7/24/2022

REPORT

BY AHMER WAQAR

PROBLEM DESCRIPTION:

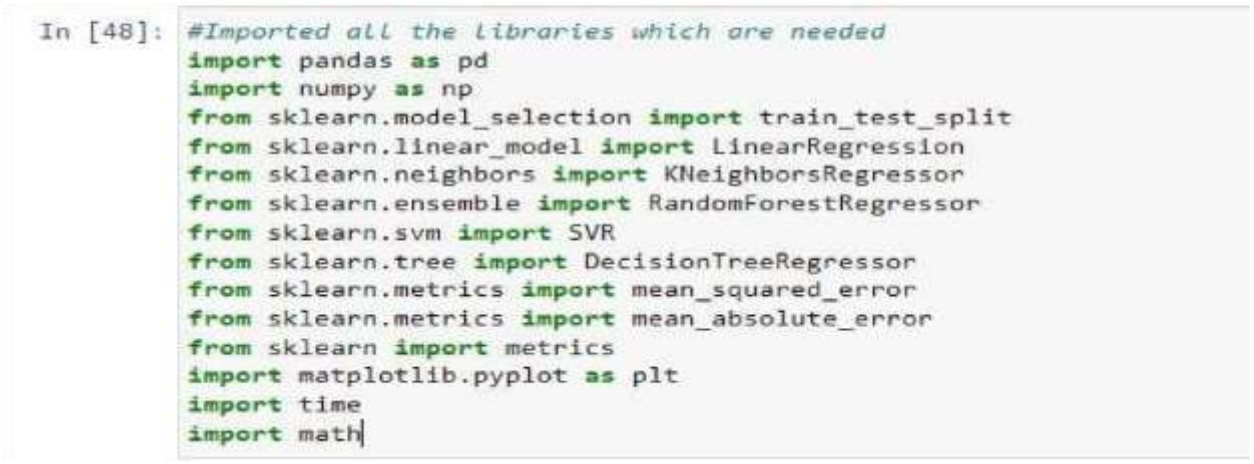
Prediction of cummulative grade point average through ML algorithms.

DATA PREPROCESSING :

Data preprocessing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In simple words, data preprocessing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

IMPORTING LIBRARIES :

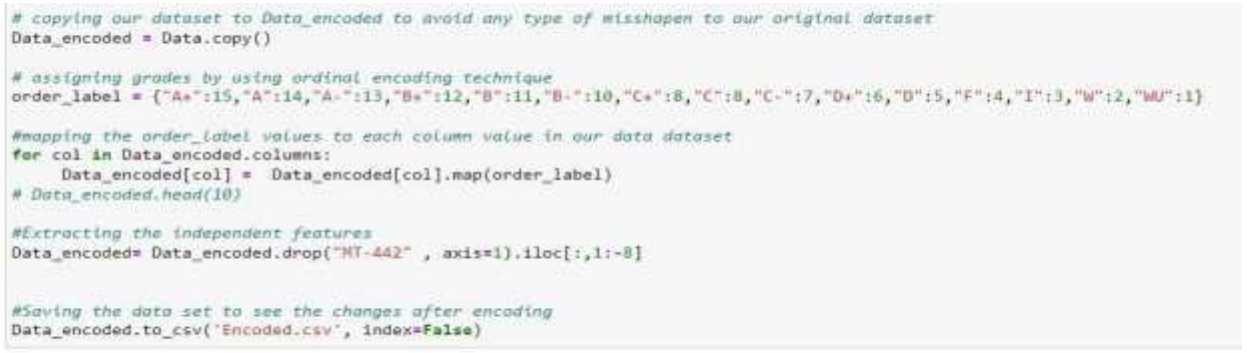
Specific data preprocessing tasks can be carried out using the specified Python libraries. One of the main step in machine learning's data preprocessing is importing all essential libraries. Some of the libraries used in this project are Numpy, Pandas, time , math and Sklearn. From the code



```
In [48]: #Imported all the libraries which are needed
         import pandas as pd
         import numpy as np
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.svm import SVR
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.metrics import mean_squared_error
         from sklearn.metrics import mean_absolute_error
         from sklearn import metrics
         import matplotlib.pyplot as plt
         import time
         import math
```

ENCODING :

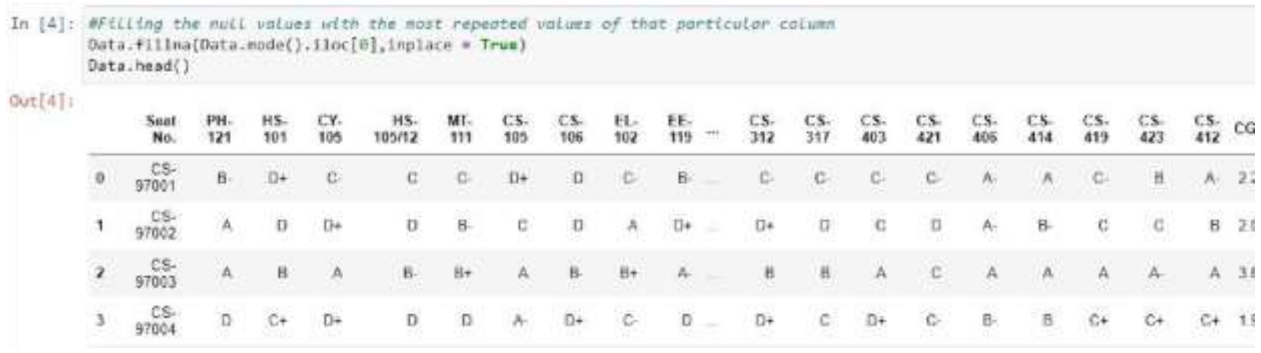
To implement some machine learning algorithms it is necessary to transform categorical data into numerical form. According to the given dataset, Ordinal encoding is implemented to convert grades into GPA. A order_label dictionary is created and the grades are then mapped to that dictionary. We have converted that encoded code into csv data file as well From the code:



```
# copying our dataset to Data_encoded to avoid any type of misshapen to our original dataset
Data_encoded = Data.copy()

# assigning grades by using ordinal encoding technique
order_label = {"A+":15,"A":14,"A-":13,"B+":12,"B":11,"B-":10,"C+":8,"C":8,"C-":7,"D+":6,"D":5,"F":4,"I":3,"W":2,"WU":1}

#mapping the order_label values to each column value in our data dataset
for col in Data_encoded.columns:
    Data_encoded[col] =  Data_encoded[col].map(order_label)
# Data_encoded.head(10)

#Extracting the independent features
Data_encoded= Data_encoded.drop("MT-442" , axis=1).iloc[:,1:-8]


#Saving the data set to see the changes after encoding
Data_encoded.to_csv('Encoded.csv', index=False)
```

FILLING ALL EMPTY SPACES:

In this technique I have filled all the empty spaces of my data set with the most repeated values. From the code:

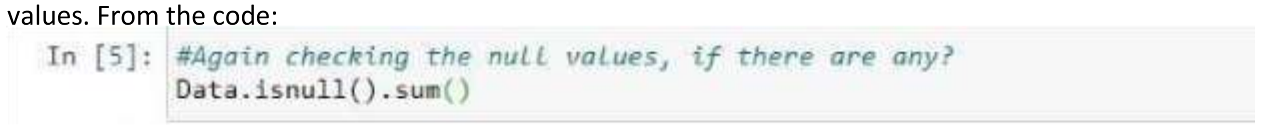


```
In [4]: #Filling the null values with the most repeated values of that particular column
        Data.fillna(Data.mode().iloc[0],inplace = True)
        Data.head()
Out[4]:
```

	Seat No.	PH-121	HS-101	CY-105	HS-105/12	MT-111	CS-105	CS-106	EL-102	EE-119	...	CS-312	CS-317	CS-403	CS-421	CS-406	CS-414	CS-419	CS-423	CS-412	CG
0	CS-97001	B-	D+	C-	C	C-	D+	D	C-	B-	...	C-	C-	C-	C-	A-	A	C-	B	A-	2.2
1	CS-97002	A	D	D+	D	B-	C	D	A	D+	...	D+	D	C	D	A-	B-	C	C	B	2.0
2	CS-97003	A	B	A	B-	B+	A	B-	B+	A-	...	B	B	A	C	A	A	A	A-	A	3.8
3	CS-97004	D	C+	D+	D	D	A-	D+	C-	D	...	D+	C	D+	C-	B-	B	C+	C+	C+	1.9

IDENTIFYING MISSING VALUES:

To identify the missing values in the dataset isna() function is used which tells us there are no missing values. From the code:

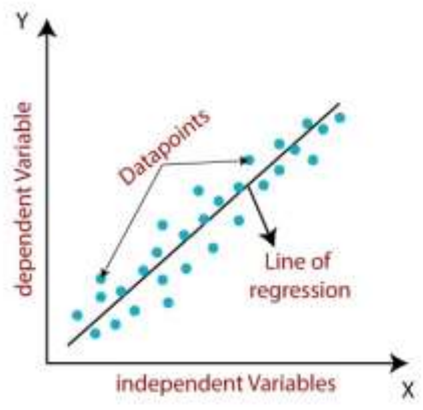


```
In [5]: #Again checking the null values, if there are any?
        Data.isnull().sum()
```

MODEL 1 (PREDICTING CGPA ON BASIS OF FIRST YEAR COURSES)

1.LINEAR REGRESSION:

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



Mathematically, we can represent a linear regression as:

- STEPS 1

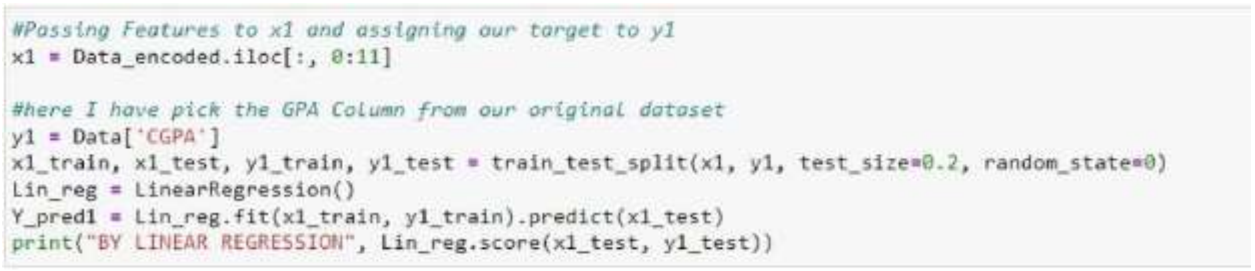
First I've picked 11 columns with complete rows and assigned it to **x1** and all the features which are assigning to **x1** are from **Data_encoded** file.

- STEPS 2

Then I assigned the Target feature **CGPA** variable to **y1** from original **DATA** file.

TRAINING A LINEAR REGRESSION FOR MODEL 1:

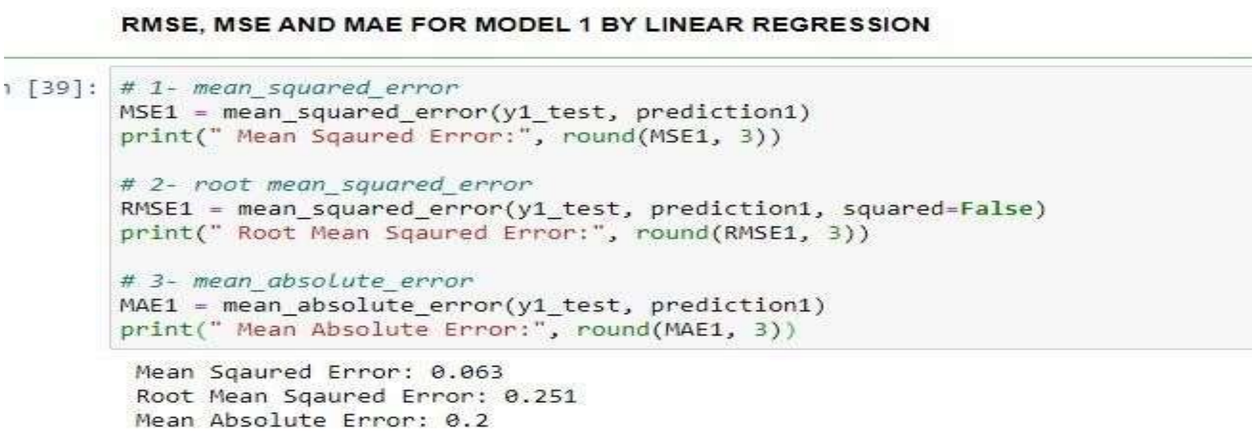
Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable.



```
#Passing Features to x1 and assigning our target to y1
x1 = Data_encoded.iloc[:, 0:11]

#here I have pick the GPA Column from our original dataset
y1 = Data['CGPA']
x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1, test_size=0.2, random_state=0)
Lin_reg = LinearRegression()
Y_pred1 = Lin_reg.fit(x1_train, y1_train).predict(x1_test)
print("BY LINEAR REGRESSION", Lin_reg.score(x1_test, y1_test))
```

CALCULATE RMSE,MSE AND MAE:

Calculating RMSE, MSE and MAE to check performance of our algorithm. From the code:



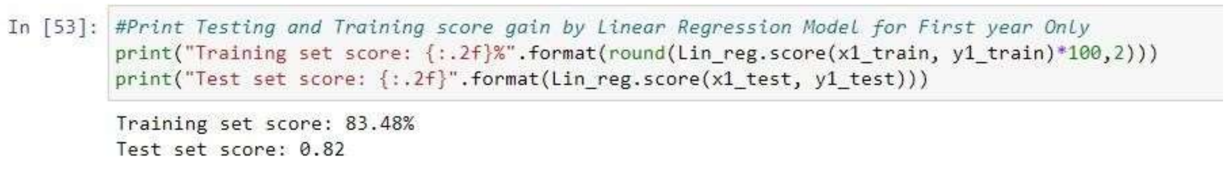
RMSE, MSE AND MAE FOR MODEL 1 BY LINEAR REGRESSION

```
[39]: # 1- mean_squared_error
MSE1 = mean_squared_error(y1_test, prediction1)
print(" Mean Sqaured Error:", round(MSE1, 3))

# 2- root mean_squared_error
RMSE1 = mean_squared_error(y1_test, prediction1, squared=False)
print(" Root Mean Sqaured Error:", round(RMSE1, 3))

# 3- mean_absolute_error
MAE1 = mean_absolute_error(y1_test, prediction1)
print(" Mean Absolute Error:", round(MAE1, 3))
```

```
Mean Sqaured Error: 0.063
Root Mean Sqaured Error: 0.251
Mean Absolute Error: 0.2
```

ACCURACY :

After training we have checked the testing and train score to find the accuracy of our model which comes out to be good. Which can be made better through other techniques . From the code:



```
In [53]: #Print Testing and Training score gain by Linear Regression Model for First year Only
print("Training set score: {:.2f}%".format(round(Lin_reg.score(x1_train, y1_train)*100,2)))
print("Test set score: {:.2f}".format(Lin_reg.score(x1_test, y1_test)))
```

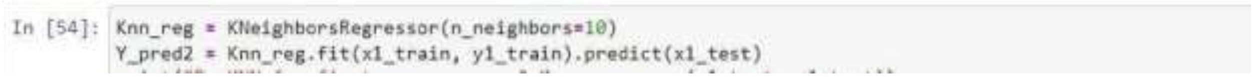
```
Training set score: 83.48%
Test set score: 0.82
```

2. K-Nearest Neighbor Regression Algorithm :

It is an algorithm for supervised machine learning. The letter "K" stands for the number of closest neighbors to a new unknown variable that needs to be forecasted or categorized. The inputs for this algorithm are courses of first year only. It is also called as lazy learner algorithm.

Training a K-Nearest Neighbor Regression Algorithm For Model 1:

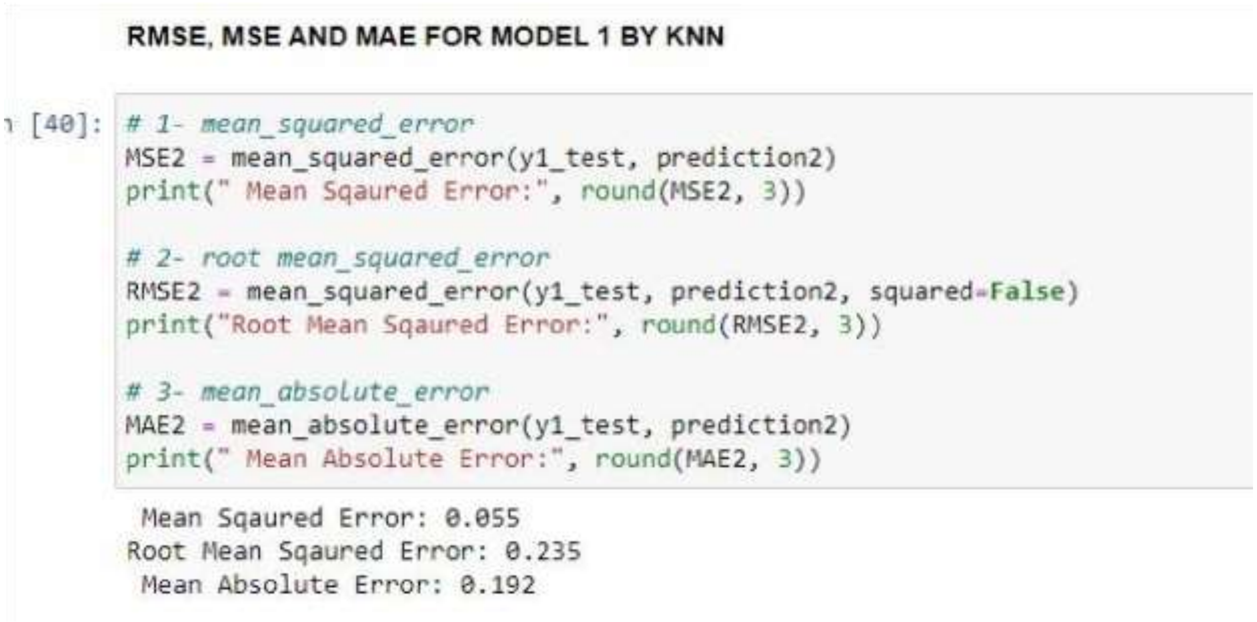
Here We have selected n_neighbors=10 to so we get optimal testing and training score. From the Code:



```
In [54]: Knn_reg = KNeighborsRegressor(n_neighbors=10)
         Y_pred2 = Knn_reg.fit(x1_train, y1_train).predict(x1_test)
```

CALCULATE RMSE,MSE AND MAE:

Calculating RMSE, MSE and MAE to check performance of our algorithm. From the code:

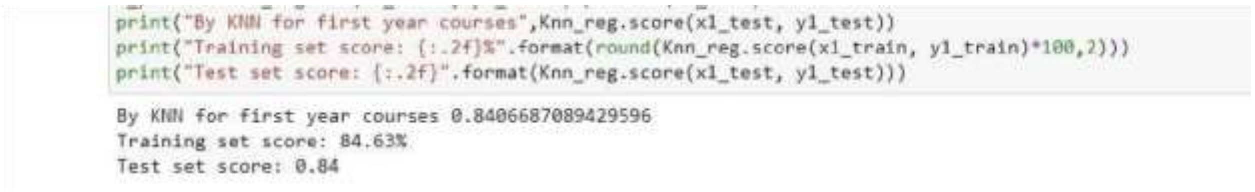


RMSE, MSE AND MAE FOR MODEL 1 BY KNN

```
n [40]: # 1- mean_squared_error
        MSE2 = mean_squared_error(y1_test, prediction2)
        print(" Mean Sqaured Error:", round(MSE2, 3))

        # 2- root mean_squared_error
        RMSE2 = mean_squared_error(y1_test, prediction2, squared=False)
        print("Root Mean Sqaured Error:", round(RMSE2, 3))

        # 3- mean_absolute_error
        MAE2 = mean_absolute_error(y1_test, prediction2)
        print(" Mean Absolute Error:", round(MAE2, 3))

         Mean Sqaured Error: 0.055
        Root Mean Sqaured Error: 0.235
         Mean Absolute Error: 0.192
```

ACCURACY

After training we have checked the testing and train score to find the accuracy of our model which is better than our previous algorithm for model 1. From the code:

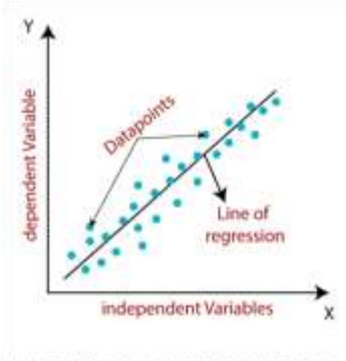


```
print("By KNN for first year courses",Knn_reg.score(x1_test, y1_test))
print("Training set score: {:.2f}%".format(round(Knn_reg.score(x1_train, y1_train)*100,2)))
print("Test set score: {:.2f}".format(Knn_reg.score(x1_test, y1_test)))

By KNN for first year courses 0.8406687089429596
Training set score: 84.63%
Test set score: 0.84
```

MODEL 2 (PREDICTING CGPA BASED ON FIRST TWO YEAR COURSES):

LINEAR REGRESSION:

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



Mathematically, we can represent a linear regression as:

- STEPS 1

First I've picked 22 columns with complete rows as the courses are increasing and assigned it to **x2** and all the features which are assigning to **x2** are from **Data_encoded** file.

- STEPS 2

Then I assigned the Target feature **CGPA** variable to **y2** from original **DATA** file.

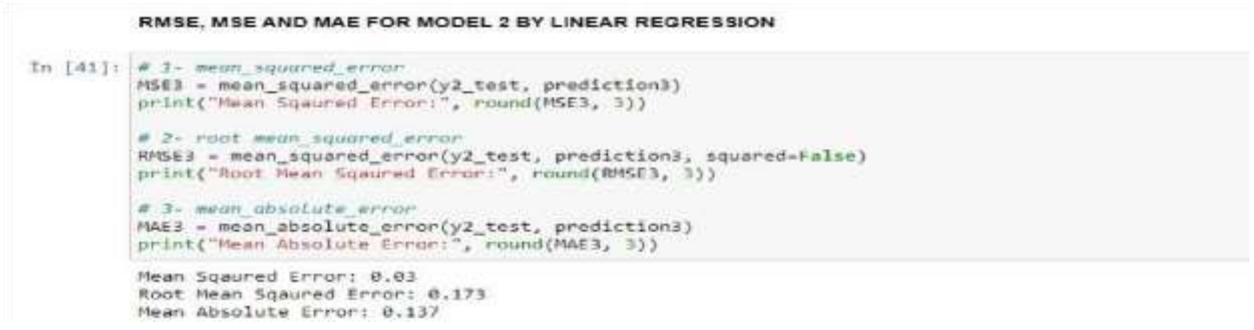
TRAINING A LINEAR REGRESSION FOR MODEL 2:

Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable.

```
In [11]: x2 = Data_encoded.iloc[:, 0:22]
         y2 = Data['CGPA']
         x2_train, x2_test, y2_train, y2_test = train_test_split(x2, y2, test_size=0.2, random_state=0)
```

CALCULATE RMSE,MSE AND MAE:

Calculating RMSE, MSE and MAE to check performance of our algorithm. From the code:

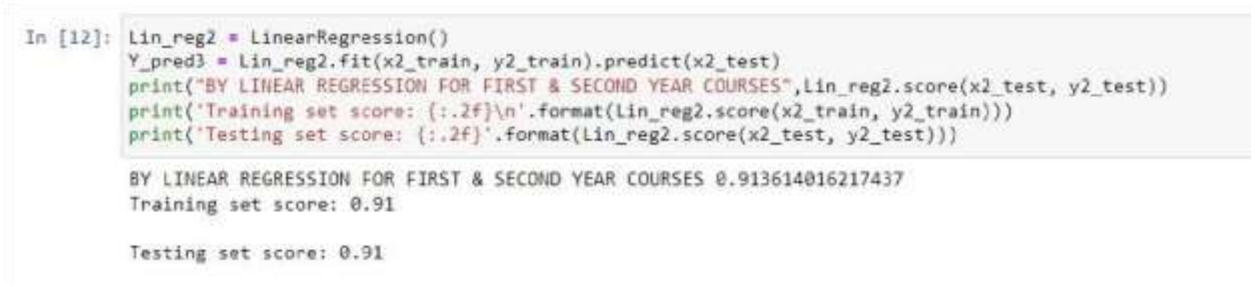


RMSE, MSE AND MAE FOR MODEL 2 BY LINEAR REGRESSION

```
In [41]: # 1- mean_squared_error
         MSE3 = mean_squared_error(y2_test, prediction3)
         print("Mean Sqaured Error:", round(MSE3, 3))

         # 2- root mean_squared_error
         RMSE3 = mean_squared_error(y2_test, prediction3, squared=False)
         print("Root Mean Sqaured Error:", round(RMSE3, 3))

         # 3- mean_absolute_error
         MAE3 = mean_absolute_error(y2_test, prediction3)
         print("Mean Absolute Error:", round(MAE3, 3))

         Mean Sqaured Error: 0.03
         Root Mean Sqaured Error: 0.173
         Mean Absolute Error: 0.137
```

ACCURACY :

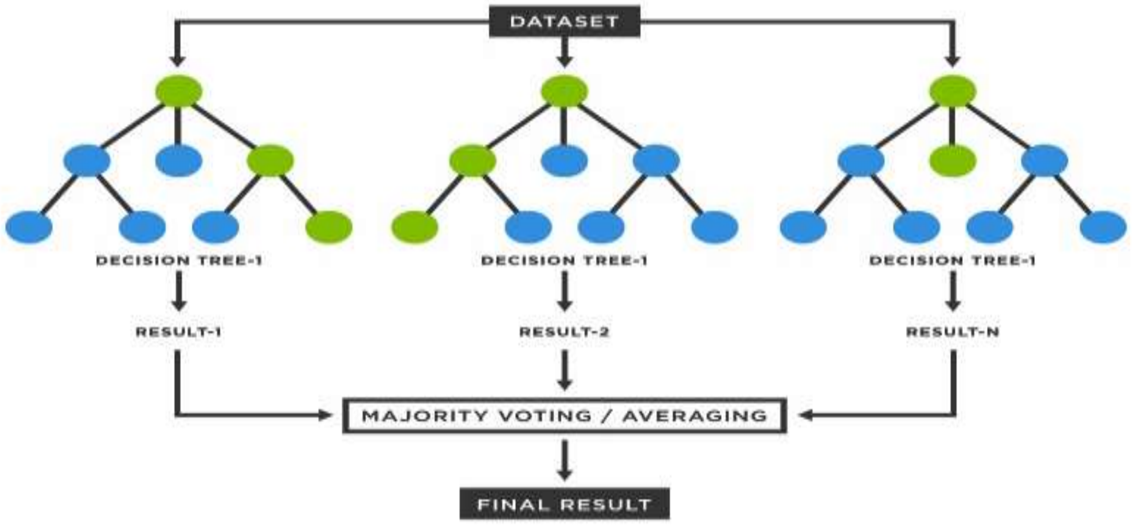
After training we have checked the testing and train score to find the accuracy of our model which comes out to much better than the model 1. This also shows that the if we increase the DATA it can significantly impact on our testing training score . From the code:



```
In [12]: Lin_reg2 = LinearRegression()
         Y_pred3 = Lin_reg2.fit(x2_train, y2_train).predict(x2_test)
         print("BY LINEAR REGRESSION FOR FIRST & SECOND YEAR COURSES",Lin_reg2.score(x2_test, y2_test))
         print('Training set score: {:.2f}\n'.format(Lin_reg2.score(x2_train, y2_train)))
         print('Testing set score: {:.2f}'.format(Lin_reg2.score(x2_test, y2_test)))

         BY LINEAR REGRESSION FOR FIRST & SECOND YEAR COURSES 0.913614016217437
         Training set score: 0.91

         Testing set score: 0.91
```

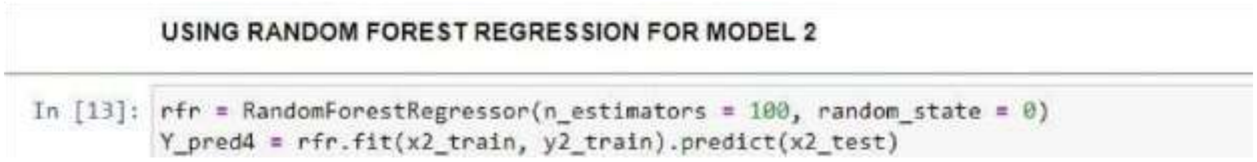
2.RANDOM FOREST ALGORITHM :

Random forest is **a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems**. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression



TRAINING A RANDOM FOREST ALGORITHM FOR MODEL 2:

Let's now begin to train out regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable. Here we **n_estimators=100** so it can generate more trees and it also had tradeoff that it'll make our algorithm slower

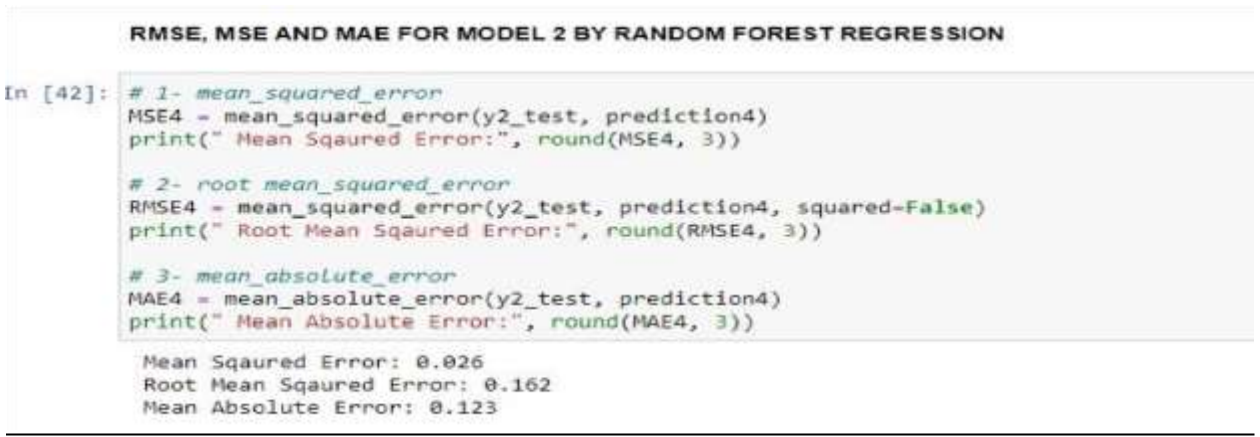


USING RANDOM FOREST REGRESSION FOR MODEL 2

```
In [13]: rfr = RandomForestRegressor(n_estimators = 100, random_state = 0)
         Y_pred4 = rfr.fit(x2_train, y2_train).predict(x2_test)
```

CALCULATE RMSE,MSE AND MAE:

Calculating RMSE, MSE and MAE to check performance of our algorithm which has decreasing values as compare to other models .From the code:

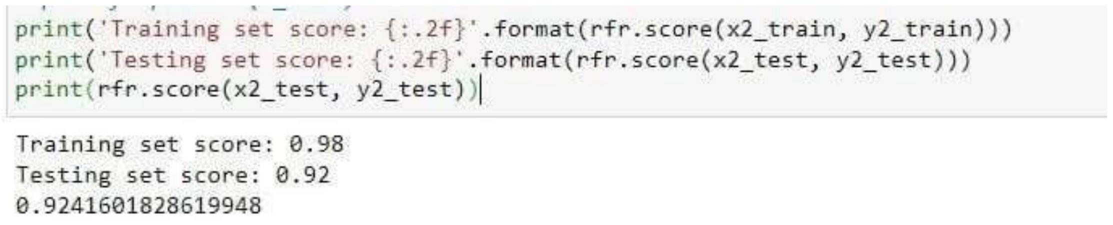


RMSE, MSE AND MAE FOR MODEL 2 BY RANDOM FOREST REGRESSION

```
In [42]: # 1- mean_squared_error
         MSE4 = mean_squared_error(y2_test, prediction4)
         print(" Mean Sqaured Error:", round(MSE4, 3))

         # 2- root mean_squared_error
         RMSE4 = mean_squared_error(y2_test, prediction4, squared=False)
         print(" Root Mean Sqaured Error:", round(RMSE4, 3))

         # 3- mean_absolute_error
         MAE4 = mean_absolute_error(y2_test, prediction4)
         print(" Mean Absolute Error:", round(MAE4, 3))

          Mean Sqaured Error: 0.026
          Root Mean Sqaured Error: 0.162
          Mean Absolute Error: 0.123
```

ACCURACY :

After training we have checked the testing and train score to find the accuracy of our model and we just got amazed by the results as the results are remarkable in this algorithm. From the code:

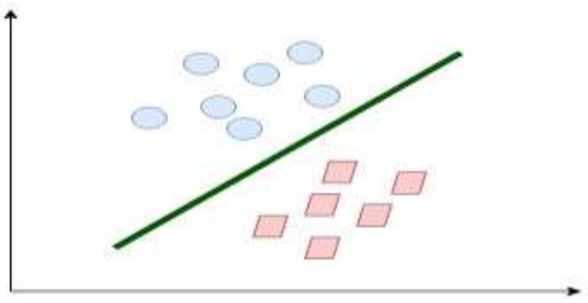


```
print('Training set score: {:.2f}'.format(rfr.score(x2_train, y2_train)))
print('Testing set score: {:.2f}'.format(rfr.score(x2_test, y2_test)))
print(rfr.score(x2_test, y2_test))

Training set score: 0.98
Testing set score: 0.92
0.9241601828619948
```

MODEL 3 (PREDICTING GPA BASED ON FIRST THREE YEAR COURSES):

1.SUPPORT VECTOR REGRESSION:

Support Vector Machines (SVM) are popularly and widely used for classification problems in machine learning. The problem of regression is to find a function that approximates mapping from an input domain to real numbers on the basis of a training sample



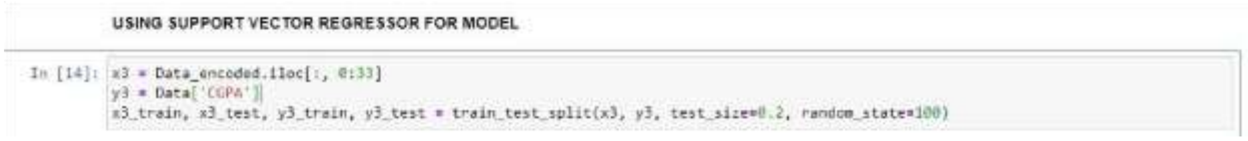
TRAINING A SUPPORT VECTOR REGRESSION FOR MODEL 3:

- STEPS 1

First I’ve picked 33 columns with complete rows as the courses are increasing and assigned it to **x3** and all the features which are assigning to **x3** are from **Data_encoded** file.

- STEPS 2

Then I assigned the Target feature **CGPA** variable to **y3** from original **DATA** file.

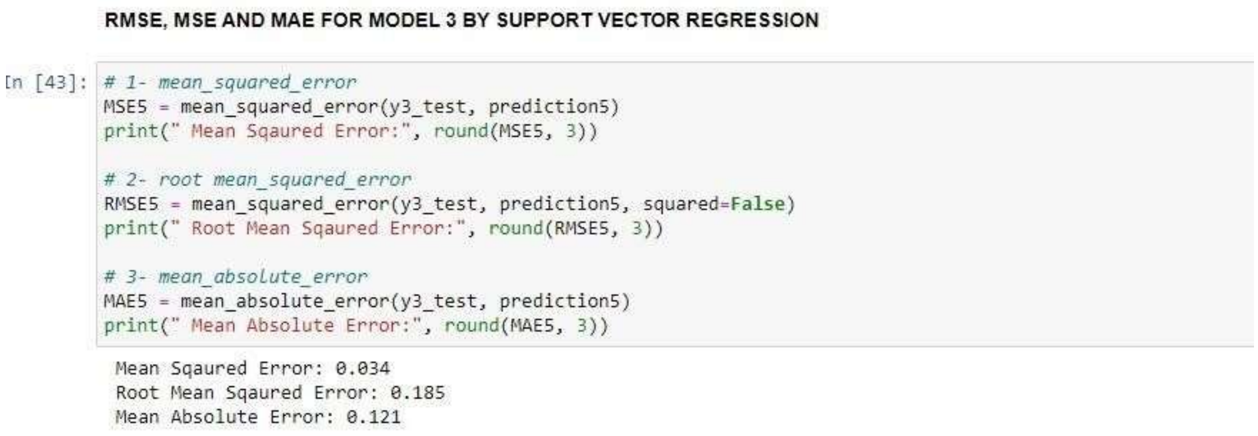


USING SUPPORT VECTOR REGRESSOR FOR MODEL

```
In [14]: x3 = Data_encoded.iloc[:, 0:33]
         y3 = Data['CGPA']
         x3_train, x3_test, y3_train, y3_test = train_test_split(x3, y3, test_size=0.2, random_state=100)
```

CALCULATE RMSE,MSE AND MAE

Calculating RMSE, MSE and MAE to check performance of our algorithm which has decreasing values as compare to other models .From the code:

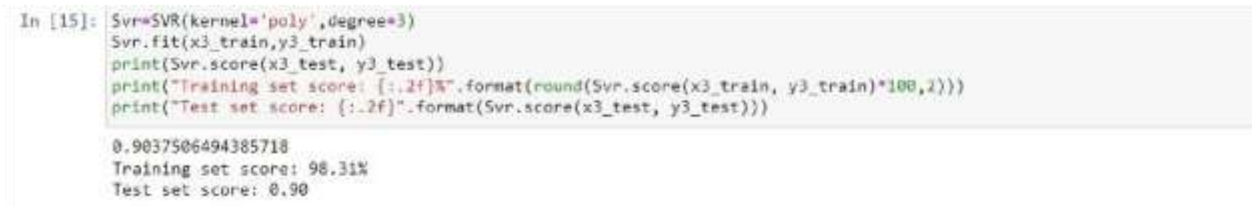


RMSE, MSE AND MAE FOR MODEL 3 BY SUPPORT VECTOR REGRESSION

```
In [43]: # 1- mean_squared_error
         MSE5 = mean_squared_error(y3_test, prediction5)
         print(" Mean Sqaured Error:", round(MSE5, 3))

         # 2- root mean_squared_error
         RMSE5 = mean_squared_error(y3_test, prediction5, squared=False)
         print(" Root Mean Sqaured Error:", round(RMSE5, 3))

         # 3- mean_absolute_error
         MAE5 = mean_absolute_error(y3_test, prediction5)
         print(" Mean Absolute Error:", round(MAE5, 3))

          Mean Sqaured Error: 0.034
          Root Mean Sqaured Error: 0.185
          Mean Absolute Error: 0.121
```

ACCURACY :

After training we have checked the testing and train score to find the accuracy of our model and we just got amazed by the results as the results are remarkable in this algorithm. From the code:



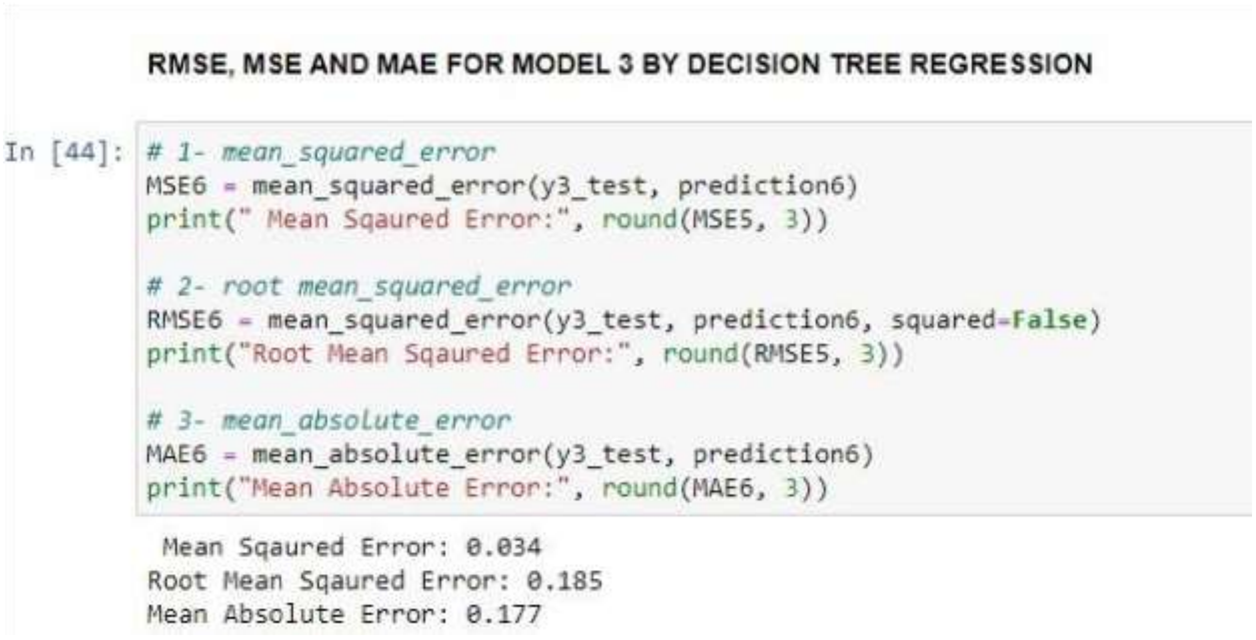
```
In [15]: Svr=SVR(kernel='poly',degree=3)
         Svr.fit(x3_train,y3_train)
         print(Svr.score(x3_test, y3_test))
         print("Training set score: {:.2f}%".format(round(Svr.score(x3_train, y3_train)*100,2)))
         print("Test set score: {:.2f}".format(Svr.score(x3_test, y3_test)))

         0.9037506494385718
         Training set score: 98.31%
         Test set score: 0.90
```

2.DECISION TREE REGRESSOR :

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/resul t is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

CALCULATE RMSE,MSE AND MAE:

Calculating RMSE, MSE and MAE to check performance of our algorithm which has decreasing values as compare to other models .From the code:

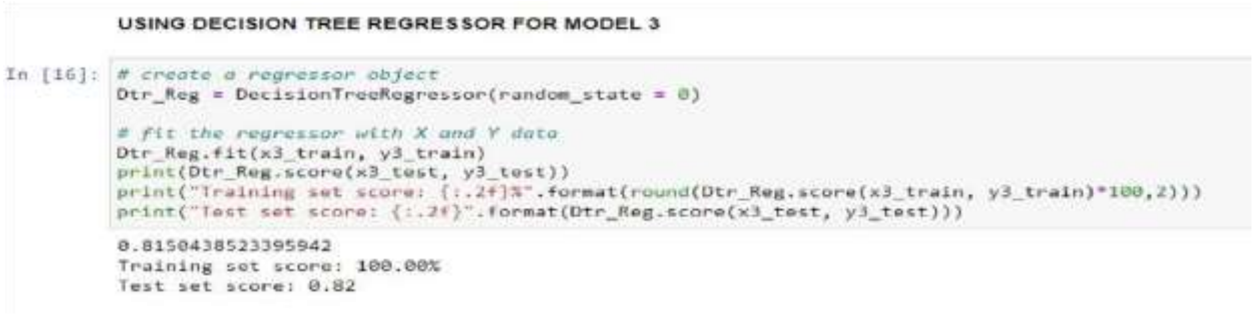


RMSE, MSE AND MAE FOR MODEL 3 BY DECISION TREE REGRESSION

```
In [44]: # 1- mean_squared_error
         MSE6 = mean_squared_error(y3_test, prediction6)
         print(" Mean Sqaured Error:", round(MSE5, 3))

         # 2- root mean_squared_error
         RMSE6 = mean_squared_error(y3_test, prediction6, squared=False)
         print("Root Mean Sqaured Error:", round(RMSE5, 3))

         # 3- mean_absolute_error
         MAE6 = mean_absolute_error(y3_test, prediction6)
         print("Mean Absolute Error:", round(MAE6, 3))

          Mean Sqaured Error: 0.034
         Root Mean Sqaured Error: 0.185
         Mean Absolute Error: 0.177
```

ACCURA CY :

After training we have checked the testing and train score to find the accuracy of our model and we can see that its situation of overfitting as it’s a low bias high variance result. From the code:

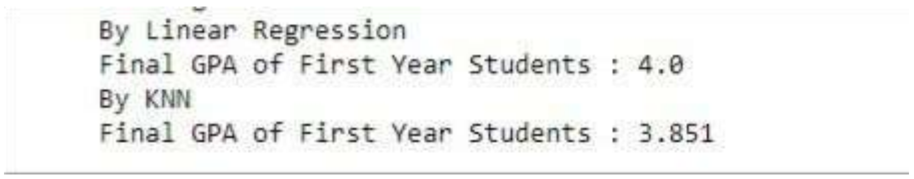


USING DECISION TREE REGRESSOR FOR MODEL 3

```
In [16]: # create a regressor object
         Dtr_Reg = DecisionTreeRegressor(random_state = 0)

         # fit the regressor with X and Y data
         Dtr_Reg.fit(x3_train, y3_train)
         print(Dtr_Reg.score(x3_test, y3_test))
         print("Training set score: {:.2f}%".format(round(Dtr_Reg.score(x3_train, y3_train)*100,2)))
         print("Test set score: {:.2f}".format(Dtr_Reg.score(x3_test, y3_test)))

         0.8150438523395942
         Training set score: 100.00%
         Test set score: 0.82
```

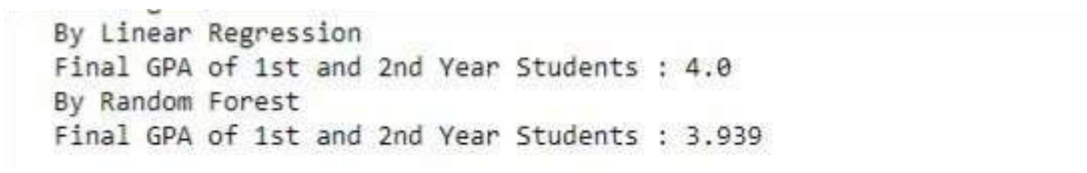
OUTPUTS :

We have provided A+ grades for all courses and the model 1 algorithms has predicted this :



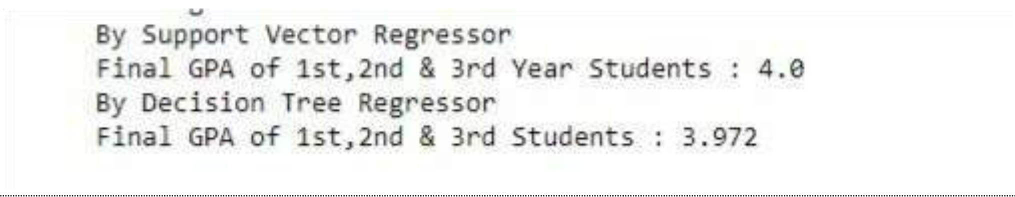
```
By Linear Regression
Final GPA of First Year Students : 4.0
By KNN
Final GPA of First Year Students : 3.851
```

We have provided A+ grades for all courses and the model 2 algorithms has predicted this :



```
By Linear Regression
Final GPA of 1st and 2nd Year Students : 4.0
By Random Forest
Final GPA of 1st and 2nd Year Students : 3.939
```

We have provided A+ grades for all courses and the model 3 algorithms has predicted this :



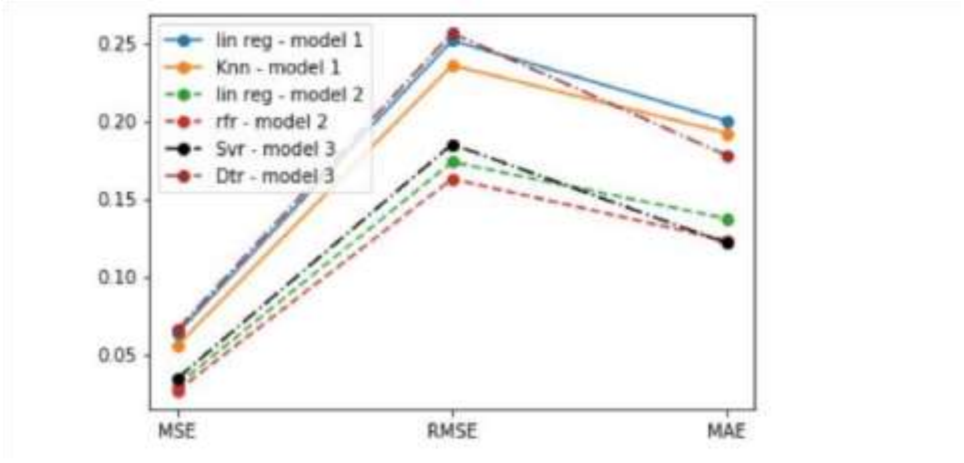
```
By Support Vector Regressor
Final GPA of 1st,2nd & 3rd Year Students : 4.0
By Decision Tree Regressor
Final GPA of 1st,2nd & 3rd Students : 3.972
```

GRAPH:

This graph illustrates the complete scanerio of which model's algorithm works best in our case and from this we have concluded that :

- SVR of model 3
- RFR model 2
- Linear regression model 2

These 3 models works excellent in our case.



REFRENCES:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

https://www.kaggle.com/code/faressayah/practical-introduction-to-10-regressionalgorithm?scriptVersionId=85709616

https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machinelearning/ https://www.youtube.com/c/codebasics/videos

https://github.com/topics/knn-classification