank, “Data Mining Practical Machine Learning Tools and Techniques”, Morgan Kaufmann Publishers Is an Imprint of Elsevier, (2005), pp. 187-427