COLLEGE STUDENT MARKS PREDICTION

PANKAJ RAINA (11910326)

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab, India

pankajraina05@gmail.com

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 descripting 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.

I. 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 quiet 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 a 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 m 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

II. 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.

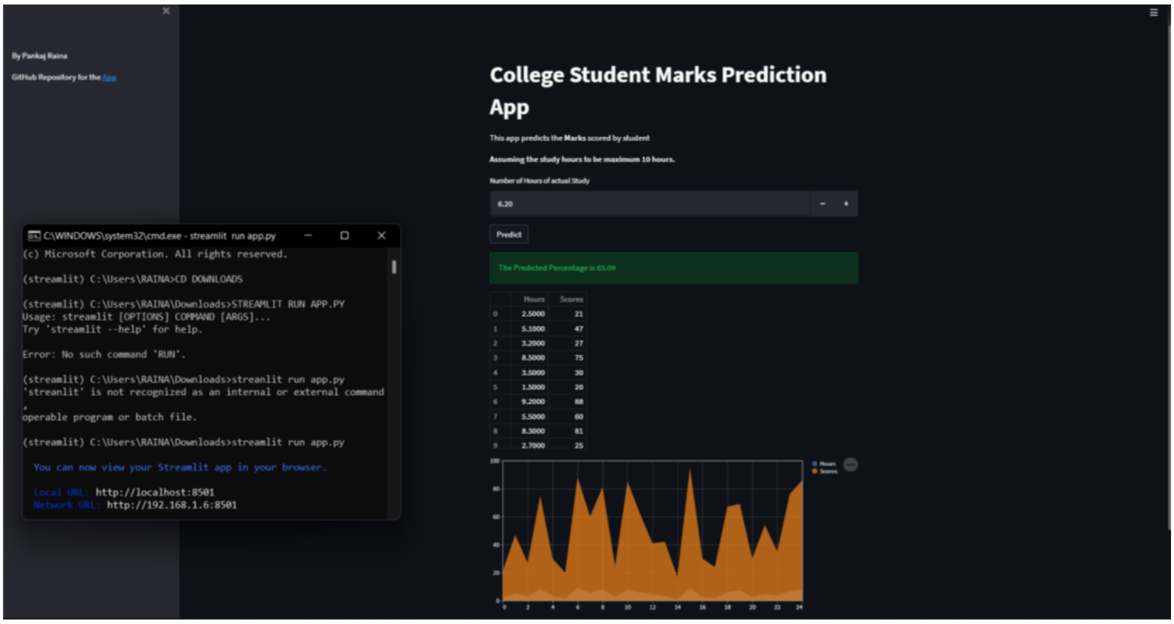
2.Platform Independent

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.

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.



By Pankaj Raina

GitHub Repository for the App

```
C:\WINDOWS\system32\cmd.exe - streamlit  run app.py
(c) Microsoft Corporation. All rights reserved.

(streamlit) C:\Users\RAINA>CD DOWNLOADS

(streamlit) C:\Users\RAINA\Downloads>STREAMLIT RUN APP.PY
Usage: streamlit [OPTIONS] COMMAND [ARGS]...
Try 'streamlit --help' for help.

Error: No such command 'RUN'.

(streamlit) C:\Users\RAINA\Downloads>streanlit run app.py
'streanlit' is not recognized as an internal or external command
,
operable program or batch file.

(streamlit) C:\Users\RAINA\Downloads>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.6:8501
```

College Student Marks Prediction App

This app predicts the **Marks** scored by student

Assuming the study hours to be maximum 10 hours.

Number of Hours of actual Study

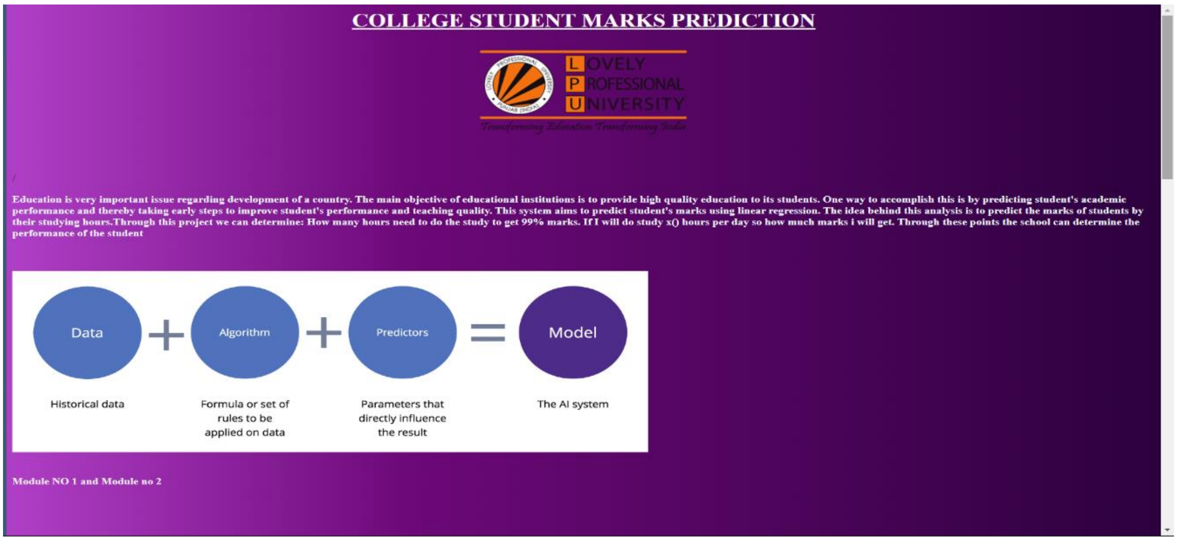
6.20

The Predicted Percentage is 63.09

	Hours	Scores
0	2.5000	21
1	5.1000	47
2	3.2000	27
3	8.5000	75
4	3.5000	30
5	1.5000	20
6	9.2000	88
7	5.5000	60
8	8.3000	81
9	2.7000	25

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.

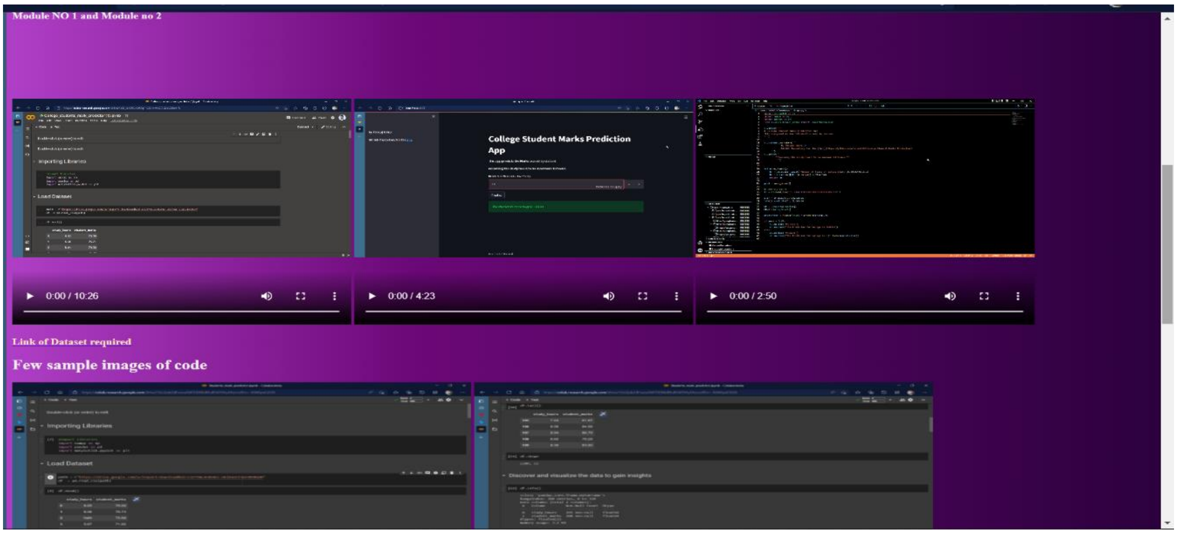


COLLEGE STUDENT MARKS PREDICTION

Education is very important issue regarding development of a country. The main objective of educational institutions is to provide high quality education to its students. One way to accomplish this is by predicting student's academic performance and thereby taking early steps to improve student's performance and teaching quality. This system aims to predict student's marks using linear regression. The idea behind this analysis is to predict the marks of students by their studying hours.Through this project we can determine: How many hours need to do the study to get 99% marks. If I will do study x() hours per day so how much marks i will get. Through these points the school can determine the performance of the student

Module NO 1 and Module no 2

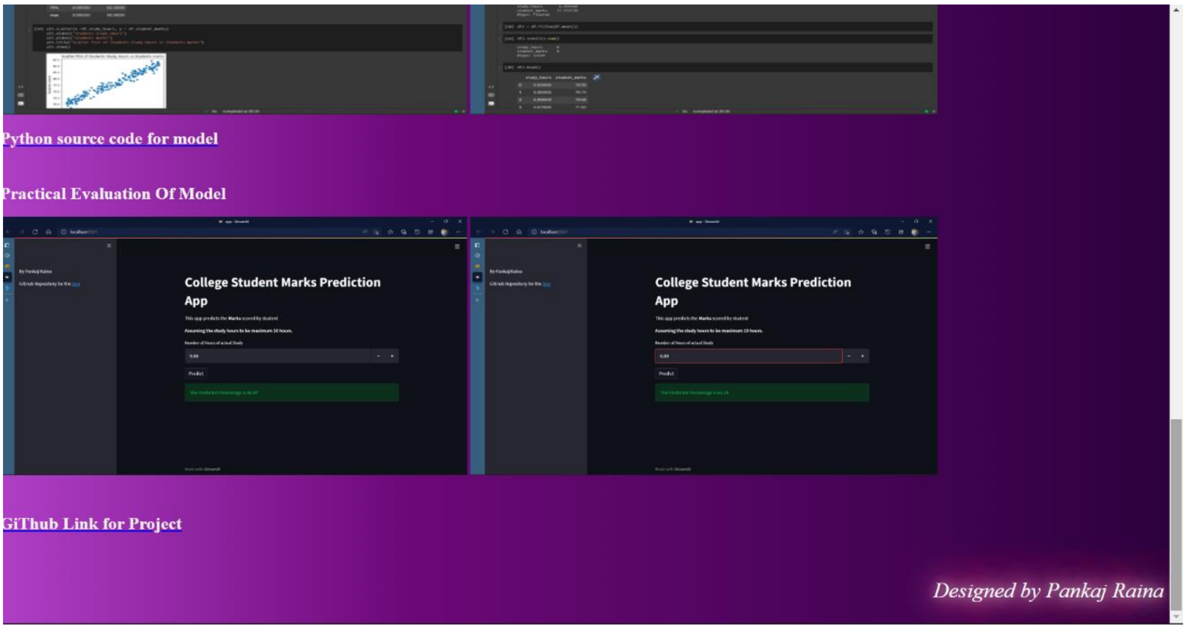
LOVELY
PROFESSIONAL
UNIVERSITY
Transforming Education Transforming India
Data
+
Algorithm
+
Predictors
=
Model
Historical data
Formula or set of rules to be applied on data
Parameters that directly influence the result
The AI system



Module NO 1 and Module no 2

Link of Dataset required

Few sample images of code



Python source code for model

Practical Evaluation Of Model

College Student Marks Prediction App
College Student Marks Prediction App

GiThub Link for Project

Designed by Pankaj Raina

III. LITERATURE REVIEW

A core objective of a leamer 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 R2 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 R2 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 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. 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 r2 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.

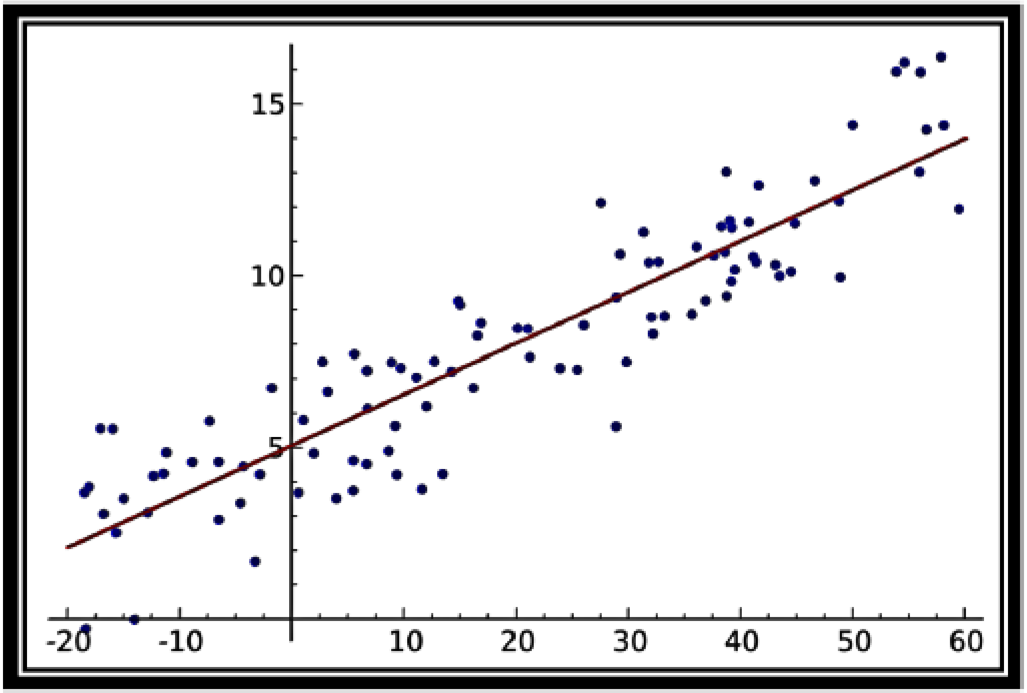
IV. Machine Learning Algorithms

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 M Followings are the Algorithms of Python Machine Learning -

1. Linear Regression

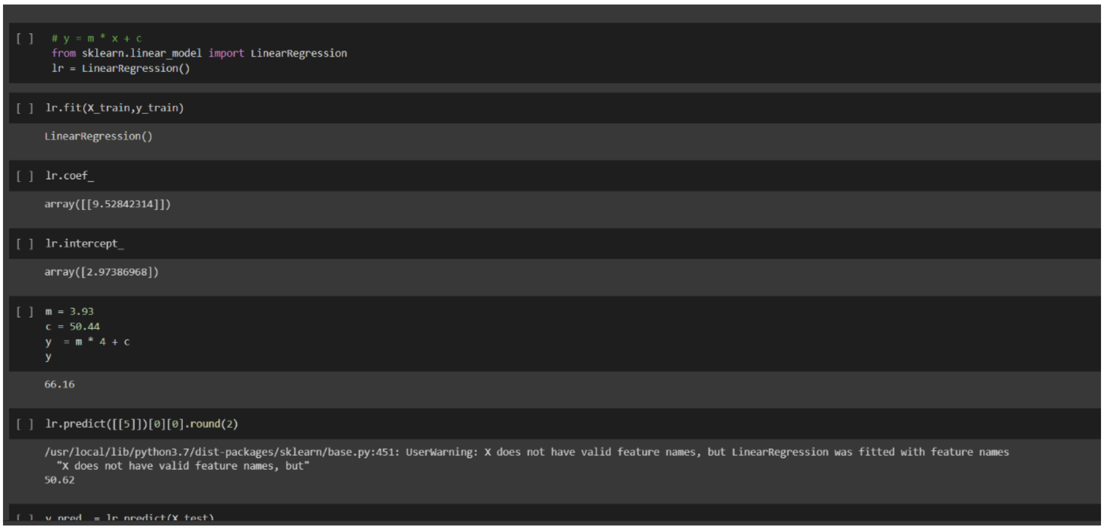
Linear regression is one of the supervised Machine learning algorithms in Python that observes continuous features and predicts an outcome. Depending on whether it runs on a single variable or on many features, we can call it simple linear regression or multiple linear regression. This is one of the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

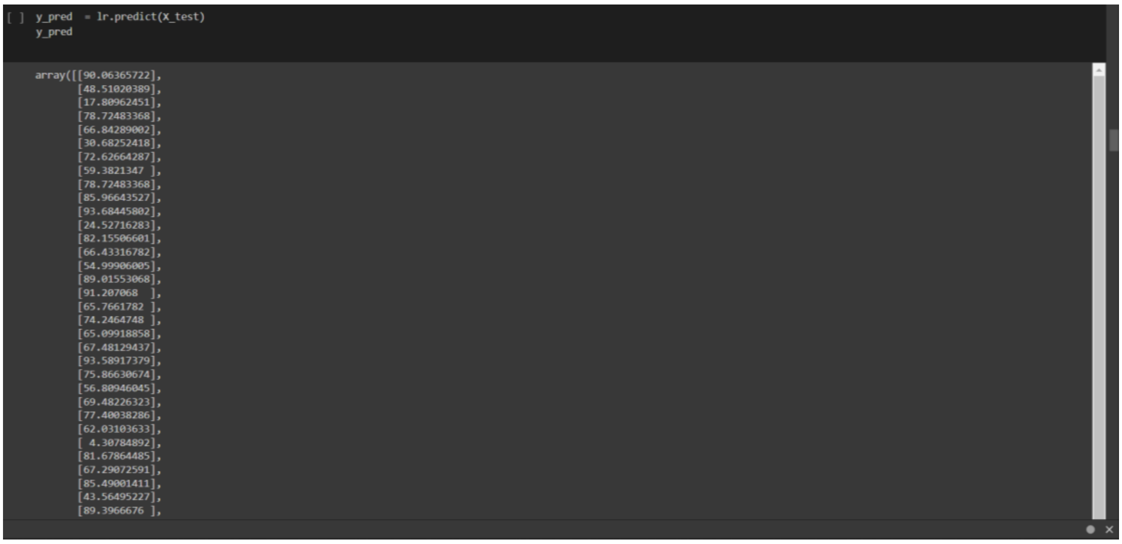


Practical use of Linear Regression

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
```

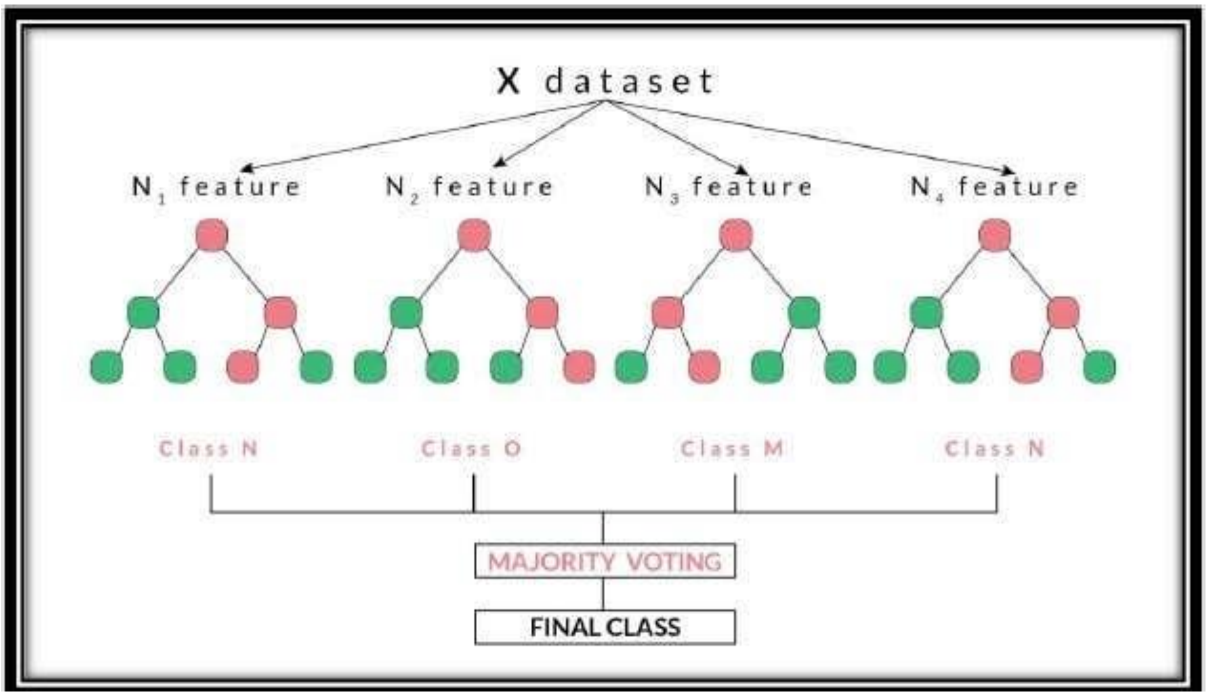
```
array([[90.06365722],
       [48.51020389],
       [17.80962451],
       [78.72483368],
       [66.84289002],
       [30.68252418],
       [72.62664287],
       [59.3821347 ],
       [78.72483368],
       [85.96643527],
       [93.68445802],
       [24.52716283],
       [82.15506601],
       [66.43316782],
       [54.99906005],
       [89.01553068],
       [91.207068  ],
       [65.7661782 ],
       [74.2464748 ],
       [65.09918858],
       [67.48129437],
       [93.58917379],
       [75.86630674],
       [56.80946045],
       [69.48226323],
       [77.40038286],
       [62.03103633],
       [ 4.30784892],
       [81.67864485],
       [67.29072591],
       [85.49001411],
       [43.56495227],
       [89.3966676 ],
```

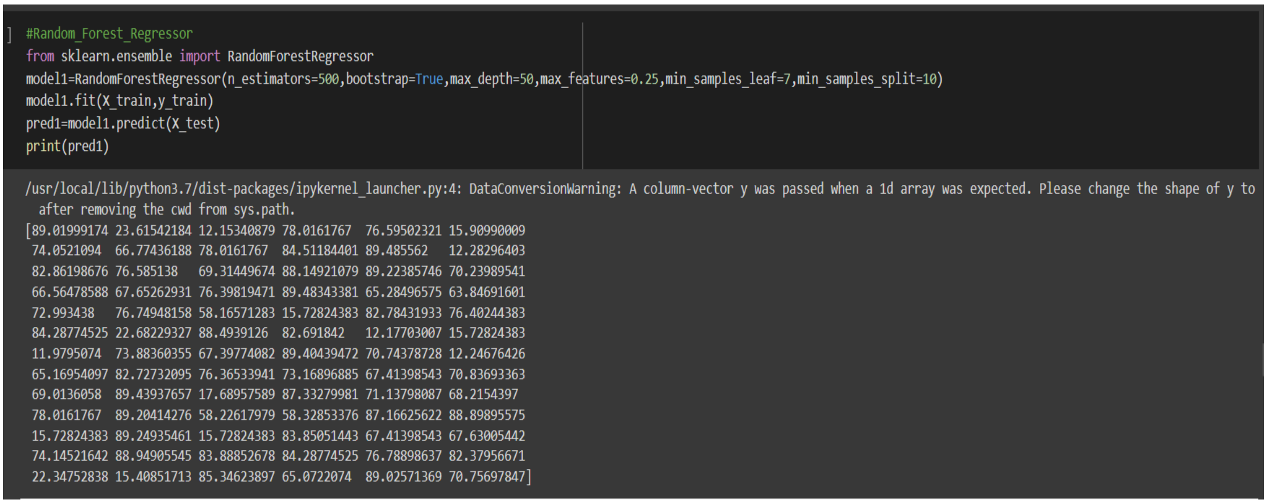
```
lr.score(X_test,y_test)
```

```
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.

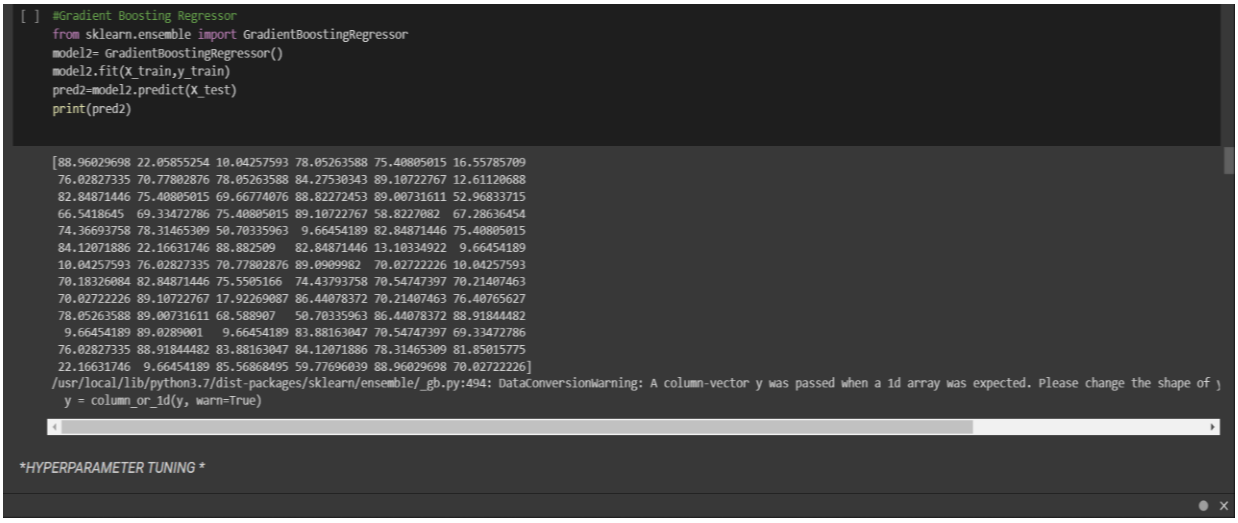




```
]  #Random_Forest_Regressor
   from sklearn.ensemble import RandomForestRegressor
   model1=RandomForestRegressor(n_estimators=500,bootstrap=True,max_depth=50,max_features=0.25,min_samples_leaf=7,min_samples_split=10)
   model1.fit(X_train,y_train)
   pred1=model1.predict(X_test)
   print(pred1)

   /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 58.22617979 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.



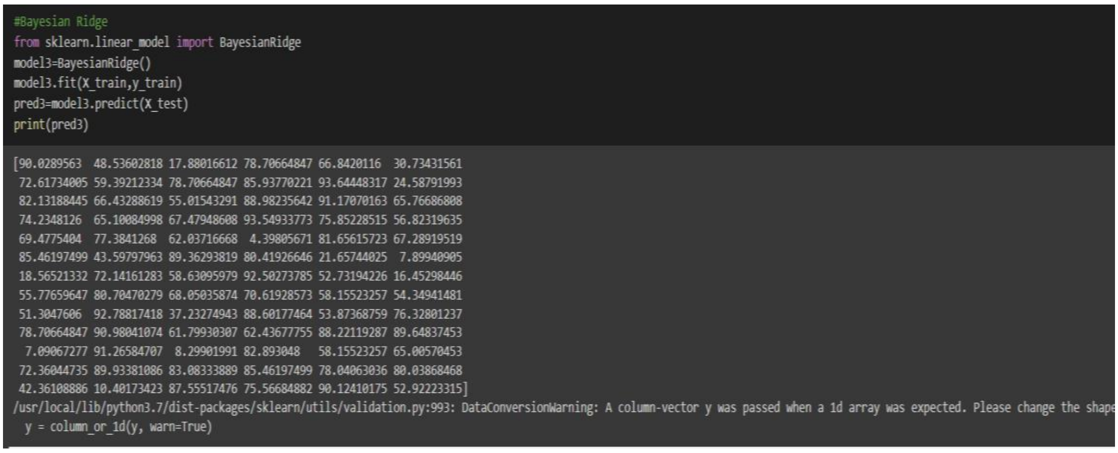
```
[ ]  #Gradient Boosting Regressor
     from sklearn.ensemble import GradientBoostingRegressor
     model2= GradientBoostingRegressor()
     model2.fit(X_train,y_train)
     pred2=model2.predict(X_test)
     print(pred2)

     [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□λ)=N(w□0,λ−1Ip)p(w□λ)=N(w□0,λ−1Ip)
This resulting model is called Bayesian Ridge Regression and in scikit-learn **sklearn.linear_model.BeyesianRidge** 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)
```

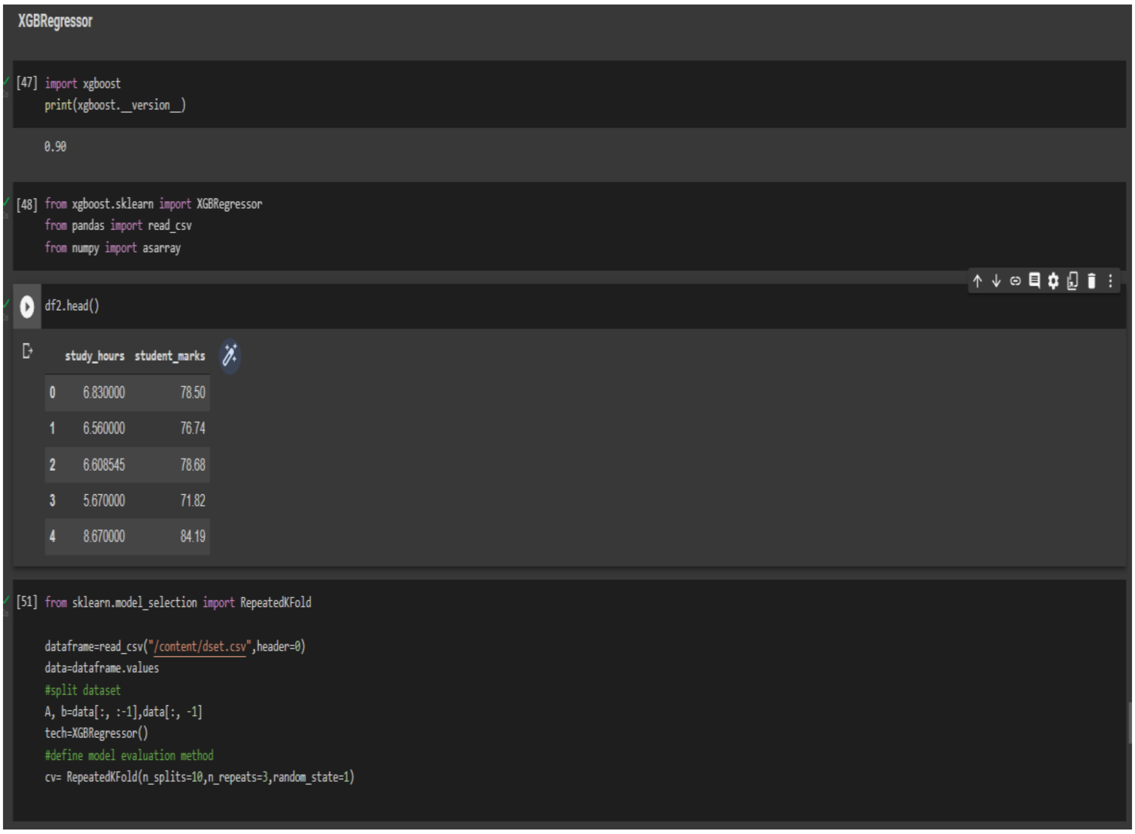
```
[90.0289563  48.53602818 17.88016612 78.70664847 66.8420116  30.73431561
 72.61734005 59.39212334 78.70664847 85.93770221 93.64448317 24.58791993
 82.13188445 66.43288619 55.01543291 88.98235642 91.17070163 65.76686808
 74.2348126  65.10084998 67.47948608 93.54933773 75.85228515 56.82319635
 69.4775404  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 58.63095979 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__)
```

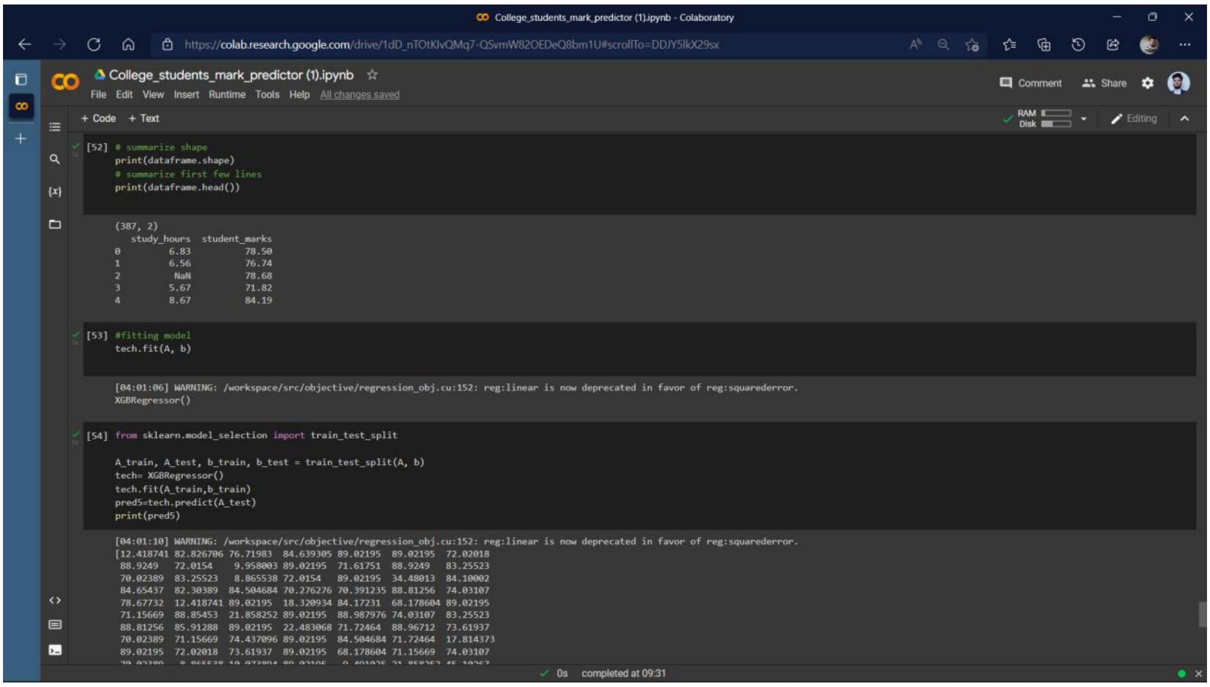
```
0.90
```

```
[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 - Colaboratory
https://colab.research.google.com/drive/1dD_nTOtKlvQMq7-QSvmW82OEDeQ8bm1U#scrollTo=DDJY5lkX29sx
College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing

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

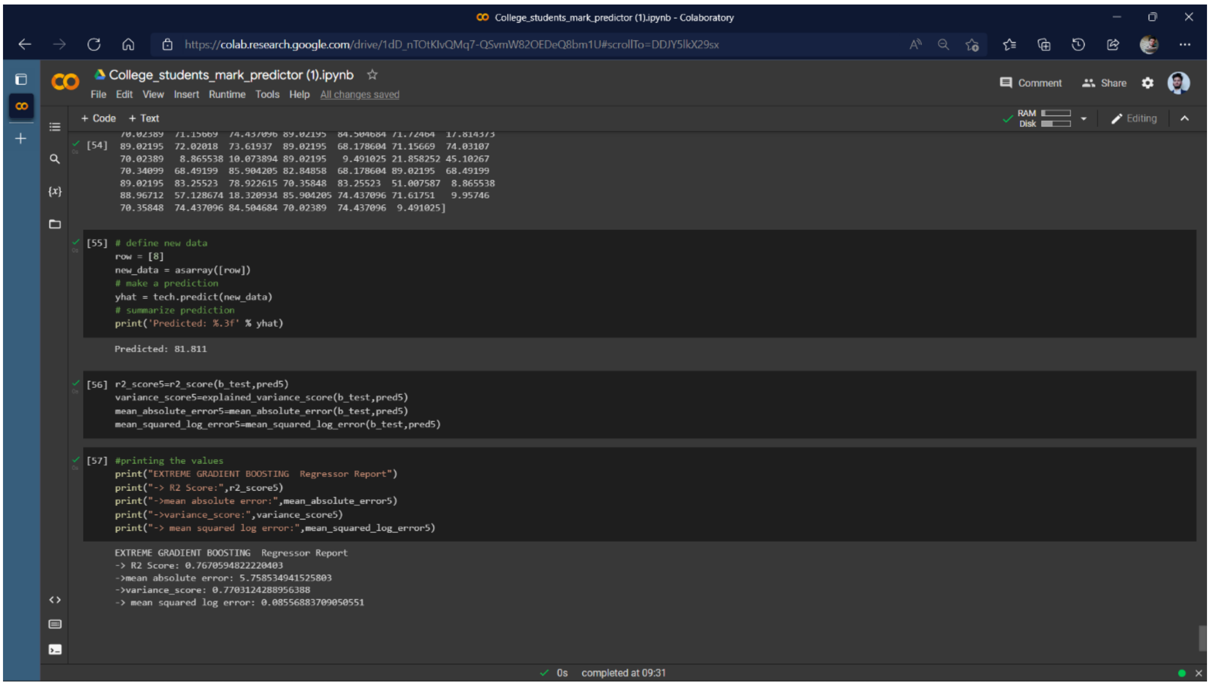
```
(387, 2)
   study_hours  student_marks
0         6.83          78.50
1         6.56          76.74
2          NaN          78.68
3         5.67          71.82
4         8.67          84.19
```

```
[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)
```

```
[04:01:10] 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   8.865538 72.0154   89.02195  34.48013  84.10002
 84.65437  82.30389  84.504684 70.276276 70.391235 88.81256  74.03107
 78.67732  12.418741 89.02195  18.320934 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  22.483068 71.72464  88.96712  73.61937
 70.02389  71.15669  74.437096 89.02195  84.504684 71.72464  17.814373
 89.02195  72.02018  73.61937  89.02195  68.178604 71.15669  74.03107
```



College_students_mark_predictor (1).ipynb - Colaboratory
https://colab.research.google.com/drive/1dD_nTOtKlvQMq7-QSvmW82OEDeQ8bm1U#scrollTo=DDJY5lkX29sx
College_students_mark_predictor (1).ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
Editing
0s completed at 09:31

```
      70.02389  71.15669  74.437096 89.02195  84.504684 71.72464  17.814373
[54]  89.02195  72.02018  73.61937  89.02195  68.178604 71.15669  74.03107
      70.02389   8.865538 10.073894 89.02195   9.491025 21.858252 45.10267
      70.34099  68.49199  85.904205 82.84858  68.178604 89.02195  68.49199
      89.02195  83.25523  78.922615 70.35848  83.25523  51.007587  8.865538
      88.96712  57.128674 18.320934 85.904205 74.437096 71.61751   9.95746
      70.35848  74.437096 84.504684 70.02389  74.437096  9.491025]
```

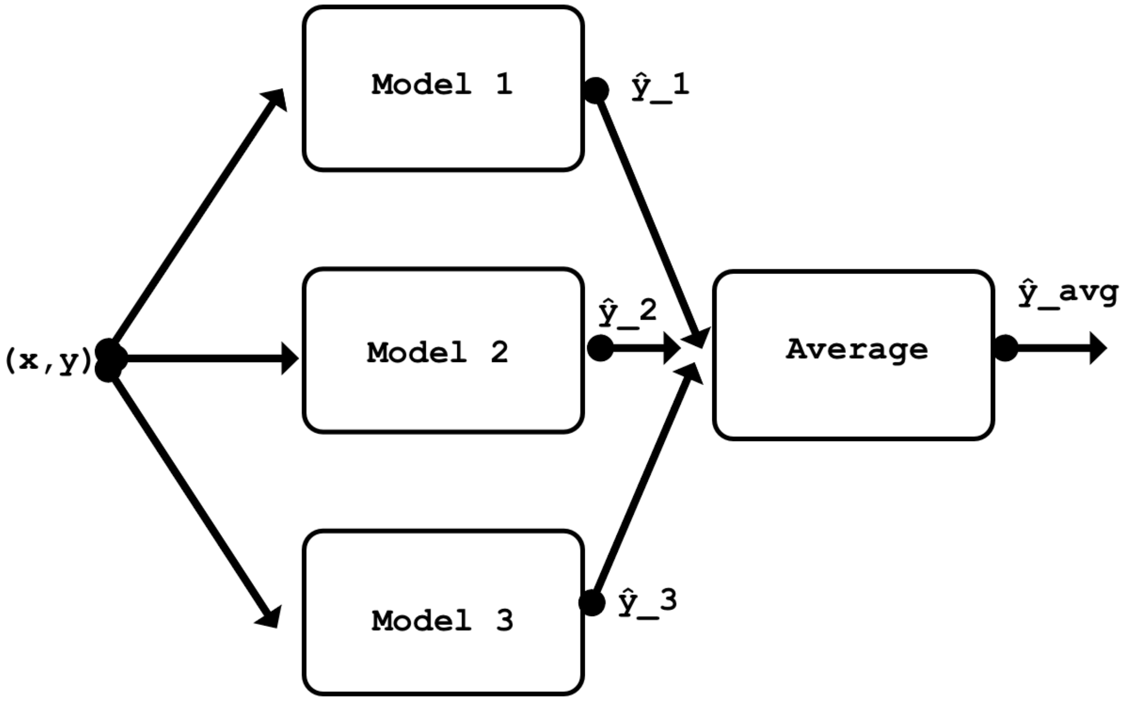
```
[55] # define new data
     row = [8]
     new_data = asarray([row])
     # make a prediction
     yhat = tech.predict(new_data)
     # summarize prediction
     print('Predicted: %.3f' % yhat)

     Predicted: 81.811
```

```
[56] r2_score5=r2_score(b_test,pred5)
     variance_score5=explained_variance_score(b_test,pred5)
     mean_absolute_error5=mean_absolute_error(b_test,pred5)
     mean_squared_log_error5=mean_squared_log_error(b_test,pred5)
```

```
[57] #printing the values
     print("EXTREME GRADIENT BOOSTING  Regressor Report")
     print("-> R2 Score:",r2_score5)
     print("->mean absolute error:",mean_absolute_error5)
     print("->variance_score:",variance_score5)
     print("-> mean squared log error:",mean_squared_log_error5)

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

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***.



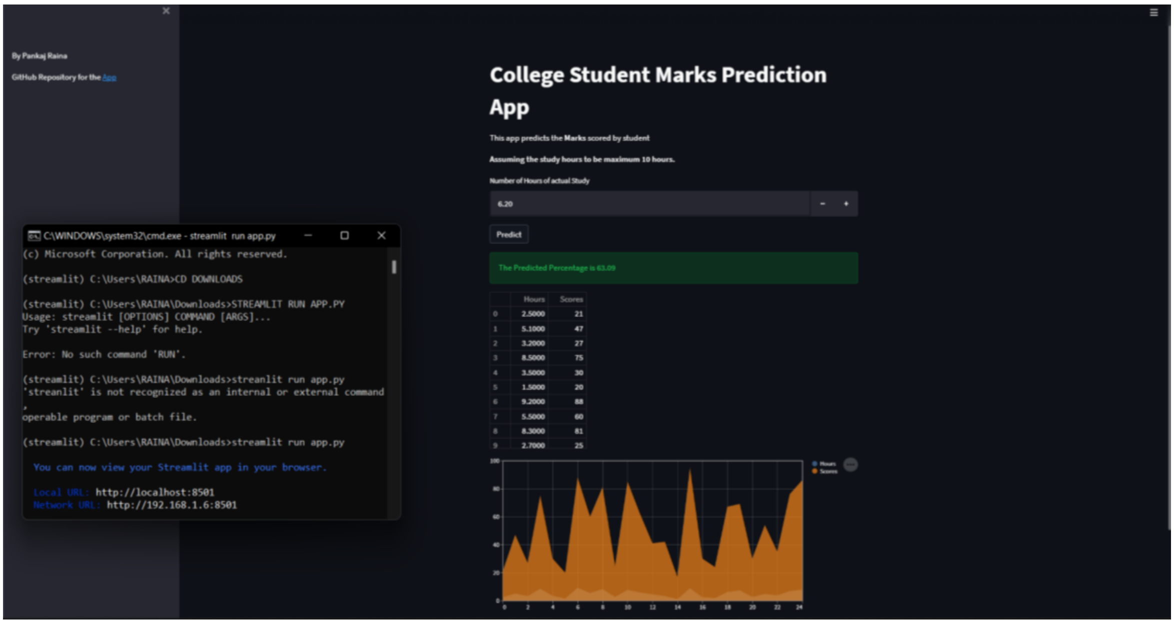
```
#accuracy
from sklearn.metrics import r2_score
ensemble_prediction=(pred1*0.4+pred2*0.4
                    +pred3*0.1+pred4*0.1)
r2_score_ensemble=r2_score(y_test,ensemble_prediction)
print("Ensemble MoDEL Accuracy")
print(r2_score_ensemble)
```

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)
```



By Pankaj Raina

GitHub Repository for the App

```
C:\WINDOWS\system32\cmd.exe - streamlit  run app.py
(c) Microsoft Corporation. All rights reserved.

(streamlit) C:\Users\RAINA>CD DOWNLOADS

(streamlit) C:\Users\RAINA\Downloads>STREAMLIT RUN APP.PY
Usage: streamlit [OPTIONS] COMMAND [ARGS]...
Try 'streamlit --help' for help.

Error: No such command 'RUN'.

(streamlit) C:\Users\RAINA\Downloads>streanlit run app.py
'streanlit' is not recognized as an internal or external command
,
operable program or batch file.

(streamlit) C:\Users\RAINA\Downloads>streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.6:8501
```

College Student Marks Prediction App

This app predicts the **Marks** scored by student

Assuming the study hours to be maximum 10 hours.

Number of Hours of actual Study

6.20

The Predicted Percentage is 63.09

	Hours	Scores
0	2.5000	21
1	5.1000	47
2	3.2000	27
3	8.5000	75
4	3.5000	30
5	1.5000	20
6	9.2000	88
7	5.5000	60
8	8.3000	81
9	2.7000	25

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.

V. 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 hyper parameter 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.

VI. 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.

VII. 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.

VIII. 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