

Report

on

College Student Marks Prediction

Submitted in partial fulfilment of the requirements for the award of degree

in the department of

Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB



LOVELY
PROFESSIONAL
UNIVERSITY

Submitted by:

Pankaj Raina (11910326)

Section: KM055

Date of submission: March 30, 2022

Declaration

I hereby declare that the project work entitled "College Student Marks Prediction" submitted to the Lovely Professional University Punjab, is a record of an original work done by me under the guidance of Dr. Sagar Pande, Professor in Dept Of Computer Science & Engineering, Lovely Professional University, and this project work is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Index

S.No	Topic
1.	Abstract
2.	Introduction
3.	Literature Review
4.	Technology Implemented
5.	Advantages and Disadvantages
6.	Concept of Hyperparameter Tuning
7.	Result and Comparative Analysis
8.	Pictures of Python File
9.	Conclusion
10.	Future Scope
11.	Reference

Abstract

In this project, I was asked to experiment with a real-world dataset, and to explore how machine learning algorithms can be used to ease our work and to predict the new output with the help of the given data. I was expected to gain experience using a common machine learning library and was expected to submit a report about the dataset and the algorithms used. After performing the required tasks on a dataset of my choice, this report is describing the brief of my project.

Various machine learning algorithms are used and comparative analysis within those algorithms has been made. An API is also developed to show the practical evaluation and live working of machine learning model. Furthermore, HTML website is also created to show the sequenced and statistical manner to show the basic building blocks of ML project.

Introduction

In this project I used hybridised dataset so that I can work on larger dataset. I trained our model to predict students marks according to the student study hours for that we used linear regression algorithms but after using linear regression to train our model I found that the accuracy of model is 81% which is quite low so in order to increase our model accuracy I used two other algorithms which are random forest regressor and gradient boosting regressor and after that I used ensemble model to combine the accuracy of my model which rose from 81% to 92% which is quite good. Afterwards I calculated the R2 score, variance score, absolute error and mean squared log error for each algorithm and later used those to compare the random forests regressor and gradient boosting regressor's R2 score, variance score, absolute error and mean squared log error to compare with the regressor's R2 score, variance score, absolute error and mean squared log error of linear regression model. Machine Learning can be thought of as the study of a list of sub-problems, viz: decision making, clustering, classification, forecasting, deep-learning, inductive logic programming, support vector machines, reinforcement learning, similarity and metric learning, genetic algorithm, sparse dictionary learning, etc. Supervised learning, or classification is the machine learning task of inferring a function from a labelled data. In Supervised learning, we have a training set, and a test set. The training and test set consists of a set of examples consisting of input and output vectors, and the goal of the supervised learning algorithm is to infer a function that maps the input vector to the output vector with minimal error. In an optimal scenario, a model trained on a set of examples will classify an unseen example in a correct fashion, which requires the model to generalize from the training set in a reasonable way. In layman's terms, supervised learning can be termed as the process of concept learning, where a brain is exposed to a set of inputs and result vectors and the brain learns the concept that relates said inputs to output. A wide array of supervised machine learning algorithms are available to the machine learning enthusiast, for example Neural Networks Decision Trees, Support Vector Machines Random Forest, Naive Bayes Classifier, Bayes Net, Majority Classifier(4.7.8.9) etc. and they each have their own merits and demerits. There is no single algorithm that works for all cases, as merited by the No free lunch theorem. In this project, I tried and found patterns in a dataset. which is a sample of marks

obtained by studying for certain number of hours and attempt to throw various intelligently picked algorithms at the data and see what sticks.

For the case of hyper parameter tuning, I have used grid search to select the important parameters. Another important machine learning algorithm used is XG boost regressor called as extreme gradient boosting is generally an open-source library to implement gradient boosting algorithm in an effective and efficient manner.

Dataset: My dataset is a collection of result of marks obtained by studying for certain number of hours. I collected this dataset by my own, for this I asked students about their marks and their study hours. By doing so I was able to create a appropriate dataset consisting of samples of students marks and study hours. Given below is how my dataset looks like: -

Study hours	Student marks
6.83	78.5
6.56	76.74
	78.68
5.67	71.82
8.67	84.19
7.55	81.18
6.67	76.99
8.99	85.46
5.19	70.66
6.75	77.82
6.59	75.37
8.56	83.88
7.75	79.5
7.9	80.76
8.19	83.08
6.55	76.03
6.36	76.04

But these values are very less to predict actual precision and r^2 score, so hereby I introduced another set of 200 values to make this dataset a hybridised dataset in order to make comparative analysis.

Literature Review

Theory

A core objective of a learner is to generalize from its experience. The computational analysis of linear regression algorithms on my dataset as with other algorithms like random forest regressor and gradient boosting regressor also and their performance on the dataset. As most of the study on this dataset is limited to Linear regression based model but the dataset which they were using was small and on that small dataset they got around 94% accuracy which is quiet good but we cannot neglect the fact the dataset was also very small so in order to further increase the work on this dataset I hybridised the dataset and then again used the algorithm which was used till this date on this dataset which was linear regression what I found is that after using linear regression on the hybridised dataset the accuracy dropped to 81% which is not that good but what is the benefit of 95% if you get it from very small dataset. So in order to rid of this problem I fine-tuned my model and before that I cleaned my dataset, in order to clean my dataset I checked my dataset whether it contain duplicate values or some null values and by doing so I found that my dataset consist of some null values or we can say that some void spaces, so in order to fix that I used mean function to fill those void spaces and after doing that I trained my model with another algorithm which is random forest regressor. Random forest regressor used ensemble model training which is to get a combination of accuracies of different algorithms as we are going to use three different algorithms to train or model which is linear regression, random forest regressor and gradient boosting regressor. After using random forest regressor I used gradient boosting regressor which is also very powerful algorithm as not only it works with continuous target variable but also it works with categorical target variable. And after using all three algorithm I calculated the ensemble model which I got around 94% which is good. Most of the studies or I could say all the study on this dataset till this date is just limited to linear regression only so my novelty in this project is that I used two other algorithms to train my model so that I could get a good accuracy on a larger dataset which is what I got after my model training. In order to further go further beyond I calculated the R2 score, variance score, mean absolute error and mean squared log error. Basically, R2 score is

just to evaluate the performance of a regression-based model. Also sometimes known as a coefficient of determination. What I have learnt is if R^2 score is above 0.9 then your model is good. Variance score is how far the actual value differ from the average of predicted value and it should not be less than 60% if it is greater than 60% then your model is performing good. Mean squared log error it is how close our fitted lines to the data point for it the lower the mean squared log error the higher is the accuracy. So after calculating the R^2 score, variance score, absolute error and mean squared log error for each algorithm and later on used those to compare the random forests regressor and gradient boosting regressor's R^2 score, variance score, absolute error and mean squared log error to compare with the regressor's R^2 score, variance score, absolute error and mean squared log error of linear regression model. The bounds on the performance are quite common. The bias variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, the model is subject to overfitting and generalization will be poor.

In addition to performance bounds, learning theorists study the time complexity and feasibility of learning in computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

In various machine learning models designed on this data set by others is restricted to original dataset and linear regression only. But I have introduced various other algorithms to make it an efficient and effective ml model. Random forest regression, Gradient Boosting Technique, Bayesian Ridge and XG Boost regressor have been used to generate a desired output in less time and space complexity.

Comparative analysis has also been made that isn't done earlier. the comparison of r^2 score of various algorithms is made with the help of graphs in particular bar graph. The predicted output of algorithms is also compared with help of graphs.

To choose a set of optimal hyper parameter for learning algorithm hyper parameter tuning is also introduced and grid search that is used to define a search space as a grid of hyper parameter values and evaluate every position in the grid.

Technology Implemented

Python - The New Generation Language

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for an emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Features

1.Interpreted

In Python there is no separate compilation and execution steps like C/C++. It directly run the program from the source code. Internally, Python converts the source code into an intermediate form called bytecodes which is then translated into native language of specific computer to run it.

2.Platform Independent

Python programs can be developed and executed on the multiple operating system platform. Python can be used on Linux, Windows, Macintosh, Solaris and many more.

3.Multi-Paradigm

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming.

4.Simple

Python is a very simple language. It is a very easy to learn as it is closer to English language. In python more emphasis is on the solution to the problem rather than the syntax.

5. Rich Library Support

Python standard library is very vast. It can help to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML, WAV files, cryptography, GUI and many more.

Free and Open Source

Firstly, Python is freely available. Secondly, it is open source. This means that its source code is available to the public. We can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open-Source Software). As the Python community, we're all headed toward one goal - an ever-bettering Python.

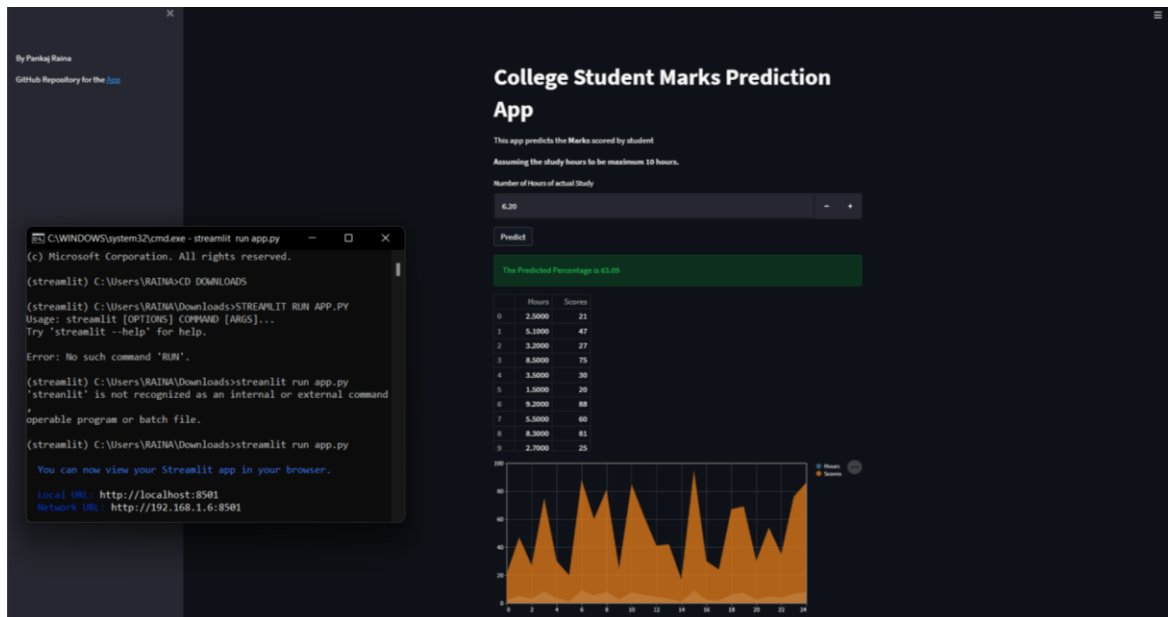
Streamlit

It is an application platform interface or GUI based technology to showcase the practical evaluation of your machine learning model.

Streamlit makes it easy for you to visualise, mutate and shared data. The API references organised by activity type like displaying data or optimising performance.

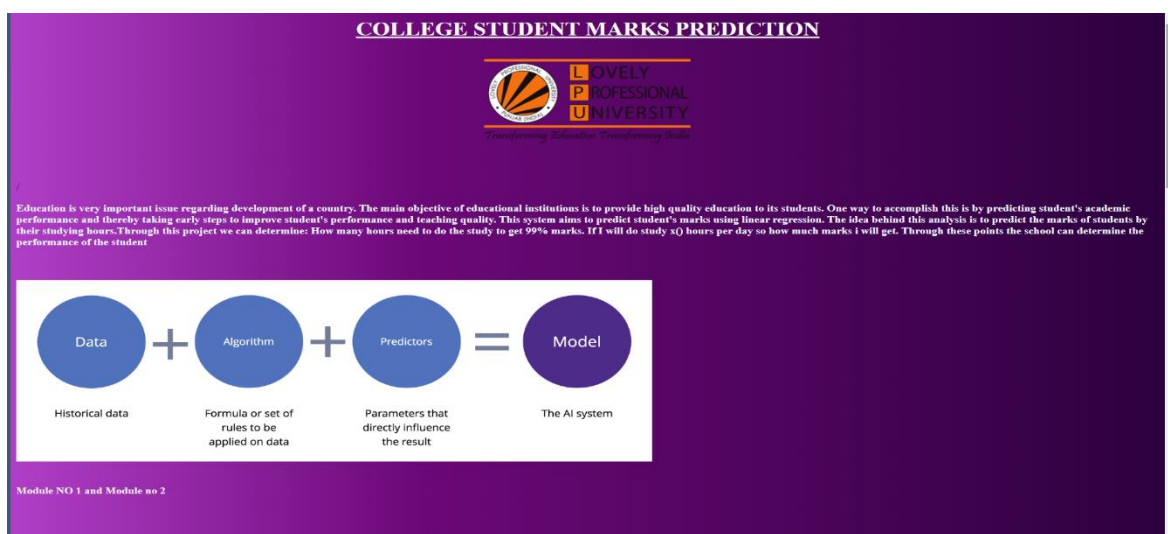
User doesn't know about front end it is not necessary that user find it very difficult to use streamlit. Users need to have basic knowledge of Python language then it is good to go to use this friendly platform.

User must use basic Python functions to get input from the front-end screen and from the backend take a desired output value from the data set.

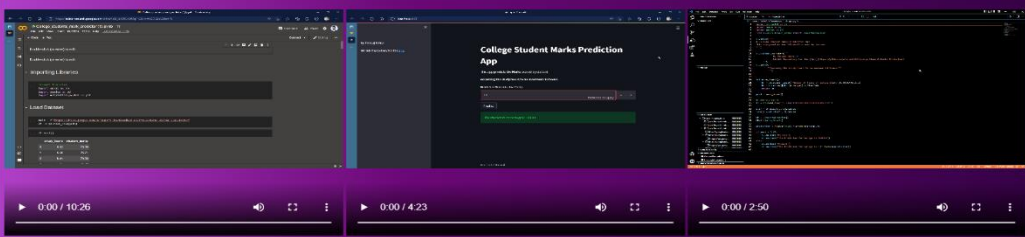


HTML

To showcase the sequence manner of working of ml model and GUI platform Hypertext markup language is used to display the desired output in a web browser. To style the website cascading style sheets is also used. It gives the complete description how to use the platform end a better understanding to the code part of model.



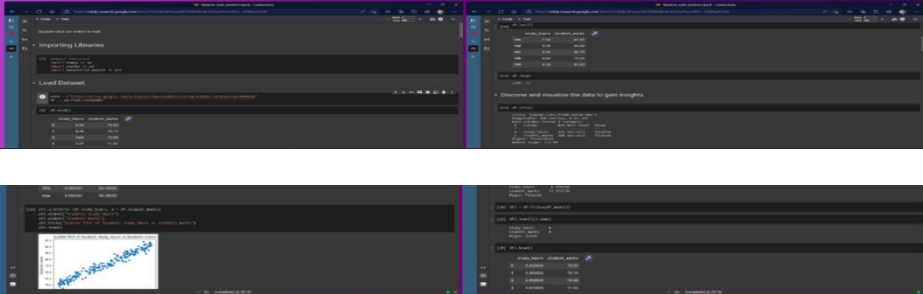
Module NO 1 and Module no 2



0:00 / 10:26 0:00 / 4:23 0:00 / 2:50

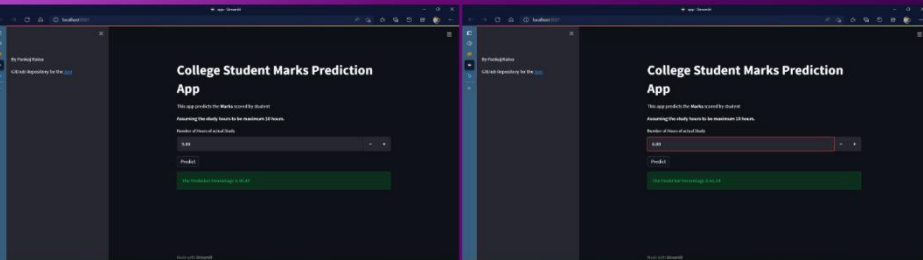
[Link of Dataset required](#)

Few sample images of code



[Python source code for model](#)

Practical Evaluation Of Model



[Github Link for Project](#)

Designed by Pankaj Raina

Types of Machine Learning

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve. Broadly Machine Learning can be categorized into four categories.

1. Supervised Learning
2. Unsupervised Learning

3.Reinforcement Learning

4.Semi-supervised Learning

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more. accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly.

Supervised Learning

Supervised Learning is a type of learning in which we are given a data set and we already know what correct output are should look like, having the idea that there is a relationship between the input and output. Basically, it is learning task of learning a function that maps an input to an output based on example input output pairs. It infers a function from labeled training data consisting of a set of training examples. Supervised learning problems are categorized

Unsupervised Learning

Unsupervised Learning is a type of learning that allows us to approach problems with little or no idea what our problem should look like. We can derive the structure by clustering the data based on a relationship among the variables in data. With unsupervised learning there is no feedback based on prediction result. Basically, it is a type of self-organized learning that helps in finding previously unknown patterns in data set without pre-existing label.

Reinforcement Learning

Reinforcement learning is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most

relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the illegal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best.

Semi-supervised Learning

Semi-supervised learning falls somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training-typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method can considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it/ learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

Because the nature of predictive value from the data set is continuous, we are using regression that is a type of supervised learning and all the other algorithms sharing same or higher priorities with regression. the predicted output shows a significance that all the algorithms work fine and predict similar kind of output with slight increase and decrease in accuracy and other parameters. Except linear regression in order to generate a desired output that is fine predicted value from continuous data we are using random forest regression, gradient boosting, Bayesian ridge and XGBOOST regressor.

Machine Learning Algorithms

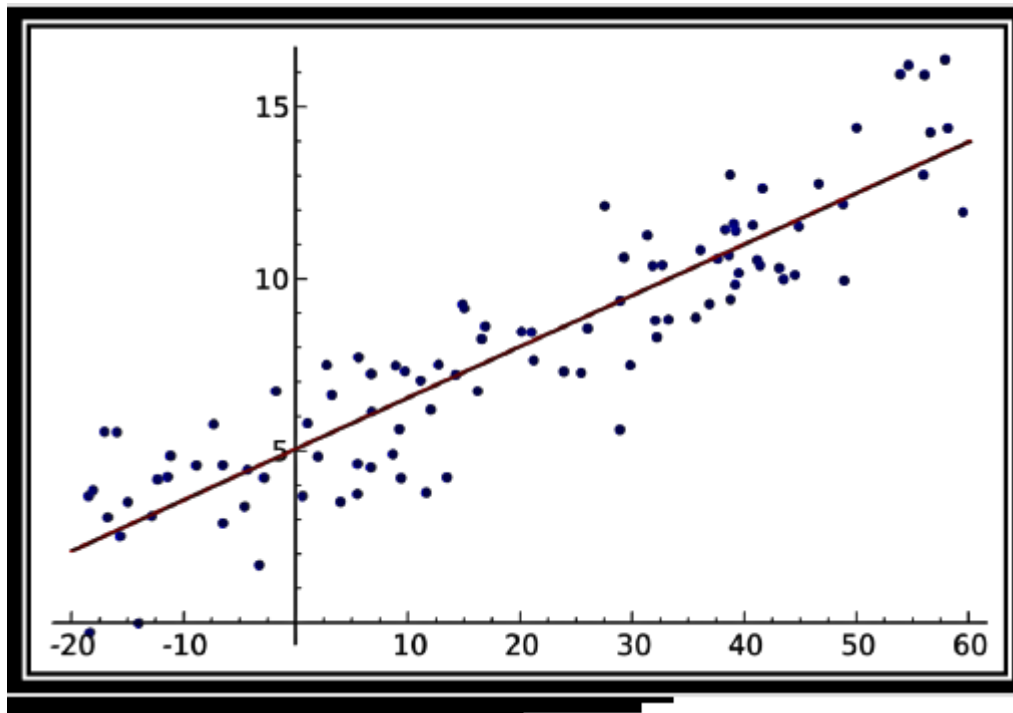
There are many types of Machine Learning Algorithms specific to different use cases. As we work with datasets, a machine learning algorithm works in two stages. We usually split the data around 20%-80% between testing and training stages. Under supervised learning, we split a dataset into a training data and test data in Python. Following are the Algorithms of Python Machine Learning -

1. Linear Regression

Linear regression is one of the supervised Machine learning algorithms in Python that observes continuous features and predicts an outcome. Depending on whether it runs on a single variable or on many features, we can call it simple linear regression or multiple linear regression. This is one of the most popular Python ML algorithms and often under-

appreciated. It assigns optimal weights to variables to create a line $ax+b$ to predict the output. We often use linear regression to estimate real values like several calls and costs of houses based on continuous variables. The regression line is

the best line that fits $Y=a*X+b$ to denote a relationship between independent and dependent variables.



Practical use of Linear Regression

To make a relationship between a dependent and independent variable linear regression is used. To predict the values an important parameter is used to calculate the score that is `lr.score()`.


```
[ ] # y = m * x + c
    from sklearn.linear_model import LinearRegression
    lr = LinearRegression()

[ ] lr.fit(X_train,y_train)

    LinearRegression()

[ ] lr.coef_

    array([[9.52842314]])

[ ] lr.intercept_

    array([2.97386968])

[ ] m = 3.93
    c = 50.44
    y = m * 4 + c
    y

    66.16

[ ] lr.predict([[5]])[0][0].round(2)

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  "X does not have valid feature names, but"
50.62

[ ] y_pred = lr.predict(X_test)
```

```
[ ] y_pred = lr.predict(X_test)
    y_pred
```

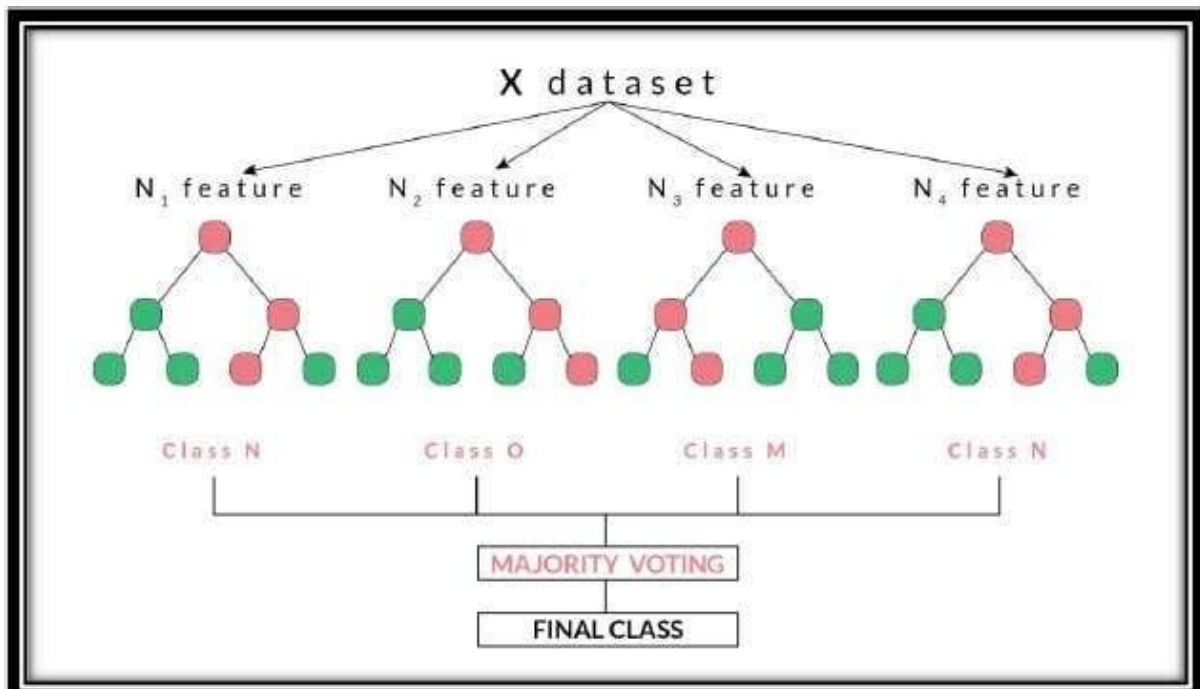
```
array([[98.06365722],
       [48.51820389],
       [17.88962451],
       [78.72483368],
       [66.84289002],
       [30.68252418],
       [72.62664287],
       [59.38211347],
       [78.72483368],
       [85.96643527],
       [93.68445802],
       [24.52716283],
       [82.15506601],
       [66.4316782],
       [54.92906005],
       [89.01553068],
       [91.287068],
       [65.7661782],
       [74.2464748],
       [65.09918858],
       [67.48129437],
       [93.58917379],
       [75.86630674],
       [56.88946045],
       [69.48226323],
       [77.48038286],
       [62.03189633],
       [4.38784892],
       [81.67864485],
       [67.29072591],
       [85.49001411],
       [43.56495227],
       [89.3966676],
```

```
lr.score(X_test,y_test)
```

```
0.8159074226676759
```

Random Forest

A random forest is an ensemble of decision trees. In order to classify every new object based on its attributes, trees vote for class-cache tree provides a classification. The classification with the most votes wins in the forest. 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_Forest_Regressor
from sklearn.ensemble import RandomForestRegressor
model1=RandomForestRegressor(n_estimators=500,bootstrap=True,max_depth=50,max_features=0.25,min_samples_leaf=7,min_samples_split=10)
model1.fit(X_train,y_train)
pred1=model1.predict(X_test)
print(pred1)
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to after removing the cwd from sys.path.

```
[89.01999174 23.61542184 12.15340879 78.0161767 76.59502321 15.90990009
74.0521094 66.77436188 78.0161767 84.51184401 89.485562 12.28296403
82.86198676 76.585138 69.31449674 88.14921079 89.22385746 70.23989541
66.56478588 67.65262931 76.39819471 89.48343381 65.28496575 63.84691601
72.993438 76.74948158 58.16571283 15.72824383 82.78431933 76.40244383
84.28774525 22.68229327 88.4939126 82.691842 12.17703007 15.72824383
11.9795074 73.88360355 67.39774082 89.40439472 70.74378728 12.24676426
65.16954097 82.72732095 76.36533941 73.16896885 67.41398543 70.83693363
69.0136058 89.43937657 17.68957589 87.33279981 71.13798087 68.2154397
78.0161767 89.20414276 58.22617079 58.32853376 87.16625622 88.89895575
15.72824383 89.24935461 15.72824383 83.85051443 67.41398543 67.63005442
74.14521642 88.94905545 83.88852678 84.28774525 76.78898637 82.37956671
22.34752838 15.40851713 85.34623897 65.0722074 89.02571369 70.75697847]
```

GRADIENT BOOSTING

Gradient boosting is a method standing out for its prediction speed and accuracy, particularly with large and complex datasets. From Kaggle competitions to machine learning solutions for business, this algorithm has produced the best results. We already know that errors play a major role in any machine learning algorithm. There are mainly two types of error, bias error and variance error. Gradient boost algorithm *helps us minimize bias error* of the model.

```
[ ] #Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor
model2= GradientBoostingRegressor()
model2.fit(X_train,y_train)
pred2=model2.predict(X_test)
print(pred2)
```

```
[88.96029698 22.05855254 10.04257593 78.05263588 75.40805015 16.55785709
76.02827335 70.77802876 78.05263588 84.27530343 89.10722767 12.61120688
82.84871446 75.40805015 69.66774076 88.82272453 89.00731611 52.96833715
66.5418645 69.33472786 75.40805015 89.10722767 58.8227082 67.28636454
74.36693758 78.31465309 50.70335963 9.66454189 82.84871446 75.40805015
84.12071886 22.16631746 88.882509 82.84871446 13.10334922 9.66454189
10.04257593 76.02827335 70.77802876 89.0909982 70.02722226 10.04257593
70.18326084 82.84871446 75.5505166 74.43793758 70.54747397 70.21407463
70.02722226 89.10722767 17.92269087 86.44078372 70.21407463 76.40765627
78.05263588 89.00731611 68.588907 50.70335963 86.44078372 88.91844482
9.66454189 89.0289001 9.66454189 83.88163047 70.54747397 69.33472786
76.02827335 88.91844482 83.88163047 84.12071886 78.31465309 81.85015775
22.16631746 9.66454189 85.56868495 59.77696039 88.96029698 70.02722226]
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:494: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y
y = column_or_1d(y, warn=True)
```

*HYPERPARAMETER TUNING *

BAYESIAN RIDGE

Bayesian ridge regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates. The output or response 'y' is assumed to drawn from a probability distribution rather than estimated as a single value. One of the most useful types of Bayesian regression is Bayesian Ridge regression which estimates a probabilistic model of the regression problem. Here the prior for the coefficient w is given by spherical Gaussian as follows $p(w|\lambda) = N(w|0, \lambda^{-1}I_p)$

This resulting model is called Bayesian Ridge Regression and in scikit-learn **sklearn.linear_model.BayesianRidge** module is used for Bayesian Ridge Regression.

```
#Bayesian Ridge
from sklearn.linear_model import BayesianRidge
model3=BayesianRidge()
model3.fit(X_train,y_train)
pred3=model3.predict(X_test)
print(pred3)
```

```
[90.0289563 48.53602818 17.88016612 78.70664847 66.8420116 30.73431561
 72.61734005 59.39212334 78.70664847 85.93770221 93.64448317 24.58791993
 82.13188445 66.43288619 55.01543291 88.98235642 91.17070163 65.76686808
 74.2348126 65.10084998 67.47948608 93.54933773 75.85228515 56.82319635
 69.4775404 77.3841268 62.03716668 4.39805671 81.65615723 67.28919519
 85.46197499 43.59797963 89.36293819 80.41926646 21.65744025 7.89940905
 18.56521332 72.14161283 58.62095979 92.50273785 52.73194226 16.45298446
 55.77659647 80.70470279 68.05035874 70.61928573 58.15523257 54.34941481
 51.3047606 92.78817418 37.23274943 88.60177464 53.87368759 76.32801237
 78.70664847 90.98041074 61.79930307 62.43677755 88.22119287 89.64837453
 7.09067277 91.26584707 8.29901991 82.893048 58.15523257 65.00570453
 72.36044735 89.93381086 83.08333889 85.46197499 78.04063036 80.03868468
 42.36108886 10.40173423 87.55517476 75.56684882 90.12410175 52.92223315]
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape
y = column_or_1d(y, warn=True)
```

XGBOOST REGRESSOR

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance. Extreme Gradient Boosting (XGBoost) is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm.

Shortly after its development and initial release, XGBoost became the go-to method and often the key component in winning solutions for a range of problems in machine learning competitions.

Regression predictive modeling problems involve predicting a numerical value such as a dollar amount or a height. **XGBoost** can be used directly for **regression predictive modeling**.

```
XGBRegressor
```

```
[47] import xgboost
      print(xgboost.__version__)

0.90
```

```
[48] from xgboost.sklearn import XGBRegressor
      from pandas import read_csv
      from numpy import asarray
```

```
df2.head()
```

	study_hours	student_marks
0	6.830000	78.50
1	6.560000	76.74
2	6.608545	78.68
3	5.670000	71.82
4	8.670000	84.19

```
[51] from sklearn.model_selection import RepeatedKFold

dataframe=read_csv("/content/dset.csv",header=0)
data=dataframe.values
#split dataset
A, b=data[:, :-1],data[:, -1]
tech=XGBRegressor()
#define model evaluation method
cv= RepeatedKFold(n_splits=10,n_repeats=3,random_state=1)
```

```
College_students_mark_predictor(1).ipynb - Collaboratory  
← → ↺ 🏠 🔍 https://colab.research.google.com/drive/1dDnTOIKVCMq7-QSvmWB2OEdeQ8bm1U?scrollTo=DDjY5lKx29sx A 🔍 ⚙️ ⌂ 👤 ...  
  
🔗 College_students_mark_predictor(1).ipynb ☆  
File Edit View Insert Runtime Tools Help All changes saved  
  
+ Code + Text RAM Disk Editing  
  
[52] # summarize shape  
print(dataframe.shape)  
# summarize first few lines  
print(dataframe.head())  
  
(387, 2)  
study_hours student_marks  
0      6.83      78.50  
1      6.56      76.74  
2      NaN      78.68  
3      5.67      81.82  
4      6.67      84.19  
  
[53] #fitting model  
tech.fit(A, b)  
  
[04:01:06] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.  
XGBRegressor()  
  
[54] from sklearn.model_selection import train_test_split  
  
A_train, A_test, b_train, b_test = train_test_split(A, b)  
tech = XGBRegressor()  
tech.fit(A_train, b_train)  
pred5 = tech.predict(A_test)  
print(pred5)  
  
[04:01:10] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.  
[[12.418741 82.826706 76.71983 84.639305 89.02195 89.02195 72.02018  
88.9249 72.0154 9.958003 89.02195 71.61751 88.9249 83.25523  
70.02389 83.25523 2.865538 72.0154 89.02195 34.48013 84.10802  
84.65437 82.38389 84.584684 70.276276 70.391235 88.81256 74.03107  
78.67732 12.418741 89.02195 18.328934 84.17231 68.178604 89.02195  
71.15669 88.85453 21.858252 89.02195 88.987976 74.03107 83.25523  
88.81256 85.91288 89.02195 22.483868 71.72464 88.96712 73.61937  
70.03389 71.15669 74.437096 89.02195 84.584684 71.72464 17.816373  
89.02195 72.02018 73.61937 89.02195 68.178604 71.15669 74.03107  
70.03389 8.000000 10.000000 10.000000 10.000000 11.000000 12.000000]]
```

```
+ Code + Text
File Edit View Insert Runtime Tools Help All changes saved
RAM
Disk
Comment Share Settings
[54]
76.041309 71.130819 74.451070 89.02195 89.368004 71.76804 11.815313
89.02195 72.02018 73.61937 89.02195 68.178604 71.15669 74.83187
70.02389 8.865538 10.073894 89.02195 9.491025 21.858252 45.10267
70.34099 68.49199 85.984205 82.84858 68.178604 89.02195 68.49199
89.02195 83.25523 78.922615 70.35848 83.25523 51.007587 8.865538
88.90212 57.128674 15.128934 85.384205 74.437096 71.61751 9.95745
70.35848 74.437096 84.504684 70.02389 74.437096 9.491025

[55] # define new data
row = [8]
new_data = asarray([row])
# make a prediction
yhat = tech.predict(new_data)
# summarize prediction
print('Predicted: %.3f' % yhat)

Predicted: 81.811

[56] r2_score=r2_score(b_test,pred5)
variance_score=explained_variance_score(b_test,pred5)
mean_absolute_error=mean_absolute_error(b_test,pred5)
mean_squared_log_error=mean_squared_log_error(b_test,pred5)

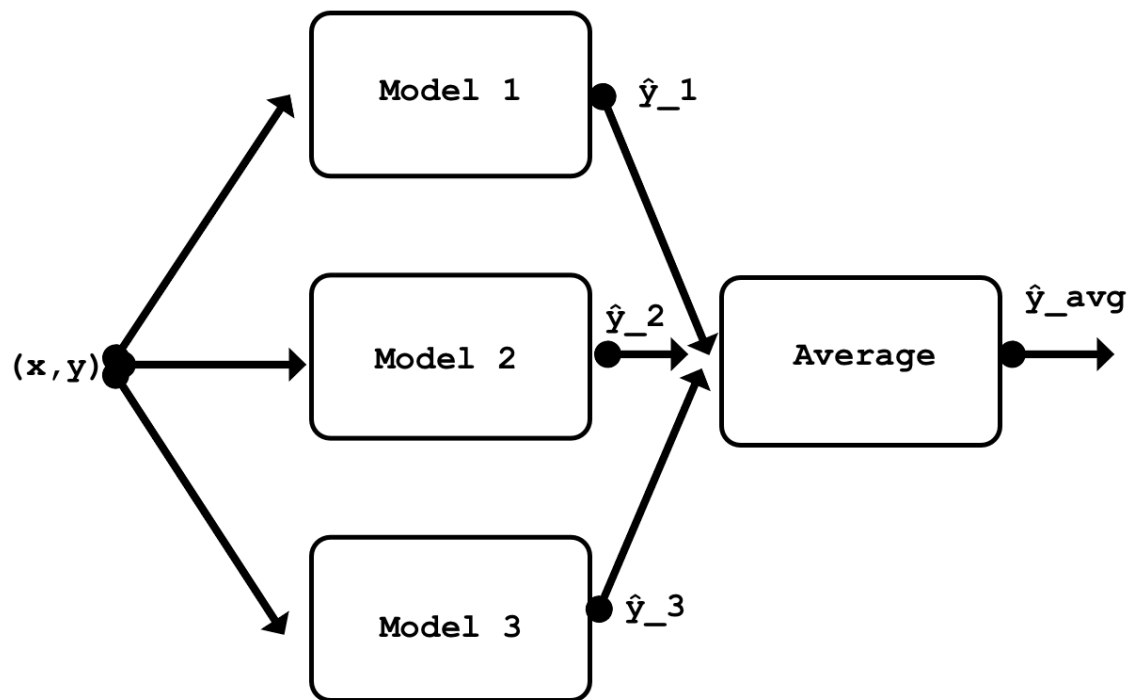
[57] #printing the values
print("EXTREME GRADIENT BOOSTING Regressor Report")
print("-> R2 Score:",r2_score)
print("->mean absolute error:",mean_absolute_error)
print("->variance_score:",variance_score)
print("-> mean squared log error:",mean_squared_log_error)

EXTREME GRADIENT BOOSTING Regressor Report
-> R2 Score: 0.7670594822280403
->mean absolute error: 5.758534941525803
->variance_score: 0.7703124288956388
-> mean squared log error: 0.0856883709050551

0s completed at 09:31
```

ENSEMBLE MODEL ACCURACY

A single algorithm may not make the perfect prediction for a given dataset. Machine learning algorithms have their limitations and producing a model with high accuracy is challenging. If we build and **combine** multiple models, the overall accuracy could get boosted. The combination can be implemented by aggregating the output from each model with two objectives: reducing the model error and maintaining its generalization. The way to implement such aggregation can be achieved using some techniques. Some textbooks refer to such architecture as *meta-algorithms*.



```
#accuracy
from sklearn.metrics import r2_score
ensemble_prediction=(pred1*0.4+pred2*0.4
                    +pred3*0.1+pred4*0.1)
r2_score_ensemble=r2_score(y_test,ensemble_prediction)
print("Ensemble MoDEL Accuracy")
print(r2_score_ensemble)
```

Advantages and Disadvantages

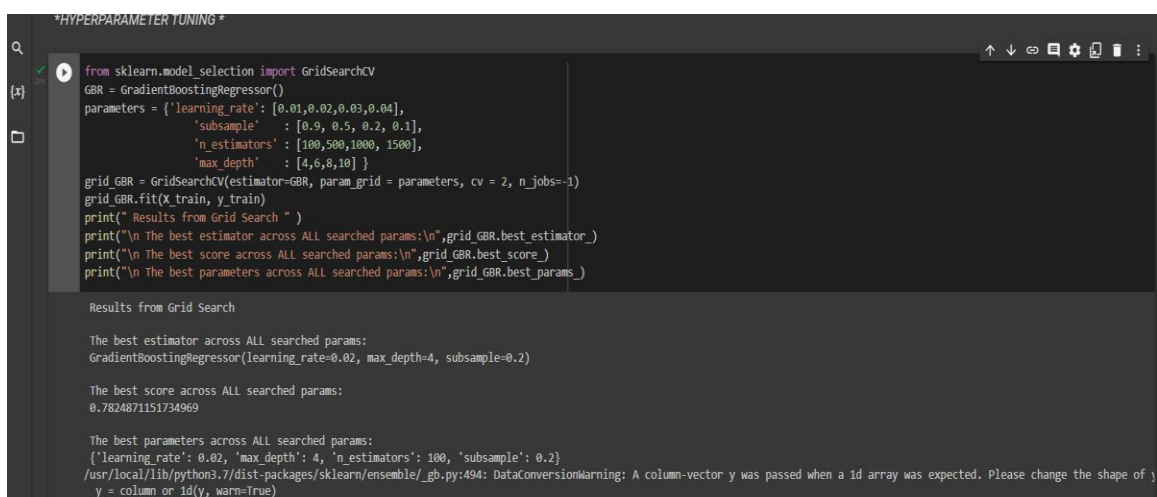
Advantages of Gradient Boosting in our model are it provides predictive accuracy that cannot be trumped, and it can optimize on different loss functions and provides several hyperparameter tuning options that make the function fit very flexible. And also, no data pre-processing required - often works great with categorical and numerical values as is. Let's also take advantages of logistic regression its advantages are it is easier to implement, interpret, and very efficient to train it makes no assumption about distribution of glasses in feature space it can easily excel to multiple classes and a natural prognostic view of glass prediction it not only provides a pleasure of how appropriate a predictor but also its direction of association whether positive or negative it is very fast at classifying unknown record accuracy for many simple dataset and it performs well when the dataset linearly separable it can interpret model coefficients as indicators of future important logistic regression is less land overheating but it can over hi dimensional data cells one may consider regularisation L1 and L2 techniques to avoid overfitting these scenarios.

Now let's see of the disadvantages of the algo and techniques used to build this model. Gradient Boosting Models will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting. computationally expensive - often require many trees (>1000) which can be time and memory exhaustive. the high flexibility results in many parameters that interact and influence heavily the behaviour of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning. Let's see some of the disadvantages of using linear regression, If the number of observations is lesser than the number of features, Logistic Regression should not be used, otherwise, it may lead to overfitting. The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. Non-linear problems can't be solved with logistic regression because it has a linear decision surface. Linearly separable data is rarely found in real-world scenarios. It is tough to obtain complex relationships using logistic regression. More powerful and compact algorithms such as Neural Networks can easily outperform this algorithm.

CONCEPT OF HYPERPARAMETER TUNING

There are various machine learning models which are all different in some way or the other but what varies them from each other is input parameters, these parameters are known as hyper parameters. the purpose of hyper parameter is to define the architecture of the model and moreover the best part is that you get a choice to select these for your model. Most of the times we don't have knowledge of optimal values for hyper parameters which would generate the best model output. So, to get the best desired output we let it explore and select the optimal model architecture automatically. This phenomenon of selection procedure for hyper parameter is known as hyper parameter tuning. to predict the future values, we use the concept of regression in machine learning and to select the best parameters we use hyper parameter tuning that uses the optimal model architecture. Hyperparameter tuning is one of the most important steps in machine learning. As the ML algorithms will not produce the highest accuracy out of the box. You need to tune their hyperparameters to achieve the best accuracy.

I have used grid search CV. CV stands for cross validation. grid search CV tries all the exhaustive combinations of parameter value supplied by you and chooses the best out of it. consider below example if you are providing a list of values to try all hyper parameters then it will try all possible combinations.



```
*HYPERPARAMETER TUNING*

from sklearn.model_selection import GridSearchCV
GBR = GradientBoostingRegressor()
parameters = {'learning_rate': [0.01, 0.02, 0.03, 0.04],
              'subsample' : [0.9, 0.5, 0.2, 0.1],
              'n_estimators': [100, 500, 1000, 1500],
              'max_depth' : [4, 6, 8, 10] }

grid_GBR = GridSearchCV(estimator=GBR, param_grid = parameters, cv = 2, n_jobs=-1)
grid_GBR.fit(X_train, y_train)
print(" Results from Grid Search ")
print("\n The best estimator across ALL searched params:\n", grid_GBR.best_estimator_)
print("\n The best score across ALL searched params:\n", grid_GBR.best_score_)
print("\n The best parameters across ALL searched params:\n", grid_GBR.best_params_)

Results from Grid Search

The best estimator across ALL searched params:
GradientBoostingRegressor(learning_rate=0.02, max_depth=4, subsample=0.2)

The best score across ALL searched params:
0.7824871151734969

The best parameters across ALL searched params:
{'learning_rate': 0.02, 'max_depth': 4, 'n_estimators': 100, 'subsample': 0.2}
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:494: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y
y = column_or_1d(y, warn=True)
```

Result and Comparative Analysis

Linear regression is used to study the linear relationship between a dependent variable Y (students marks) and one or more independent variables X (study hour).

The dependent variable Y must be continuous, while the independent variables may be either continuous, binary, or categorical. The initial judgment of a possible relationship between two continuous variables should always be made based on a scatter plot (scatter graph). This type of plot will show whether the relationship is linear (figure 1) or nonlinear (figure 2).

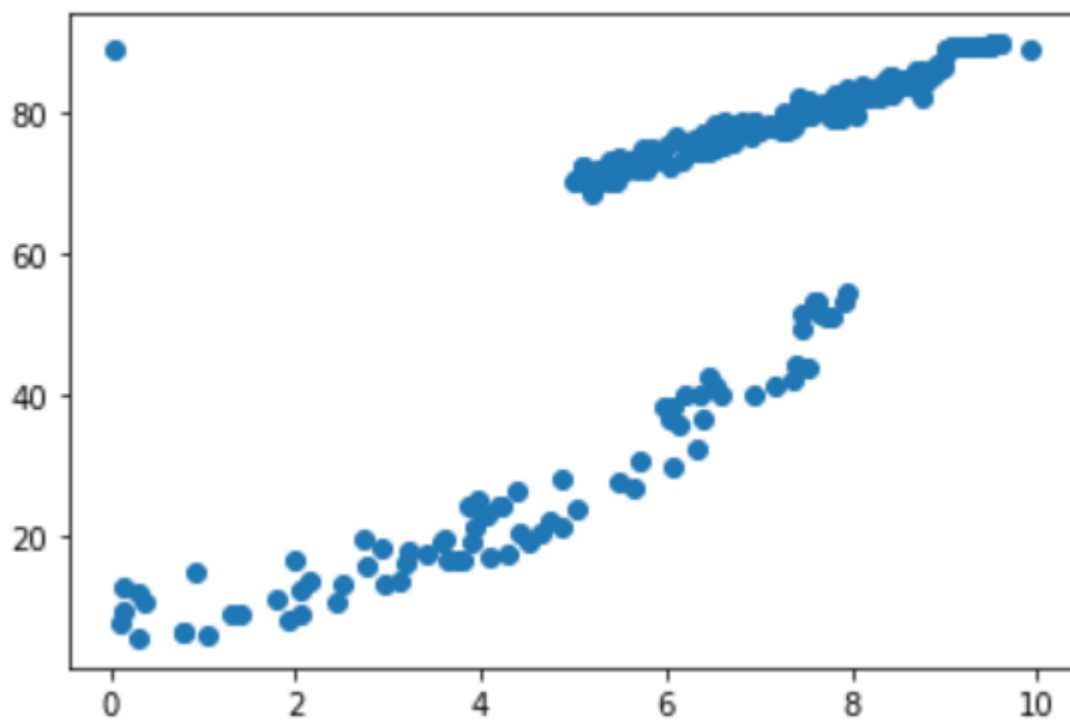


Figure 1

A scatter plot showing linear relationship.

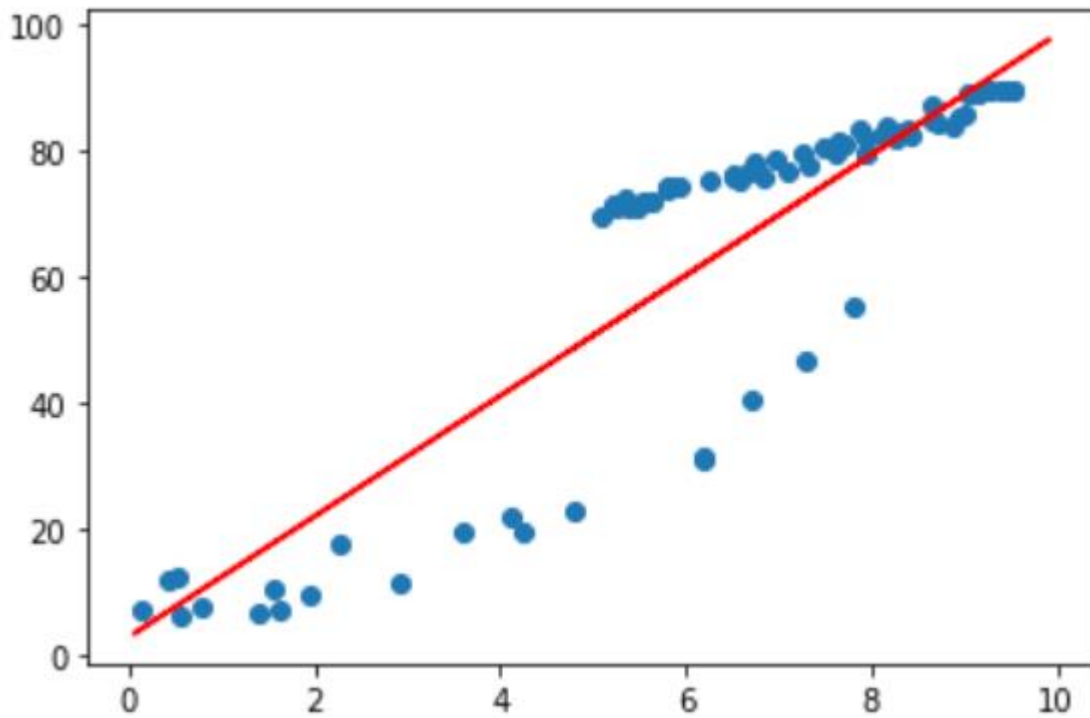


Figure 2

A scatter graph showing exponential relationship. In this case, it would not be appropriate to compute a coefficient of determination or a regression line.

Performing a linear regression makes sense only if the relationship is linear. Other methods must be used to study nonlinear relationships. The variable transformations and other, more complex techniques that can be used for this purpose will not be discussed in this article.

As shown below the predicted data is in simpler format. It is showing how our model is predicting the students marks according to the given marks obtained with respect to the study time.

WORKING

The first step is to import important libraries which include numpy, pandas and matplotlib. The main aim of Numpy is to work with arrays and domain of linear algebra pandas is to analyse data and to manipulate the data. matplotlib is used for graph plotting.

```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

The second step includes loading the data set

```
path = r"/content/dset.csv"
df = pd.read_csv(path)
```

head and tail functions are used to check whether right data is present or not and returns first n number and last n number of rows

```
df.head()
df.tail()
```

To check the shape in particular the number of rows and columns present in data set shape function is used

```
df.shape
```

To discover and raise the data to gain insights info function is used that gives the non-null count and data type and column description

```
df.info()
```

In order to get the brief of your ML model describe function is used to calculate count mean standard deviation minimum value less than 25% value less than 50% values less than 75%

```
df.describe()
```

In order to establish relationship between variables through graph scatter function is used:

```
plt.scatter(x=df.study_hours, y=df.student_marks)
plt.xlabel("Students Study Hours")
plt.ylabel("Students marks")
plt.title("Scatter Plot of Students Study Hours vs Students marks")
plt.show()
```

Data cleaning has been done, the main idea of data cleaning is to identify and remove errors and duplicate data as well, in order to create a reliable data set and to prepare the data for machine learning algorithms.

```
df.isnull().sum()
```

```
df.mean()
```

```
df2 = df.fillna(df.mean())
```

Splitting of the data set is done to train and test because user has only access to training data and the ratio of training and testing dataset is split into 80% to 20%.

```
X = df2.drop("student_marks", axis = "columns")
y = df2.drop("study_hours", axis = "columns")
print("shape of X = ", X.shape)
print("shape of y = ", y.shape)
```

```
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=51)
print("shape of X_train = ", X_train.shape)
print("shape of y_train = ", y_train.shape)
print("shape of X_test = ", X_test.shape)
print("shape of y_test = ", y_test.shape)
```

Linear regression has been used for the predictive analysis of the desired output, As it is a linear approach to establish relationship between a scalar response and one or more explanatory variables for one variable it is simple linear regression for more than one it is multiple linear regression. Basically used to predict the value of variable from another variable.

```
# y = m * x + c
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

```
lr.fit(X_train, y_train)
```

```
lr.predict([[5]])[0][0].round(2)
```

After fitting the model fine tuning has been done to achieve the desired output by making small adjustments and calculating the score that tell us the accuracy of algorithm used

```
lr.score(X_test, y_test)
```

The predicted values and the original values are shown in graph as below:

```
pd.DataFrame(np.c_[X_test, y_test, y_pred], columns = ["study_hours", "student_marks_original", "student_marks_predicted"])
```

	study_hours	student_marks_original	student_marks_predicted
0	9.140	89.140	90.063657
1	4.779	22.701	48.510204
2	1.557	10.429	17.809625
3	7.950	82.030	78.724834
4	6.703	40.602	66.842890
...
73	0.771	7.892	10.320284
74	8.880	83.640	87.586267
75	7.620	79.530	75.580454
76	9.150	89.150	90.158941
77	5.240	70.780	52.902807

78 rows x 3 columns

As we can see there are 78 Rows and 3 columns shown in figure active presidents the study of student marks original and student marks predicted after using algorithm. somewhere we find that the original Marks and predicted marks are almost same but in various cases after hybridising the data set there is a slight difference in predicted values and original values and the accuracy dipped too from 95% to 89%.

As confusion matrix and accuracy somewhere is used for classification problems. So, for my model I used some other techniques to get predicted values with different algorithms and to compare them with each other for that I used ensemble model learning method which is used by Random Forest Regressor. What is ensemble learning it is a technique that combines predicted prediction for multiple machines running algorithms.

But for regression problems confusion matrix can be made by converting continuous output and using cut off variable but in my case, there is only one class present show the output of confusion matrix that I will get after converting continuous data is one element in an array. the below is the description of confusion matrix and how to convert the continuous data.

```
cutoff=0.7
y_pred_classes=np.zeros_like(y_pred)
y_pred_classes[y_pred>cutoff]=1
y_test_classes=np.zeros_like(y_pred)
y_test_classes[y_pred>cutoff]=1
from sklearn.metrics import confusion_matrix
confusion_matrix=confusion_matrix(y_test_classes,y_pred_classes)
print(confusion_matrix)
```

OUTPUT - [[78]]

At last, we are using ensemble model accuracy which we got after using various techniques. I am getting accuracy around 0.89 which is 89%.

To get the accuracy for each technique I used so for that I am calculating R2 score, variance score, mean absolute error, mean squared log error.

```
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import explained_variance_score
r2_score1=r2_score(y_test,pred1)
variance_score1=explained_variance_score(y_test,pred1)
mean_absolute_error1=mean_absolute_error(y_test,pred1)
mean_squared_log_error1=mean_squared_log_error(y_test,pred1)

r2_score2=r2_score(y_test,pred2)
variance_score2=explained_variance_score(y_test,pred2)
mean_absolute_error2=mean_absolute_error(y_test,pred2)
mean_squared_log_error2=mean_squared_log_error(y_test,pred2)

r2_score3=r2_score(y_test,pred3)
variance_score3=explained_variance_score(y_test,pred3)
mean_absolute_error3=mean_absolute_error(y_test,pred3)
mean_squared_log_error3=mean_squared_log_error(y_test,pred3)

r2_score4=r2_score(y_test,pred4)
variance_score4=explained_variance_score(y_test,pred4)
mean_absolute_error4=mean_absolute_error(y_test,pred4)
mean_squared_log_error4=mean_squared_log_error(y_test,pred4)
```

R2 score is just to evaluate the performance for regression what I have learned is about 0.2910 your model is cool various code is how far from the actual value from the average of predicted values we should not be less than 60% greater than 60% 10% it is how close are printed lines 2 data points means the lower the mean squared log error the higher the accuracy.

After printing the R2 score, mean absolute error, variance score and mean squared log error for each algorithm we can see all the algorithms are getting different values and among them Random Forest regressor's report is the best which is “

```
Ranom Forest Regressor Report
-> R2 Score: 0.8969636583812532
->mean absolute error: 5.181152184684641
->variance_score: 0.8970639329275524
-> mean squared log error: 0.06055997246944537
```

```
Gradient Boosting Report
-> R2 Score: 0.8851455670055234
->mean absolute error: 4.8197961791467545
->variance_score: 0.8852157300620376
-> mean squared log error: 0.03855770404561922
```

```
Bayesian Ridge Report
-> R2 Score: 0.815825684186779
->mean absolute error: 8.888554638743011
->variance_score: 0.8159972494994563
-> mean squared log error: 0.10787939404705035
```

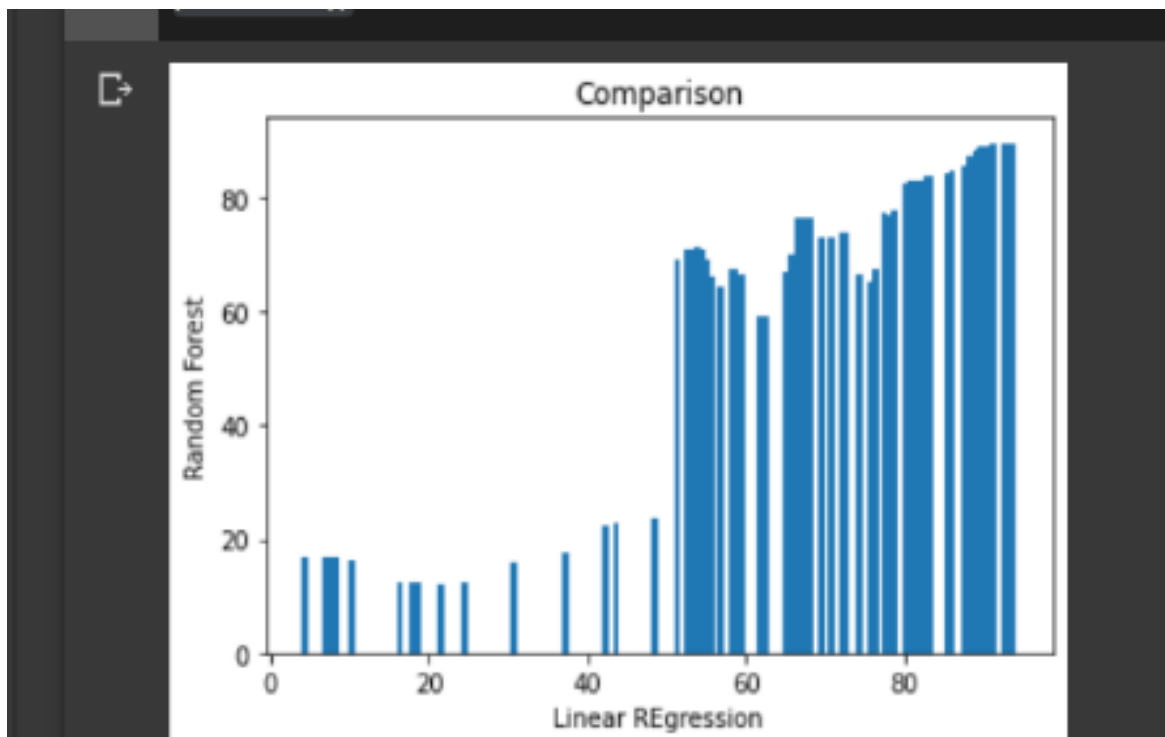
```
Linear Regression Report
-> R2 Score: 0.8159074226676759
->mean absolute error: 8.889062220544739
->variance_score: 0.8160810014668257
-> mean squared log error: 0.10784519850935857
```

```
EXTREME GRADIENT BOOSTING Regressor Report
-> R2 Score: 0.7670594822220403
->mean absolute error: 5.758534941525803
->variance_score: 0.7703124288956388
-> mean squared log error: 0.08556883709050551
```

after using different algorithms, I chose linear regression and random forest regressor so that I can compare them according to the predicted values.

The comparison between linear regression and random forest regression is shown as:

```
plt.bar(pred4,pred1)
plt.title('Comparison')
plt.xlabel('Linear REgression')
plt.ylabel('Random Forest')
plt.show()
```



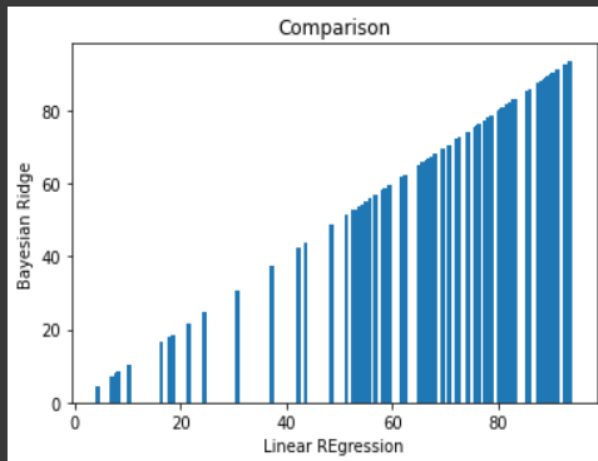
Likewise the old comparison reason has been made in between linear regression and Bayesian ridge.

```
plt.bar(pred4,pred3)
plt.title('Comparison')
plt.xlabel('Linear REgression')
plt.ylabel('Bayesian Ridge')
```

```
plt.show()
```

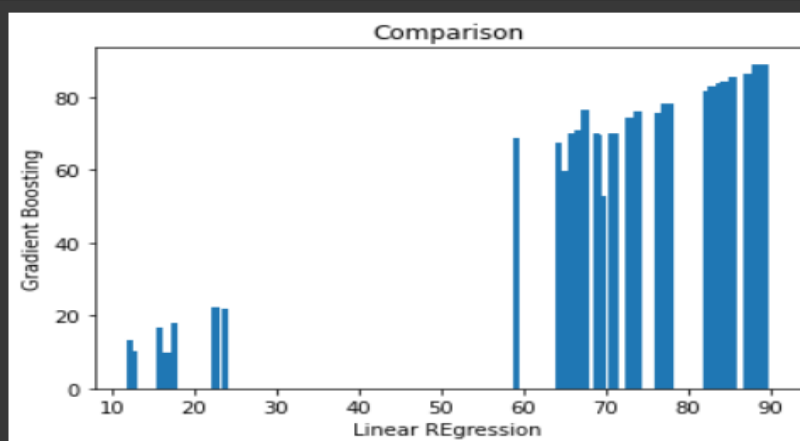


```
plt.bar(pred4,pred3)
plt.title('Comparison')
plt.xlabel('Linear REgression')
plt.ylabel('Bayesian Ridge')
plt.show()
```



Another comparison has been made in between linear regression and gradient boosting

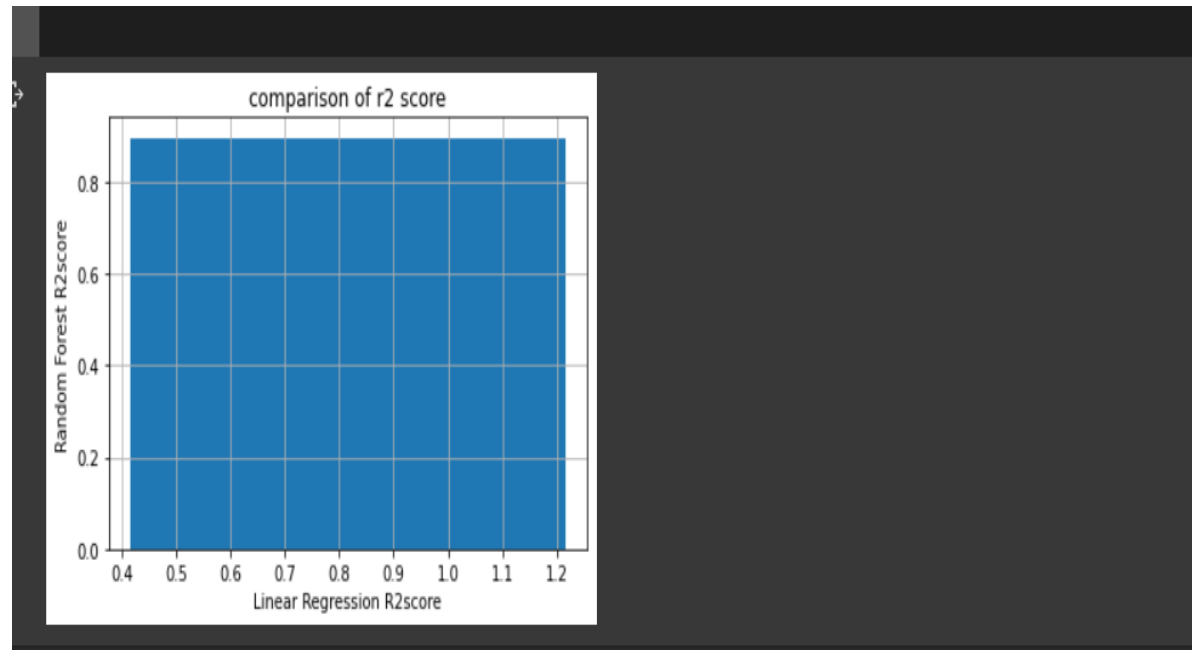
```
44] plt.bar(pred1,pred2)
plt.title('Comparison')
plt.xlabel('Linear REgression')
plt.ylabel('Gradient Boosting')
plt.show()
```



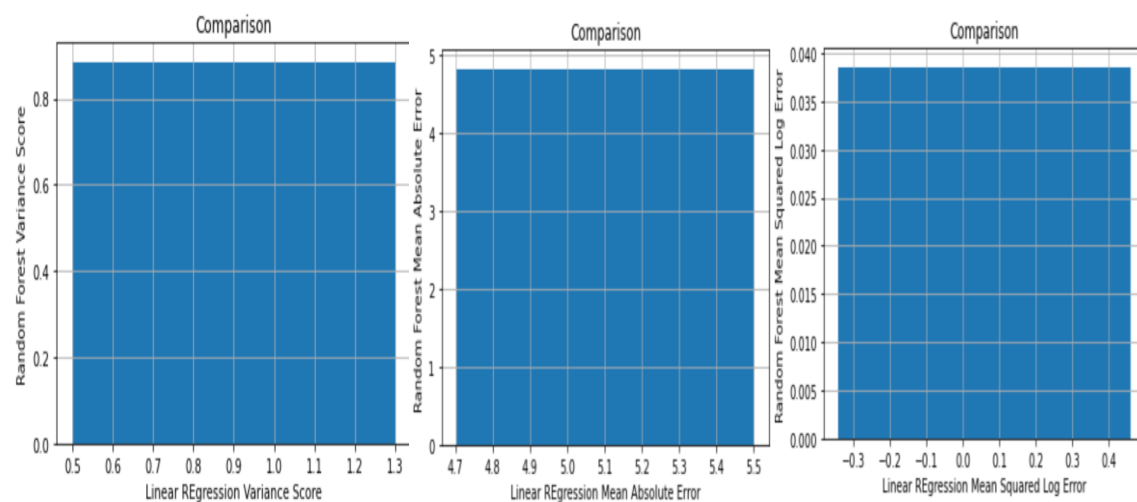
Here you can see the comparison between them as according to their predicted values.

After that I compared linear regression and random forests R2 score and shown below is what I got.

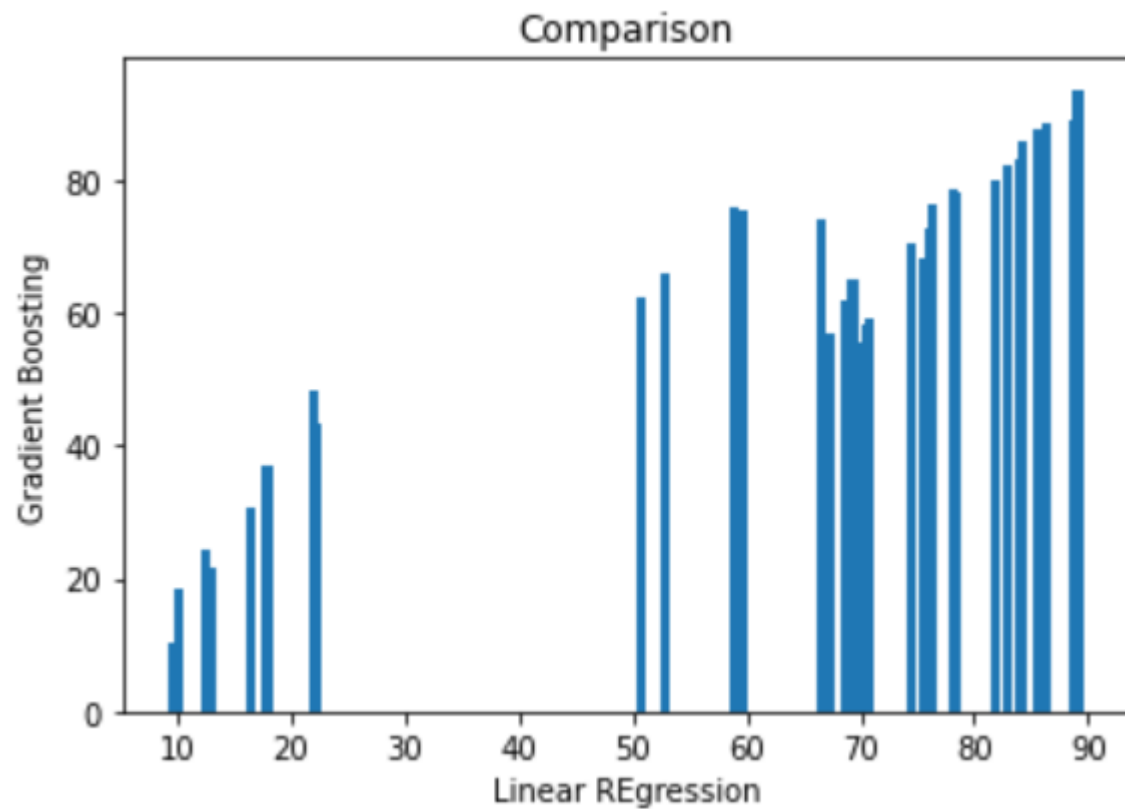
```
a=r2_score1
b=r2_score4
plt.title("comparison of r2 score")
plt.bar(b,a)
plt.xlabel("Linear Regression R2score")
plt.ylabel("Random Forest R2score")
plt.grid()
```

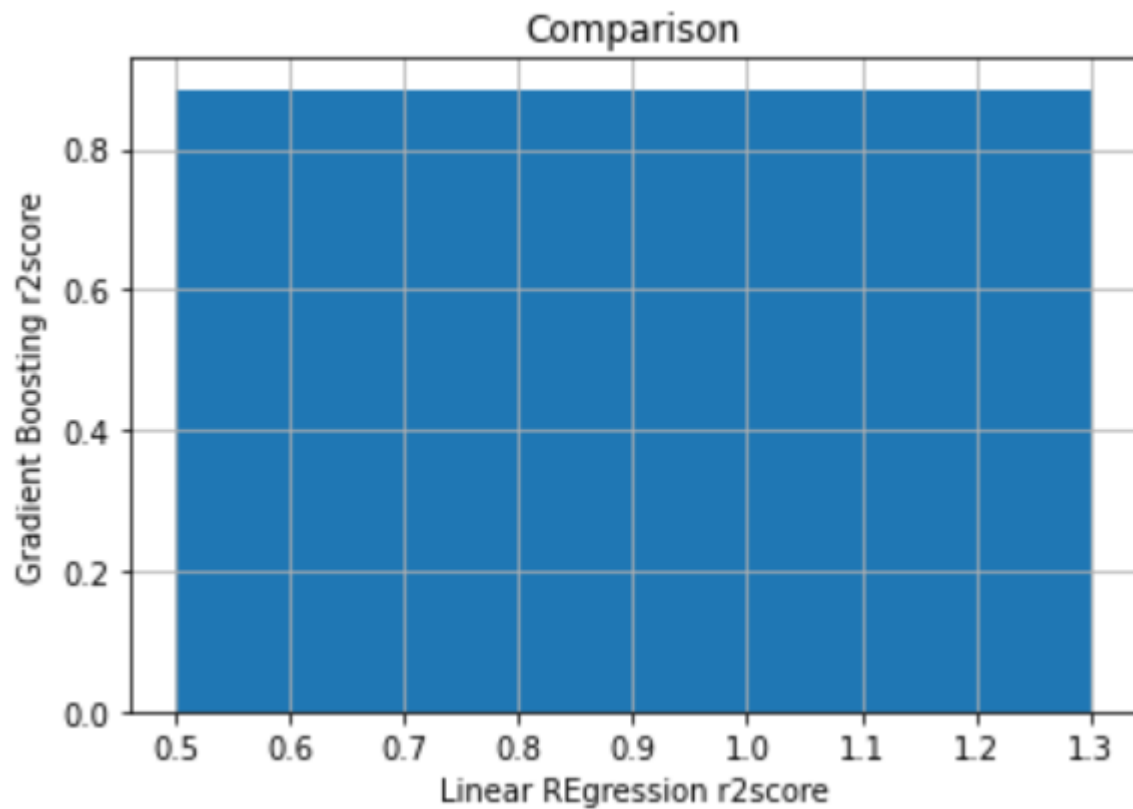


After showing comparison between linear regression and random forest on the basis of predicted values and R2 score now we also gonna compare them on the basis of their variance score, mean absolute error and mean squared log error. Given below are the bar graphs pics showing the comparison.



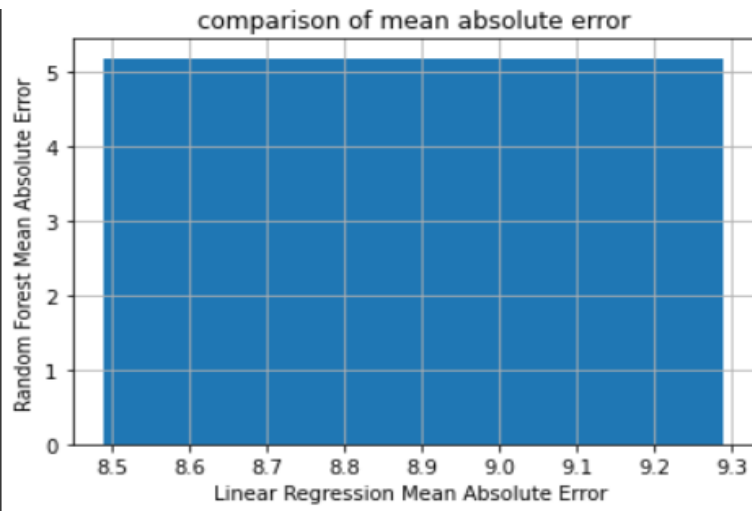
After that I showed comparison between Linear regression and Gradient boosting regressor based on their predicted values and their R2 score. Given below first picture show the comparison based on their predicted values and in the second one it shows the comparison based on their R2 score.





After showing comparison between linear regression and Gradient boosting regressor based on predicted values and R2 score now we also going to compare them on the basis of their variance score, mean absolute error and mean squared log error. Given below are the bar graphs pics showing the comparison.

```
c=mean_absolute_error1
d=mean_absolute_error4
plt.title("comparison of mean absolute error")
plt.bar(d,c)
plt.xlabel("Linear Regression Mean Absolute Error ")
plt.ylabel("Random Forest Mean Absolute Error")
plt.grid()
plt.show()
```



To calculate the accuracy of our model we used Ensemble model accuracy and got around 89%

```
#accuracy
from sklearn.metrics import r2_score
ensemble_prediction=(pred1*0.4+pred2*0.4
                    +pred3*0.1+pred4*0.1)
r2_score_ensemble=r2_score(y_test,ensemble_prediction)
print("Ensemble MoDEL Accuracy")
print(r2_score_ensemble)
```

```
Ensemble MoDEL Accuracy
0.895466030675074
```

Other than linear regression various other algorithms are used to predict the output, one of the are as under:

XGBOOST REGRESSOR

XGBoost stands for "Extreme Gradient Boosting" and it is an implementation of gradient boosting trees algorithm. The XGBoost is a popular supervised machine learning model with characteristics like computation speed, parallelization, and performance. Extreme Gradient Boosting (XGBoost) is an open-source library that provides an efficient and effective implementation of the gradient boosting algorithm.

Shortly after its development and initial release, XGBoost became the go-to method and often the key component in winning solutions for a range of problems in machine learning competitions.

Regression predictive modeling problems involve predicting a numerical value such as a dollar amount or a height. **XGBoost** can be used directly for **regression predictive modeling**.

```
import xgboost
print(xgboost.__version__)
```

```
0.90
```

```
from xgboost.sklearn import XGBRegressor
from pandas import read_csv
from numpy import asarray
```

```
df2.head()
```

```
from sklearn.model_selection import RepeatedKFold

dataframe=read_csv("/content/dset.csv",header=0)
data=dataframe.values
#split dataset
A, b=data[:, :-1],data[:, -1]
tech=XGBRegressor()
#define model evaluation method
cv= RepeatedKFold(n_splits=10,n_repeats=3,random_state=1)
```

```
# summarize shape
print(dataframe.shape)
# summarize first few lines
print(dataframe.head())
```

```
(387, 2)
  study_hours  student_marks
0         6.83           78.50
1         6.56           76.74
2          NaN           78.68
3         5.67           71.82
4         8.67           84.19
```

```
#fitting model
tech.fit(A, b)
```

```
from sklearn.model_selection import train_test_split

A_train, A_test, b_train, b_test = train_test_split(A, b)
tech= XGBRegressor()
tech.fit(A_train,b_train)
pred5=tech.predict(A_test)
print(pred5)
```

```
define new data
row = [8]
new_data = asarray([row])
# make a prediction
yhat = tech.predict(new_data)
# summarize prediction
print('Predicted: %.3f' % yhat)
```

```
Predicted: 81.811
```


STREAMLIT (API)

An API is developed using STREAMLIT to showcase the practical evaluation of machine learning model. it takes input from user and generate a desired output that is a predicting value comparison to the original value present in the data set.

```
import streamlit as st
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

st.write("""
# College Student Marks Prediction App
This app predicts the **Marks** scored by student
""")

st.sidebar.markdown('''
    By Pankaj Raina \n
    GitHub Repository for the
    [App](https://github.com/venom005/College-Student-Marks-Prediction)
''')

st.write('''
    **Assuming the study hours to be maximum 10 hours.**
''')

def user_input():
    hr = st.number_input("Number of Hours of actual Study",0.00,10.00,6.2)
    hr = np.array([[hr]]).astype(np.float64)
    return hr

pred = user_input()

#Loading dataset
df = pd.read_csv(r"C:\Users\RAINA\Downloads\Data.txt")

attr = df.iloc[:, :-1].values
labels = df.iloc[:, 1].values

LR = LinearRegression()
LR.fit(attr, labels)

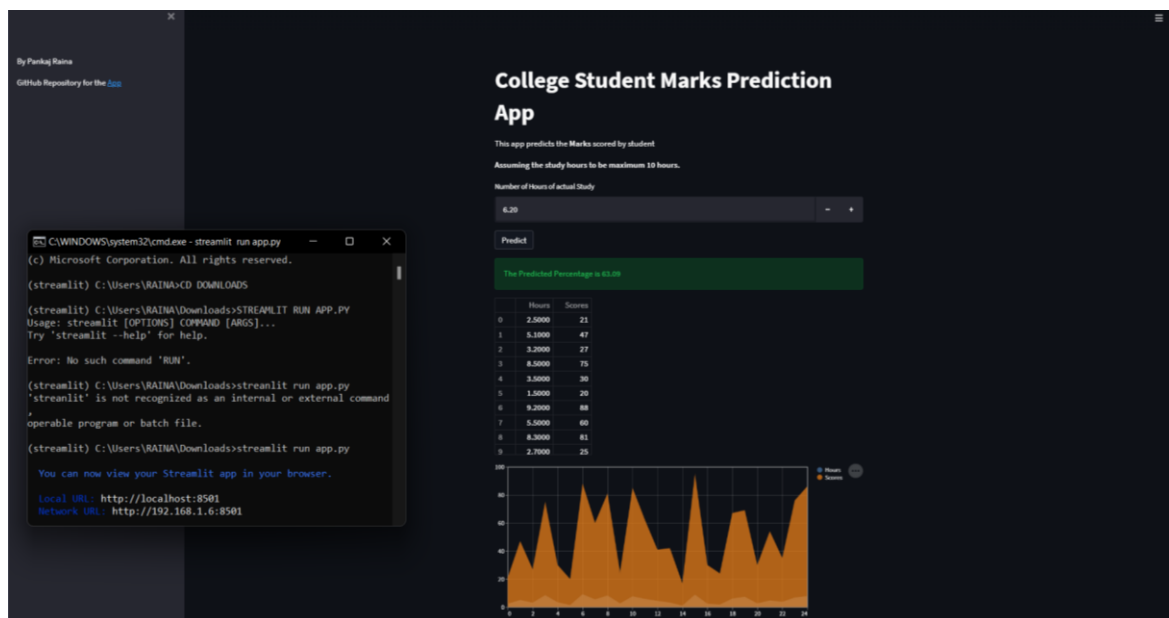
prediction = round(float(LR.predict(pred)),2)

if pred > 9.97:
```

```

    st.button("Predict")
    st.success("The Predicted Percentage is 100.00")
else:
    st.button("Predict")
    st.success("The Predicted Percentage is {}".format(prediction))
st.dataframe(df)
st.area_chart(df)

```



Again, the first priority is to import important libraries and machine learning algorithm for the prediction of the data. Write function is used to show the content on the screen.

SideBar. mark down is used to show the sidebar and content within it.

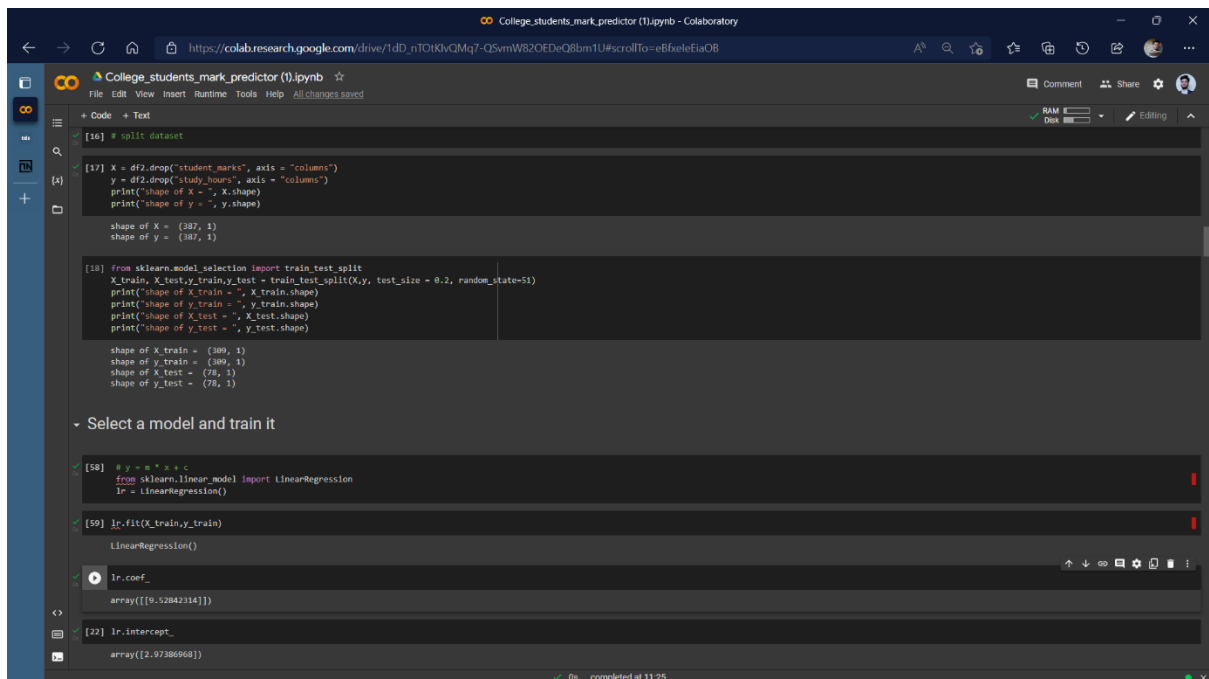
user input function is used to take input from the user in the form of array which takes value as hr and data type is float 64.

next step is to load the data set and provide the main attributes and to fit the linear regression model for the predictive analysis.

Prediction has been made using ground function to get the accurate prediction and condition is applied that is if a certain number of values entered then only you will get the desired output else you will get error.

To compare and to get the description data frame and area chart function has been used.

Pictures of Python File



```
[16] # split dataset

[17] x = df.drop('student_marks', axis = 'columns')
y = df.drop('study_hours', axis = 'columns')
print("shape of x = ", x.shape)
print("shape of y = ", y.shape)

shape of x = (387, 1)
shape of y = (387, 1)

[18] 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=51)
print("shape of x_train = ", X_train.shape)
print("shape of y_train = ", y_train.shape)
print("shape of X_test = ", X_test.shape)
print("shape of y_test = ", y_test.shape)

shape of X_train = (309, 1)
shape of y_train = (309, 1)
shape of X_test = (78, 1)
shape of y_test = (78, 1)

+ Select a model and train it

[58] # y = m * x + c
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

[59] lr.fit(X_train, y_train)
LinearRegression()

[60] lr.coef_
array([[9.52842314]])

[22] lr.intercept_
array([2.97386968])
```

0s completed at 11:25

College_students_mark_predictor (1).ipynb - Colaboratory

https://colab.research.google.com/drive/1dD_nT0KvQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=eBfxeleEiaOB

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[52.90288956]]

```

[26] pd.DataFrame(np.c_[X_test, y_test, y_pred], columns = ["study_hours", "student_marks_original", "student_marks_predicted"])

```

	study_hours	student_marks_original	student_marks_predicted
0	9.140	89.140	90.063657
1	4.779	22.701	48.510204
2	1.557	10.429	17.809625
3	7.950	82.030	78.724834
4	6.703	40.602	66.842890
...
73	0.771	7.892	10.320284
74	8.880	83.640	87.586267
75	7.620	79.530	75.580454
76	9.150	89.150	90.158941
77	5.240	70.780	52.902807

78 rows x 3 columns

Fine-tune your model

```

[27] lr.score(X_test, y_test)

e.8159874226676759

[28] cutoff=d.7
y_pred_classes=np.zeros_like(y_pred)
y_pred_classes[y_pred>cutoff]=1
y_test_classes=np.zeros_like(y_pred)
y_test_classes[y_pred>cutoff]=1
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test_classes, y_pred_classes)
matrix([[1, 0], [0, 1)])

```

0s completed at 11:25

College_students_mark_predictor (1).ipynb - Colaboratory

https://colab.research.google.com/drive/1dD_nT0KvQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=eBfxeleEiaOB

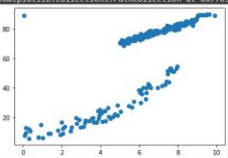
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```

[29] plt.scatter(X_train, y_train)

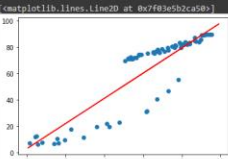
```



```

[30] plt.scatter(X_test, y_test)
plt.plot(X_train, lr.predict(X_train), color = "r")

```



```

[31] import joblib
joblib.dump(lr, "college_mark_predictor.pkl")

["college_mark_predictor.pkl"]

[32] model=joblib.load("college_mark_predictor.pkl")

```

0s completed at 11:25

```
College_students_mark_predictor (1).ipynb - Colaboratory
https://colab.research.google.com/drive/1dD_nT0kVQMq7-Q5mW82OEDeQ8bm1U#scrollTo=eBfeEiaO8

College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
[33] model.predict([[7]])[0]

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  "X does not have valid feature names, but"
69.67283169178535

[34] #linear regression
pred4 = lr.predict(X_test).sum(axis=1)

[35] #Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor
model11=RandomForestRegressor(n_estimators=500,bootstrap=True,max_depth=50,max_features=0.25,min_samples_leaf=7,min_samples_split=10)
model11.fit(X_train,y_train)
pred1=model11.predict(X_test)
print(pred1)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
after removing the cwd from sys.path.
[[89.84661351 23.79451515 12.45680793 77.90583114 76.54211508 15.77188316
 73.88551426 66.57932679 77.90583114 84.5488919 89.4852012 12.24967142
 82.86299006 76.4830378 69.11838973 88.21989692 89.22688064 69.7445639
 66.30541942 67.60802093 76.36448784 89.4839621 65.31595641 64.2770332
 72.84421081 77.18964717 59.83864491 16.76391785 82.81336801 76.381393
 84.38827867 22.71736579 88.56916834 82.65125331 12.15161841 16.76391785
 12.28438801 73.84842 67.42168084 89.39962469 78.60398311 12.61527872
 65.91494475 82.72322133 76.32267951 73.80791985 67.39309029 78.90635286
 68.95921397 89.4426719 17.48239212 87.26999889 71.19858271 67.4581649
 77.90583114 89.30053568 59.12325232 59.13798425 87.04545897 88.91608157
 16.76391785 89.24496329 16.76391785 83.82261697 67.39309029 66.98390714
 74.81694356 88.96808867 83.84529836 84.38827867 87.00356186 82.34457862
 22.35661794 16.21822713 65.40952716 65.08895158 89.87280814 70.71418499]]

[36] #Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor
model12= GradientBoostingRegressor()
model12.fit(X_train,y_train)
pred2=model12.predict(X_test)
print(pred2)

[[88.96029698 22.85855254 18.04257593 78.85263588 75.48895015 16.55785799
 76.82827335 70.77880287 78.85263588 84.27538343 89.18722767 12.61128688
 82.8481446 75.48895015 69.66774876 88.82272453 89.00711611 52.98833715
 66.54318495 69.33472785 75.48895015 89.18722767 58.8227892 67.29636454]]

0s completed at 11:25
```

```
College_students_mark_predictor (1).ipynb - Colaboratory
https://colab.research.google.com/drive/1dD_nT0kVQMq7-Q5mW82OEDeQ8bm1U#scrollTo=eBfeEiaO8

College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
*HYPERPARAMETER TUNING*

[37] from sklearn.model_selection import GridSearchCV
GBR = GradientBoostingRegressor()
parameters = {'learning_rate': [0.01,0.02,0.03,0.04],
              'subsample' : [0.9, 0.5, 0.2, 0.1],
              'n_estimators' : [100,500,1000, 1500],
              'max_depth' : [4,6,8,10] }
grid_GBR = GridSearchCV(estimator=GBR, param_grid = parameters, cv = 2, n_jobs=-1)
grid_GBR.fit(X_train, y_train)
print(" Results from Grid Search ")
print("\n The best estimator across ALL searched params:\n",grid_GBR.best_estimator_)
print("\n The best score across ALL searched params:\n",grid_GBR.best_score_)
print("\n The best parameters across ALL searched params:\n",grid_GBR.best_params_)

Results from Grid Search

The best estimator across ALL searched params:
GradientBoostingRegressor(learning_rate=0.02, max_depth=4, subsample=0.2)

The best score across ALL searched params:
0.7824871151734989

The best parameters across ALL searched params:
{'learning_rate': 0.02, 'max_depth': 4, 'n_estimators': 100, 'subsample': 0.2}
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_gb.py:494: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  y = column_or_1d(y, warn=True)

[38] #Bayesian Ridge
from sklearn.linear_model import BayesianRidge
model13=BayesianRidge()
model13.fit(X_train,y_train)
pred1=model13.predict(X_test)
print(pred1)

[[90.8289563 48.53682818 17.88816612 78.79664847 66.8420116 39.73431561
 72.61734065 59.39212334 78.79664847 85.93778221 93.64448317 24.58791993
 82.13188445 66.43288619 55.81543291 88.98235642 91.17878163 65.76686888
 74.2348126 65.38088998 67.47848688 93.54933772 75.85228515 56.82319635
 69.4775404 77.3841268 62.83716668 4.39885671 81.65615723 67.28919519
 85.46197499 43.59797963 89.36293819 88.41926646 21.65744025 7.89948895
 18.56581382 72.18161283 58.63869599 82.8823785 52.72104226 10.45288446
 55.77659647 80.78470279 68.09838774 70.81928573 58.15523257 54.34941481
 51.3847606 92.78817418 37.23274943 88.68177464 53.87368759 76.32881237
 78.78604847 90.98841874 61.79938387 62.4367775 88.22115287 89.04833753]]

0s completed at 11:25
```

College_students_mark_predictor (1).ipynb - Colaboratory

https://colab.research.google.com/drive/1dD_nT0KivQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=eBfeleEiaOB

College_students_mark_predictor (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM 100% Disk 100%

Editing

```
[39] #accuracy
from sklearn.metrics import r2_score
ensemble_prediction=(pred1*0.4+pred2*0.4
                    +pred3*0.1+pred4*0.1)
r2_score_ensemble=r2_score(y_test,ensemble_prediction)
print("Ensemble Model Accuracy")
print(r2_score_ensemble)

Ensemble Model Accuracy
0.895466830675874

[40] from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn.metrics import explained_variance_score
r2_score1=r2_score(y_test,pred1)
variance_score1=explained_variance_score(y_test,pred1)
mean_absolute_error1=mean_absolute_error(y_test,pred1)
mean_squared_log_error1=mean_squared_log_error(y_test,pred1)

r2_score2=r2_score(y_test,pred2)
variance_score2=explained_variance_score(y_test,pred2)
mean_absolute_error2=mean_absolute_error(y_test,pred2)
mean_squared_log_error2=mean_squared_log_error(y_test,pred2)

r2_score3=r2_score(y_test,pred3)
variance_score3=explained_variance_score(y_test,pred3)
mean_absolute_error3=mean_absolute_error(y_test,pred3)
mean_squared_log_error3=mean_squared_log_error(y_test,pred3)

r2_score4=r2_score(y_test,pred4)
variance_score4=explained_variance_score(y_test,pred4)
mean_absolute_error4=mean_absolute_error(y_test,pred4)
mean_squared_log_error4=mean_squared_log_error(y_test,pred4)

[41] #printing the values
print("Random Forest Regressor Report")
print("-> R2 Score:",r2_score1)
print("->mean absolute error:",mean_absolute_error1)
print("->variance score:",variance_score1)
print("-> mean squared log error:",mean_squared_log_error1)

#gradient Boosting
```

0s completed at 11:25

College_students_mark_predictor (1).ipynb - Colaboratory

https://colab.research.google.com/drive/1dD_nT0KivQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=eBfeleEiaOB

College_students_mark_predictor (1).ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM 100% Disk 100%

Editing

```
mean_squared_log_error4=mean_squared_log_error(y_test,pred4)

[40] mean_squared_log_error4=mean_squared_log_error(y_test,pred4)

[41] #printing the values
print("Random Forest Regressor Report")
print("-> R2 Score:",r2_score1)
print("->mean absolute error:",mean_absolute_error1)
print("->variance score:",variance_score1)
print("-> mean squared log error:",mean_squared_log_error1)

#gradient Boosting

print("\n")
print("Gradient Boosting Report")
print("-> R2 Score:",r2_score2)
print("->mean absolute error:",mean_absolute_error2)
print("->variance score:",variance_score2)
print("-> mean squared log error:",mean_squared_log_error2)

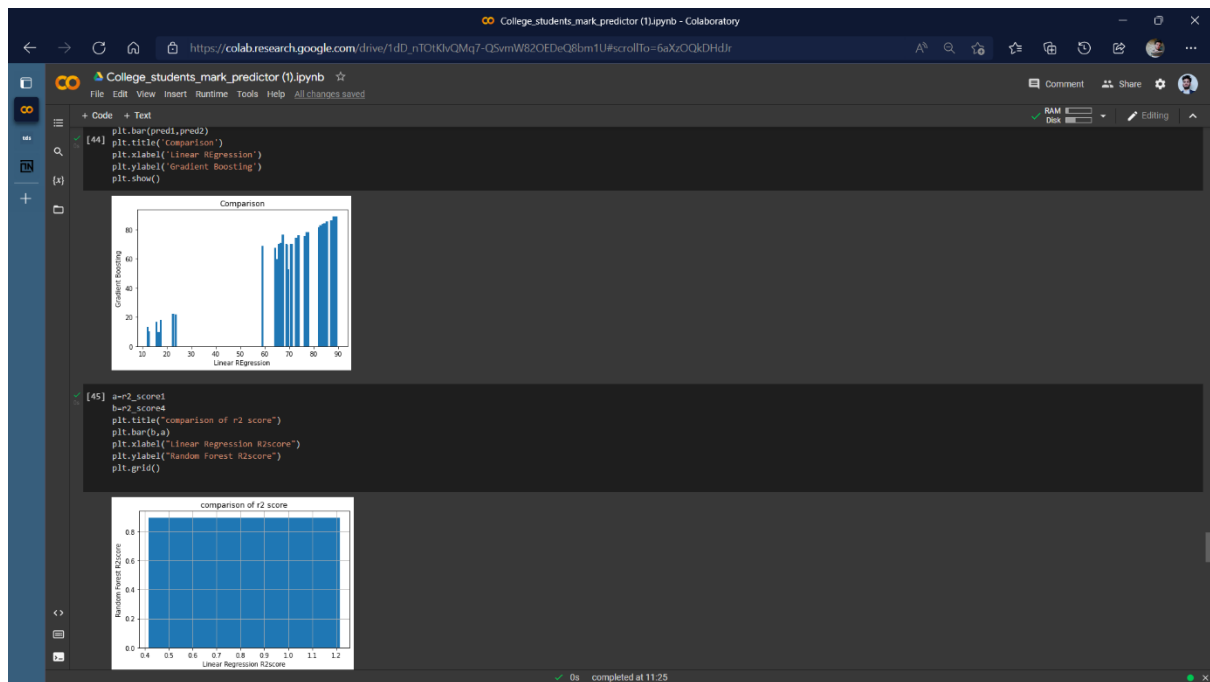
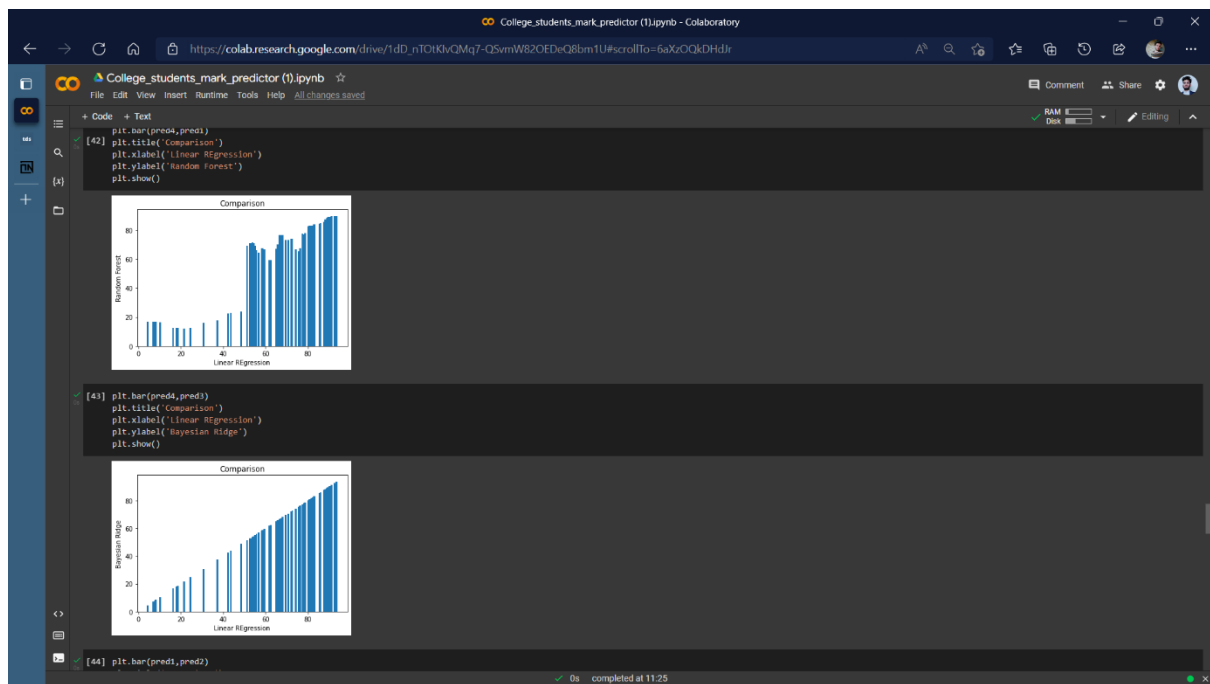
#Bayesian Ridge
print("\n")
print("Bayesian Ridge Report")
print("-> R2 Score:",r2_score3)
print("->mean absolute error:",mean_absolute_error3)
print("->variance score:",variance_score3)
print("-> mean squared log error:",mean_squared_log_error3)

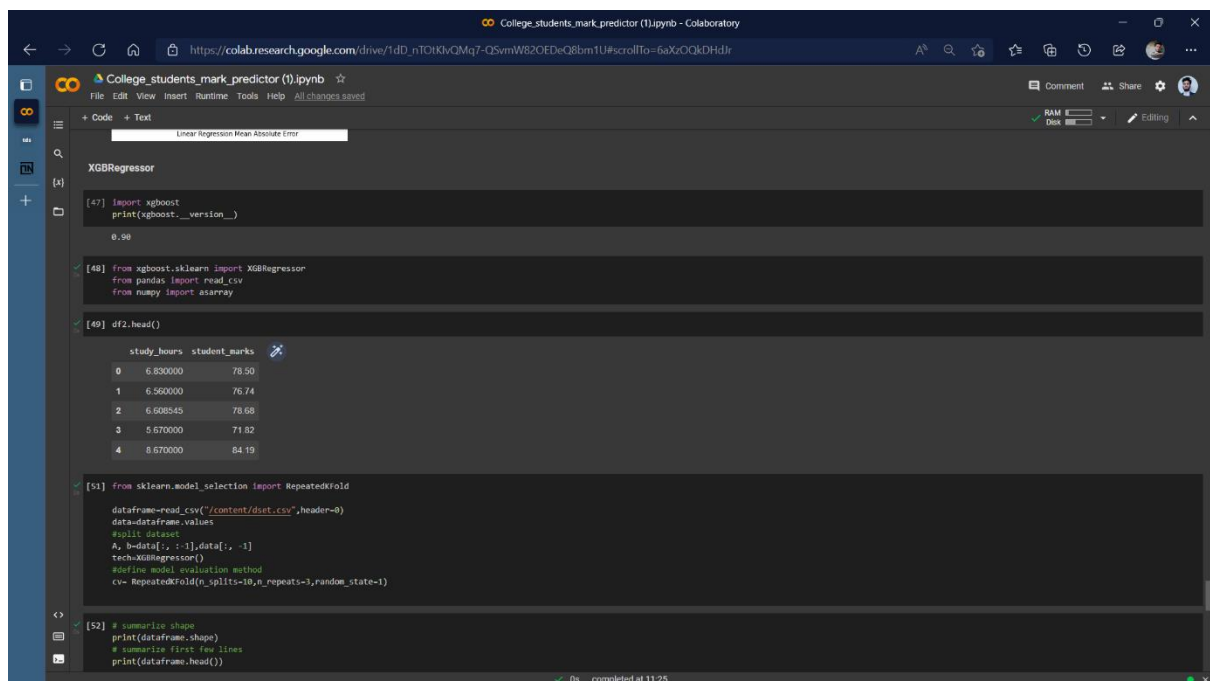
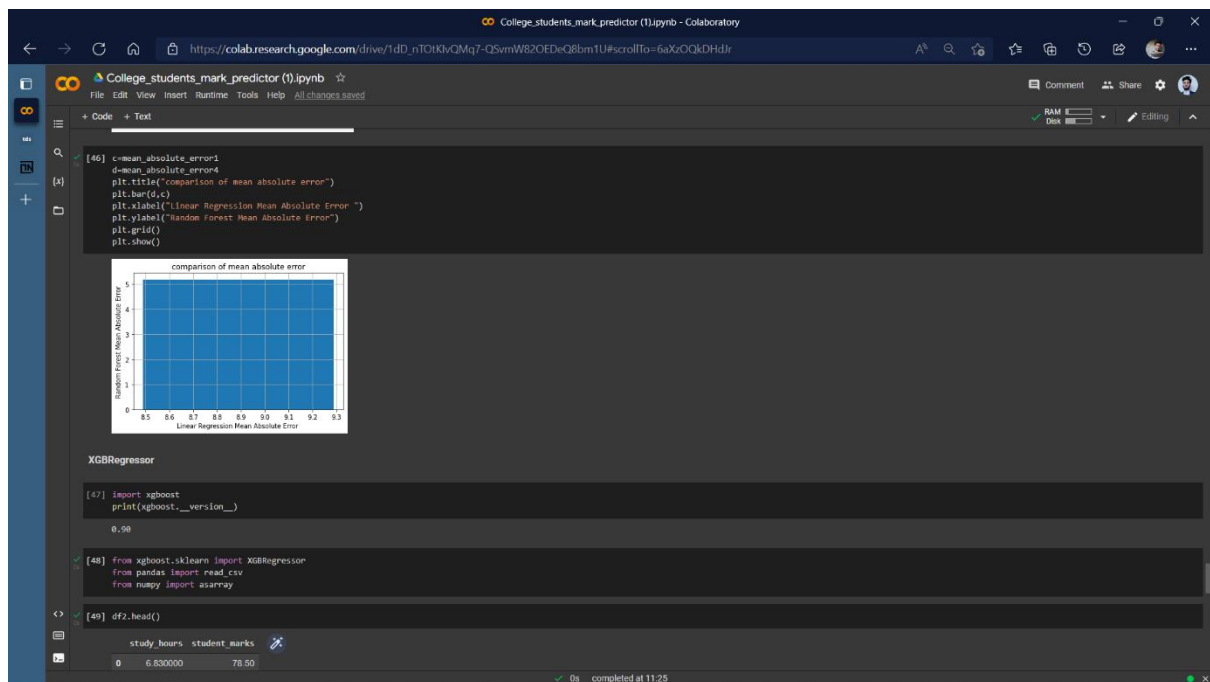
#linear regressor
print("\n")
print("Linear Regression Report")
print("-> R2 Score:",r2_score4)
print("->mean absolute error:",mean_absolute_error4)
print("->variance score:",variance_score4)
print("-> mean squared log error:",mean_squared_log_error4)

Random Forest Regressor Report
-> R2 Score: 0.896963683812532
->mean absolute error: 5.481521846864641
->variance score: 0.8978639329275524
-> mean squared log error: 0.06655997246944537

Gradient Boosting Report
-> R2 Score: 0.885145678052334
->mean absolute error: 4.8327961791467545
->variance score: 0.8852157380628376
```

0s completed at 11:25






```
College_students_mark_predictor (1).ipynb - Colaboratory
https://colab.research.google.com/drive/1dD_nT0iKvQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=6aXzOOkDHdJr

College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[52] (387, 2)
study_hours student_marks
0 6.83 78.58
1 6.56 76.74
2 NaN 78.48
3 5.67 71.82
4 8.67 84.19

[53] #fitting model
tech.fit(A, b)

[04:01:06] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
XGBRegressor()

[54] from sklearn.model_selection import train_test_split

A_train, A_test, b_train, b_test = train_test_split(A, b)
tech = XGBRegressor()
tech.fit(A_train, b_train)
pred5 = tech.predict(A_test)
print(pred5)

[04:01:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
[12.418741 82.826706 76.71983 84.639385 89.02195 89.02195 72.02018
88.02489 72.0154 9.958889 89.02195 71.61751 88.92489 83.25523
70.82389 83.25523 8.805338 72.0154 89.02195 34.48811 84.10802
84.65437 82.30389 84.504684 70.276276 70.391235 88.81256 74.63107
78.67732 12.418741 89.02195 18.320934 84.17231 68.178684 89.02195
71.15669 88.81256 21.858252 89.02195 88.902076 74.63107 83.25523
88.81256 85.91288 89.02195 22.483868 71.72464 88.96712 73.61937
70.82389 71.15669 74.437096 89.02195 84.504684 71.72464 17.814373
88.02489 72.0154 71.61751 89.02195 68.178684 71.15669 74.63107
70.82389 8.805338 10.873894 89.02195 9.491825 21.858252 45.10267
70.34899 68.49199 85.904205 82.84858 68.178684 89.02195 68.49199
89.02195 83.25523 78.022015 70.35846 83.25523 51.807537 8.805338
88.96712 57.128674 18.320934 85.904205 74.437096 71.61751 9.95746
70.35848 74.437096 84.504684 70.82389 74.437096 9.491825]

[55] # define new data
row = [1]
new_data = ndarray([row])
# make a prediction
yhat = tech.predict(new_data)
# summarize prediction
Predicted: 81.811

0s completed at 11:25
```

```
College_students_mark_predictor (1).ipynb - Colaboratory
https://colab.research.google.com/drive/1dD_nT0iKvQMq7-Q5vmW82OEDeQ8bm1U#scrollTo=6aXzOOkDHdJr

College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[55] # make a prediction
yhat = tech.predict(new_data)
# summarize prediction
print('Predicted: %.3f' % yhat)

Predicted: 81.811

[56] r2_score=r2_score(b_test,pred5)
variance_score=explained_variance_score(b_test,pred5)
mean_absolute_error=mean_absolute_error(b_test,pred5)
mean_squared_log_error=mean_squared_log_error(b_test,pred5)

[57] #printing the values
print("EXTREME GRADIENT BOOSTING Regressor Report")
print("-> R2 Score: ",r2_score)
print("->mean absolute error: ",mean_absolute_error)
print("->variance_score: ",variance_score)
print("-> mean squared log error: ",mean_squared_log_error)

EXTREME GRADIENT BOOSTING Regressor Report
-> R2 Score: 0.7678594822228483
->mean absolute error: 5.75834941525883
->variance_score: 0.7703124288956388
-> mean squared log error: 0.08556883789054551

0s completed at 11:25
```

Conclusion

I have been asked to work on a particular model designed generally to predict the marks of college students. The title of the project as mentioned above college students mark prediction is an artificial intelligence model.

We have considered a real-world data set which comprises of certain values that are fixed. We are using supervised learning and in particular linear regression to predict the output from pre saved input and data. There are various parameters there are included to get an accurate output. Beside loading data set we have discovered and visualised the data to gain insights which includes the brief detail and information regarding data set. The inclusion of scatter function made it easy to differentiate between student study hours and student marks. As in dataset it is given in form of prior information that includes student study hours respective to that of student marks. In order to prepare the data for machine learning algorithm we perform data cleaning that is to check is there any null value present and furthermore to calculate the mean. Post that we have done the data splitting just for the purpose of training, testing and to predict the model. The concept of linear regression is used here to predict the output. In order to fine tune the model, the calculation of score is very necessary and we have found that it is around 0.95 that is approximately 95%. At last, to save our model a special library is introduced named joblib that generally serves the purpose to save and load the model for prediction.

There are various other machine learning algorithms that are used to fine tune the data and to predict the required output. The plus point of these algorithms is that these give us desired output in minimum time and space complexity.

Future Scope

Future on this dataset is as vast as the limits of human mind. We can always keep learning and teaching the computers how to learn. And at the same time, wondering how some of the most complex machine learning algorithms have been running in the back of our own mind effortlessly all the time. So, in future may be many more algorithms could be used to train the model and that could give the even higher accuracy. There is intense research in machine learning at the top universities in the world. The global machine learning as a service market is rising expeditiously mainly due to the Internet revolution. The process of connecting the world virtually has generated vast amount of data which is boosting the adoption of machine learning solutions. Considering all these applications and dramatic improvements that ML. has brought us, it doesn't take a genius to realize that in coming future we will see more advanced applications of ML, applications that will stretch the capabilities of machine learning to an unimaginable level. So let us just hope that someone could work on this and make our model to give higher accuracy.

Reference

[How to find best hyperparameters using GridSearchCV in python - Thinking Neuron](#)

[College_students_mark_predictor \(1\).ipynb - Colaboratory \(google.com\)](#)

[venom005/College-Student-Marks-Prediction \(github.com\)](#)

[ML | Types of Learning – Supervised Learning - GeeksforGeeks](#)

[HTML Basic \(w3schools.com\)](#)

[Streamlit • The fastest way to build and share data apps](#)

[DataTechNotes: Regression Example with XGBRegressor in Python](#)