# STUDENT GRADE ANALYSIS AND PREDICTION

**By**

| | |
|---|---|
| R. Hanuman koushik | 2011CS040068 |
| P. Chandradeep | 2011CS040062 |
| R. Akshay kumar | 2011CS040069 |
| K. Gopi chand | 2011CS040099 |

**GUIDED BY**

**Dr. Meeravali Shaik**

**Department of Computer Science & Engineering (Cyber Security)**

**III$^{rd}$ year I$^{st}$ semester Malla Reddy University, Hyderabad**

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

# CERTIFICATE

This is to certify that this is the bonafide record of the application development entitled **"STUDENT GRADE ANALYSIS AND PREDICTION"**, submitted by **R Hanuman Koushik 2011CS040068, K Gopichand 2011CS040099, R Akshay kumar 2011CS040069, P Chandradeep 2011CS040062.** B. Tech III$^{rd}$ year I$^{st}$ semester, Department of CSE (CS) during the year 2022-23. The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

| | |
|---|---|
| **Guide** | **Head of the department** |
| **Dr. Meeravali Shaik** | **Dr. Meeravali Shaik** |
| | **CSE(Cyber Security)** |

**External Examiner**

# ACKNOWLEDGEMENT

We would like to express our gratitude to all those who extended their support and suggestions to come up with this application. Special Thanks to our mentor Dr. Meeravali Shaik whose help and stimulating suggestions and encouragement helped us all time in the due course of project development.

We sincerely thank our HOD Dr.Meeravali Shaik for his constant support and motivation all the time. A special acknowledgement goes to a friend who enthused us from the backstage. Last but not the least our sincere appreciation goes to our family who has been tolerant understanding our moods and extending timely support.

# ABSTRACT

Performance analysis of outcome based on learning is a system which will strive for excellence at different levels and diverse dimensions in the field of student's interests. This system developed to analyze and predict the student's performance only. The proposed framework analyzes the students' demographic data, study related and psychological characteristics to extract all possible knowledge from students, teachers and parents. Seeking the highest possible accuracy in academic performance prediction using a set of powerful data mining techniques. The framework succeeds to highlight the student's weak points . The realistic case study that has been conducted on 200 students proves the outstanding performance of the proposed framework in comparison with the existing ones.

# INDEX

| Contents | Page No. |
|---|---|

# CHAPTER - 1

# INTRODUCTION

## 1.1 Introduction

- Machine learning is the study of computer algorithms that improve automatically through experience. The algorithms build a model on sample data known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. It is sometimes related to computational analysis, and it is also referred as predictive analysis.
- This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school-related features) and it was collected by using school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por). In [Cortez and Silva, 2008], the two data sets were modeled under binary/five-level classification and regression tasks. Important note: the target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful (see paper source for more details).

## 1.2 Problem Statement

The problem statement can be defined as follows "Given a dataset containing attribute of 396 students where using the features available from dataset and define classification algorithms to identify whether the student performs good in final grade exam, also to evaluate different machine learning models on the dataset."

### 1.3 Objective

Our objective is to build a model that would predict whether or not a student would fail the subject or course that was being tracked. I focused on failure rates as we believed that metric to be more valuable in terms of flagging struggling students who may need more help.

To be able to pre-emptively assess which students may need the most attention is, in my opinion, an important step to personalized education.

## 1.4 Goal

- Our project target and ambition is to provide accurate prediction of student's grade performance and present analysis of the student

# CHAPTER - 2

# EXISTING AND PROPOSED SYSTEMS

## 2.1 Existing systems

The previous predictive models only focused on using the student's demographic data like gender, age, family status, family income and qualifications. In addition to the study related attributes including the homework and study hours as well as the previous achievements and grades. These previous work were only limited to provide the prediction of the academic success or failure, without illustrating the reasons of this prediction. Most of the previous researches have focused to gather more than 40 attributes in their data set to predict the student's academic performance. These attributes were from the same type of data category whether demographic, study related attributes or both, that lead to lack of diversity of predicting rules.

## 2.2 Proposed system

Without any prior academic performance in similar courses, the problem is difficult to solve; however, my model achieves 68% accuracy using only the school the student attends and the number of absences that they accrue to judge whether or not they fail. What is interesting is that my model, with these parameters, has a false pass rate of over 50%, meaning that it classifies more than half of the students who end up failing as passing instead. This number falls drastically as more information becomes available and better parameters are used, but it highlights one major area of improvement for the model.

To achieve their performance noted above, the original authors had to alternate models for each experiment, using both support vector machines and naive bayes. My support vector machine's performance closely follows the original author's results and displays a more streamlined approach to solving the problem, as the underlying model does not change. In addition, the original authors made use of all variables (excluding grade knowledge) in achieving the stated 70.6% accuracy in the third experiment, while my model makes use of only two parameters at a time to achieve similar results.

# CHAPTER - 3 REQUIREMENTS

## 3.1 Software Requirements:

- ❖ Python 3.98
- ❖ Anaconda(Jupyter)
- ❖ Programming Language: Python, HTML, CSS
- ❖ Necessary libraries ○ flask ○ matplot ○ tkinter ○ pillow ○ pandas ○ numpy

### 3.2 Hardware Requirements:

- ❖ Processor: Intel Core i5
- ❖ Mother board: Any Motherboard which support i5 processor
- ❖ Ram: 4GB DDR4 ram

# CHAPTER - 4

# DESIGN AND IMPLEMENTATION

## 4.1 . Description of the Dataset

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and school-related features) and it was collected by using school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por). In [Cortez and Silva, 2008], the two data sets were modeled under binary/five-level classification and regression tasks. Important note: the target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), while G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful (see paper source for more details).

The target value is G3, which, according to the accompanying paper of the dataset, can be binned into a passing or failing classification. If G3 is greater than or equal to 10, then the student passes. Otherwise, she fails. Likewise, the G1 and G2 features are binned in the same manner.

The data can be reduced to 4 fundamental features, in order of importance:

1. G2 score
2. G1 score
3. School
4. Absences

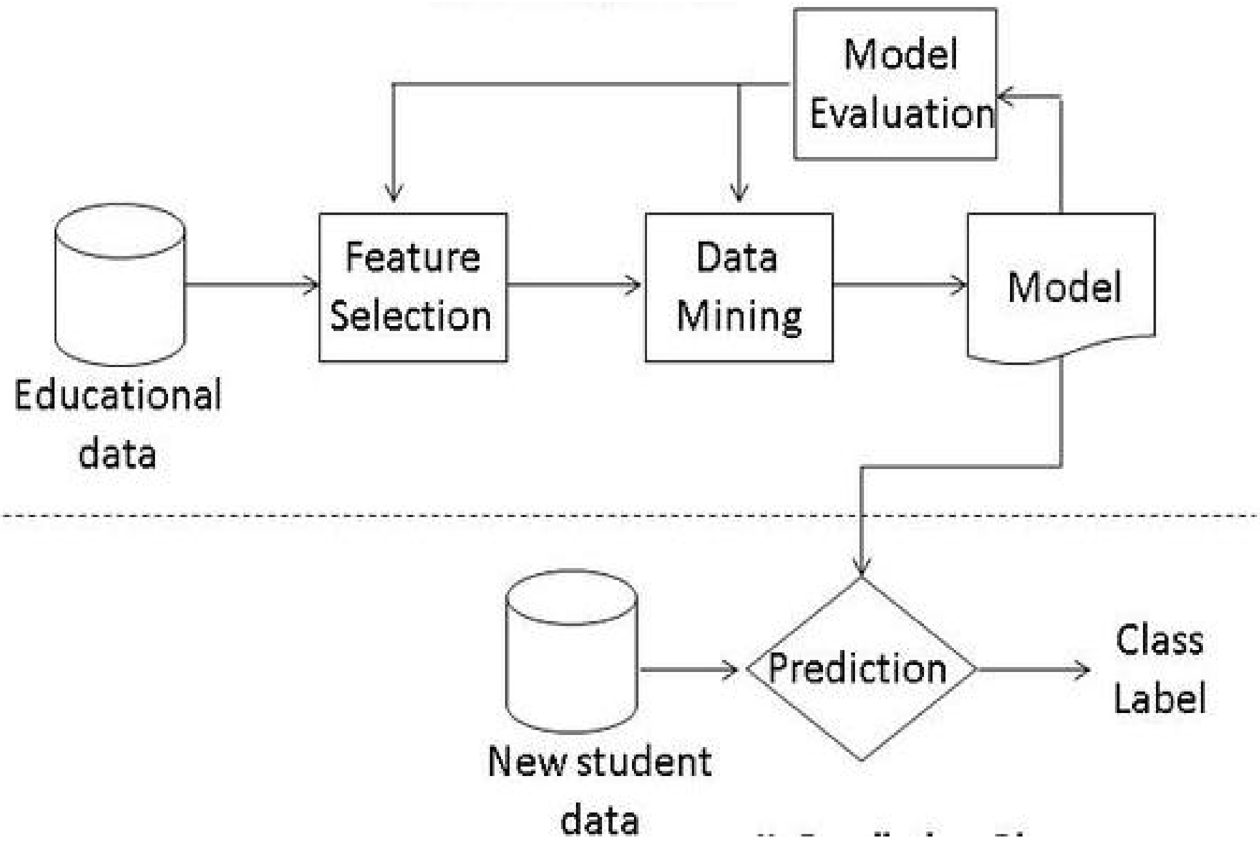
When no grade knowledge is known, School and Absences capture most of the predictive basis. As grade knowledge becomes available, G1 and G2 scores alone are enough to achieve over 90% accuracy. I experimentally discovered that the model performs best when it uses only 2 features at a time for each experiment. The model is a linear support vector machine with a regularization factor of 100. This model performed the best when compared to other models, such as naive bayes, logistic regression, and random forest classifiers.

## 4.2 . Attribute Information

- school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
- sex - student's sex (binary: 'F' - female or 'M' - male)
- age - student's age (numeric: from 15 to 22)
- address - student's home address type (binary: 'U' - urban or 'R' - rural)
- famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
- Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
- Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - “ 5th to 9th grade, 3 - “ secondary education or 4 - “ higher education)
- Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - “ 5th to 9th grade, 3 - “ secondary education or 4 - “ higher education)
- Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
- Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
- reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
- guardian - student's guardian (nominal: 'mother', 'father' or 'other')
- traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- failures - number of past class failures (numeric: n if 1<=n<3, else 4)
- schoolsup - extra educational support (binary: yes or no)

- famsup - family educational support (binary: yes or no)
- paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- activities - extra-curricular activities (binary: yes or no)
- nursery - attended nursery school (binary: yes or no)
- higher - wants to take higher education (binary: yes or no)
- internet - Internet access at home (binary: yes or no)
- romantic - with a romantic relationship (binary: yes or no)
- famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- freetime - free time after school (numeric: from 1 - very low to 5 - very high)
- goout - going out with friends (numeric: from 1 - very low to 5 - very high)
- Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- health - current health status (numeric: from 1 - very bad to 5 - very good)
- absences - number of school absences (numeric: from 0 to 93)

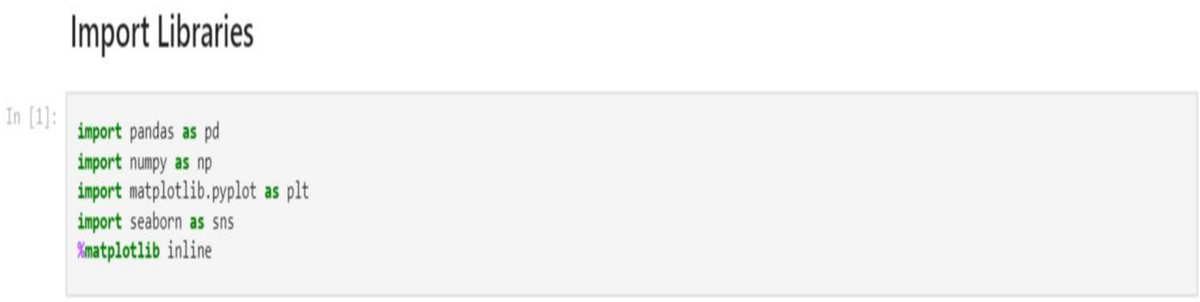
### 4.3 Block Diagram



## 4.4 Source Code and Implementation

### Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```



## Read data

In [2]:
```
stud= pd.read_csv('student-mat.csv')    # Read the dataset
```

In [3]:
```
print('Total number of students:',len(stud))
```

```
Total number of students: 395
```

In [4]:
```
stud['G3'].describe()
```

Out[4]:
```
count    395.000000
mean      10.415190
std        4.581443
min        0.000000
25%        8.000000
50%       11.000000
75%       14.000000
max       20.000000
Name: G3, dtype: float64
```

In [5]:
```
stud.info()    # Information on dataset
```

```
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
school        395 non-null object
sex           395 non-null object
age           395 non-null int64
address       395 non-null object
famsize       395 non-null object
Pstatus       395 non-null object
Medu          395 non-null int64
Fedu          395 non-null int64
Mjob          395 non-null object
Fjob          395 non-null object
reason        395 non-null object
guardian      395 non-null object
traveltime    395 non-null int64
studytime     395 non-null int64
failures      395 non-null int64
schoolsup     395 non-null object
famsup        395 non-null object
paid          395 non-null object
activities    395 non-null object
nursery       395 non-null object
higher        395 non-null object
internet      395 non-null object
romantic      395 non-null object
famrel        395 non-null int64
freetime      395 non-null int64
goout         395 non-null int64
Dalc          395 non-null int64
Walc          395 non-null int64
health        395 non-null int64
absences      395 non-null int64
G1            395 non-null int64
G2            395 non-null int64
G3            395 non-null int64
dtypes: int64(16), object(17)
memory usage: 101.9+ KB
```

In [6]:
```
stud.columns    # Dataset Columns
```

Out[6]:
```
Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

In [7]:
```
stud.describe()    # Dataset description
```

Out[7]:

| | age | Medu | Fedu | traveltime | studytime | failures | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 395.000000 | 39 |
| mean | 16.696203 | 2.749367 | 2.521519 | 1.448101 | 2.035443 | 0.334177 | 3.944304 | 3.235443 | 3.108861 | 1.481013 | 2.291139 | 3.554430 | 5.708861 | 10.908861 | 1 |
| std | 1.276043 | 1.094735 | 1.088201 | 0.697505 | 0.839240 | 0.743651 | 0.896659 | 0.998862 | 1.113278 | 0.890741 | 1.287897 | 1.390303 | 8.003096 | 3.319195 | |
| min | 15.000000 | 0.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 | 3.000000 | |
| 25% | 16.000000 | 2.000000 | 2.000000 | 1.000000 | 1.000000 | 0.000000 | 4.000000 | 3.000000 | 2.000000 | 1.000000 | 1.000000 | 3.000000 | 0.000000 | 8.000000 | |
| 50% | 17.000000 | 3.000000 | 2.000000 | 1.000000 | 2.000000 | 0.000000 | 4.000000 | 3.000000 | 3.000000 | 1.000000 | 2.000000 | 4.000000 | 4.000000 | 11.000000 | 1 |
| 75% | 18.000000 | 4.000000 | 3.000000 | 2.000000 | 2.000000 | 0.000000 | 5.000000 | 4.000000 | 4.000000 | 2.000000 | 3.000000 | 5.000000 | 8.000000 | 13.000000 | 1 |
| max | 22.000000 | 4.000000 | 4.000000 | 4.000000 | 4.000000 | 3.000000 | 5.000000 | 5.000000 | 5.000000 | 5.000000 | 5.000000 | 5.000000 | 75.000000 | 19.000000 | 1 |

In [8]:
```
stud.head()    # First 5 values of dataset
```

Out[8]:

| | school | sex | age | address | famsize | Pstatus | Medu | Fedu | Mjob | Fjob | ... | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | G2 | G3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | GP | F | 18 | U | GT3 | A | 4 | 4 | at_home | teacher | ... | 4 | 3 | 4 | 1 | 1 | 3 | 6 | 5 | 6 | 6 |
| 1 | GP | F | 17 | U | GT3 | T | 1 | 1 | at_home | other | ... | 5 | 3 | 3 | 1 | 1 | 3 | 4 | 5 | 5 | 6 |
| 2 | GP | F | 15 | U | LE3 | T | 1 | 1 | at_home | other | ... | 4 | 3 | 2 | 2 | 3 | 3 | 10 | 7 | 8 | 10 |
| 3 | GP | F | 15 | U | GT3 | T | 4 | 2 | health | services | ... | 3 | 2 | 2 | 1 | 1 | 5 | 2 | 15 | 14 | 15 |
| 4 | GP | F | 16 | U | GT3 | T | 3 | 3 | other | other | ... | 4 | 3 | 2 | 1 | 2 | 5 | 4 | 6 | 10 | 10 |

5 rows × 33 columns

In [9]:
```
stud.tail()    # Last 5 values of dataset
```

Out[9]:

| | school | sex | age | address | famsize | Pstatus | Medu | Fedu | Mjob | Fjob | ... | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | G2 | G3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 390 | MS | M | 20 | U | LE3 | A | 2 | 2 | services | services | ... | 5 | 5 | 4 | 4 | 5 | 4 | 11 | 9 | 9 | 9 |
| 391 | MS | M | 17 | U | LE3 | T | 3 | 1 | services | services | ... | 2 | 4 | 5 | 3 | 4 | 2 | 3 | 14 | 16 | 16 |
| 392 | MS | M | 21 | R | GT3 | T | 1 | 1 | other | other | ... | 5 | 5 | 3 | 3 | 3 | 3 | 3 | 10 | 8 | 7 |
| 393 | MS | M | 18 | R | LE3 | T | 3 | 2 | services | other | ... | 4 | 4 | 1 | 3 | 4 | 5 | 0 | 11 | 12 | 10 |
| 394 | MS | M | 19 | U | LE3 | T | 1 | 1 | other | at_home | ... | 3 | 2 | 3 | 3 | 3 | 5 | 5 | 8 | 9 | 9 |

5 rows × 33 columns

In [10]:
```
stud.isnull().any()    # To check any null values present in dataset
```

Out[10]:
```
school        False
sex           False
age           False
address       False
famsize       False
Pstatus       False
Medu          False
Fedu          False
Mjob          False
Fjob          False
reason        False
guardian      False
traveltime    False
studytime     False
failures      False
schoolsup     False
famsup        False
paid          False
activities    False
nursery       False
```

In [53]:
```
import cufflinks as cf
cf.go_offline()
```

In [55]:
```
stud.iplot()     # Plot for the all attributes
```

In [13]:
```
stud.iplot(kind='scatter',x='age',y='G3',mode='markers',size=8)     # Plot for age vs G3
```

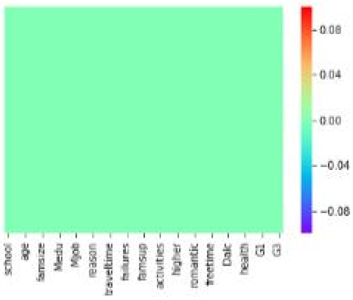
In [14]:
```
stud.iplot(kind='box')
```

In [15]:
```
stud['G3'].iplot(kind='hist',bins=100,color='blue')
```

# Data Visualization

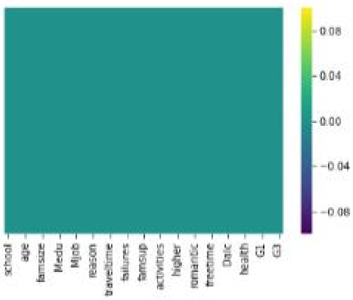
In [16]:
```
sns.heatmap(stud.isnull(),cmap="rainbow",yticklabels=False)     # To check any null values present in dataset pictorially
```

Out[16]:




In [17]:
```
sns.heatmap(stud.isnull(),cmap="viridis",yticklabels=False)     # Map color - viridis
```

Out[17]:




- There are no null values in the given dataset

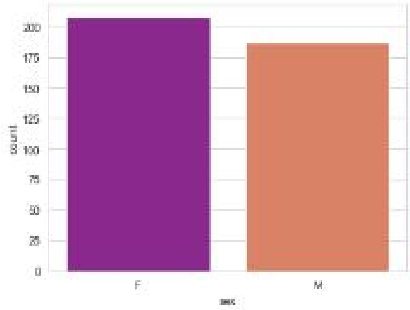
## Student's Sex

In [18]:
```
f_stud = len(stud[stud['sex'] == 'F'])     # Number of female students
print('Number of female students:',f_stud)
m_stud = len(stud[stud['sex'] == 'M'])     # Number of male students
print('Number of male students:',m_stud)
```

```
Number of female students: 208
Number of male students: 187
```

In [19]:
```
sns.set_style('whitegrid')     # male & female student representaion on countplot
sns.countplot(x='sex',data=stud,palette='plasma')
```

Out[19]:

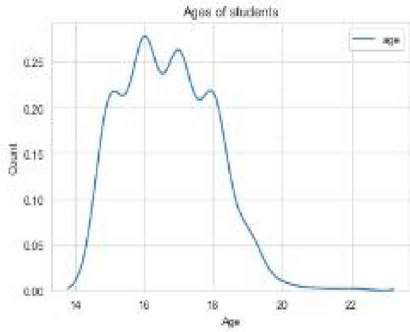



- The gender distribution is pretty even.

## Age of Students

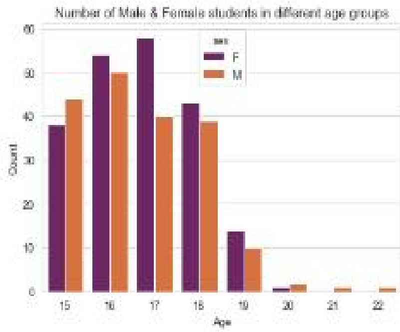
In [20]:

```
b = sns.kdeplot(stud['age'])    # Kernel Density Estimations
b.axes.set_title('Ages of students')
b.set_xlabel('Age')
b.set_ylabel('Count')
plt.show()
```




In [21]:

```
b = sns.countplot(x='age',hue='sex', data=stud, palette='inferno')
b.axes.set_title('Number of Male & Female students in different age groups')
b.set_xlabel("Age")
b.set_ylabel("Count")
plt.show()
```



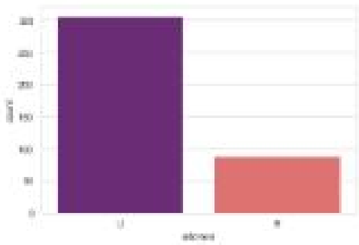

- The student age seems to be ranging from 15-19, where gender distribution is pretty even in each age group.
- The age group above 19 may be outliers, year back students or droupouts.

## Students from Urban & Rural Areas

```
u_stud = len(stud[stud['address'] == 'U'])    # Number of urban areas students
print('Number of urban students:',u_stud)
r_stud = len(stud[stud['address'] == 'R'])    # Number of rural areas students
print('Number of Rural students:',r_stud)
```

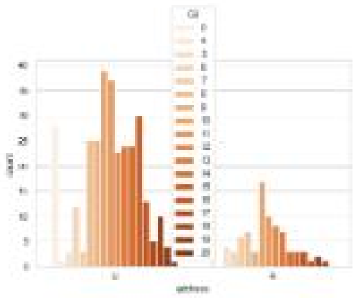
```
Number of urban students: 307
Number of Rural students: 88
```

```
sns.set_style('whitegrid')
sns.countplot(x='address',data=stud,palette='magma')    # urban & rural representaion on countplot
```




- Approximately *77.72%* students come from urban region and *22.28%* from rural region.

```
sns.countplot(x='address',hue='G3',data=stud,palette='Oranges')
```

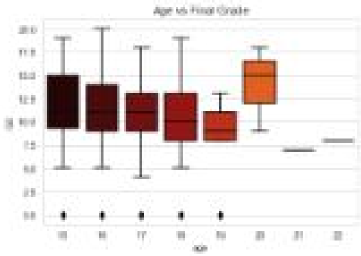



## EDA - Exploratory Data Analysis

### 1. Does age affect final grade?

```
b= sns.boxplot(x='age', y='G3',data=stud,palette='gist_heat')
b.axes.set_title('Age vs Final Grade')
```

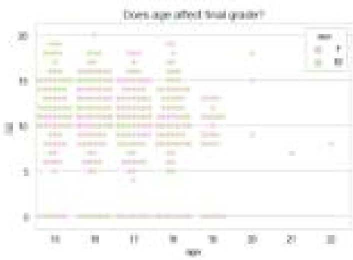
```
Text(0.5, 1.0, 'Age vs Final Grade')
```




- Plotting the distribution rather than statistics would help us better understand the data.
- The above plot shows that the median grades of the three age groups(15,16,17) are similar. Note the skewness of age group 19. (may be due to sample size). Age group 20 seems to score highest grades among all.

```
b = sns.swarmplot(x='age', y='G3',hue='sex', data=stud,palette='PiYG')
b.axes.set_title('Does age affect final grade?')
```

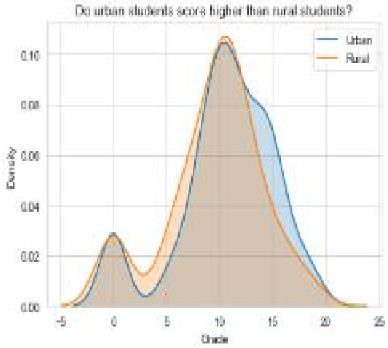
```
Text(0.5, 1.0, 'Does age affect final grade?')
```



## 2. Do urban students perform better than rural students?

In [27]:

```
# Grade distribution by address
sns.kdeplot(stud.loc[stud['address'] == 'U', 'G3'], label='Urban', shade = True)
sns.kdeplot(stud.loc[stud['address'] == 'R', 'G3'], label='Rural', shade = True)
plt.title('Do urban students score higher than rural students?')
plt.xlabel('Grade');
plt.ylabel('Density')
plt.show()
```




- The above graph clearly shows there is not much difference between the grades based on location.

In [28]:

```
stud.corr()['G3'].sort_values()
```

Out[28]:

```
failures     -0.360415
age          -0.161579
goout        -0.132791
traveltime   -0.117142
health       -0.061335
Dalc         -0.054660
Walc         -0.051939
freetime      0.011307
absences      0.034247
famrel        0.051363
studytime     0.097820
Fedu          0.152457
Medu          0.217147
G1            0.801468
G2            0.904868
G3            1.000000
Name: G3, dtype: float64
```

## Encoding categorical variables using LabelEncoder()

In [29]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
stud.iloc[:,0]=le.fit_transform(stud.iloc[:,0])
stud.iloc[:,1]=le.fit_transform(stud.iloc[:,1])
stud.iloc[:,3]=le.fit_transform(stud.iloc[:,3])
stud.iloc[:,4]=le.fit_transform(stud.iloc[:,4])
stud.iloc[:,5]=le.fit_transform(stud.iloc[:,5])
stud.iloc[:,8]=le.fit_transform(stud.iloc[:,8])
stud.iloc[:,9]=le.fit_transform(stud.iloc[:,9])
stud.iloc[:,10]=le.fit_transform(stud.iloc[:,10])
stud.iloc[:,11]=le.fit_transform(stud.iloc[:,11])
stud.iloc[:,15]=le.fit_transform(stud.iloc[:,15])
stud.iloc[:,16]=le.fit_transform(stud.iloc[:,16])
stud.iloc[:,17]=le.fit_transform(stud.iloc[:,17])
stud.iloc[:,18]=le.fit_transform(stud.iloc[:,18])
stud.iloc[:,19]=le.fit_transform(stud.iloc[:,19])
stud.iloc[:,20]=le.fit_transform(stud.iloc[:,20])
stud.iloc[:,21]=le.fit_transform(stud.iloc[:,21])
stud.iloc[:,22]=le.fit_transform(stud.iloc[:,22])
```

In [30]:

```
stud.head()
```

Out[ ]:

| | school | sex | age | address | famsize | Pstatus | Medu | Fedu | Mjob | Fjob | ... | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | G2 | G3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 18 | 1 | 0 | 0 | 4 | 4 | 0 | 4 | ... | 4 | 3 | 4 | 1 | 1 | 3 | 6 | 5 | 6 | 6 |
| 1 | 0 | 0 | 17 | 1 | 0 | 1 | 1 | 1 | 0 | 2 | ... | 5 | 3 | 3 | 1 | 1 | 3 | 4 | 5 | 5 | 6 |
| 2 | 0 | 0 | 15 | 1 | 1 | 1 | 1 | 1 | 0 | 2 | ... | 4 | 3 | 2 | 2 | 3 | 3 | 10 | 7 | 8 | 10 |
| 3 | 0 | 0 | 15 | 1 | 0 | 1 | 4 | 2 | 1 | 3 | ... | 3 | 2 | 2 | 1 | 1 | 5 | 2 | 15 | 14 | 15 |
| 4 | 0 | 0 | 16 | 1 | 0 | 1 | 3 | 3 | 2 | 2 | ... | 4 | 3 | 2 | 1 | 2 | 5 | 4 | 6 | 10 | 10 |

5 rows × 33 columns

In [ ]:

```
stud.tail()
```

Out[ ]:

| | school | sex | age | address | famsize | Pstatus | Medu | Fedu | Mjob | Fjob | ... | famrel | freetime | goout | Dalc | Walc | health | absences | G1 | G2 | G3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 390 | 1 | 1 | 20 | 1 | 1 | 0 | 2 | 2 | 3 | 3 | ... | 5 | 5 | 4 | 4 | 5 | 4 | 11 | 9 | 9 | 9 |
| 391 | 1 | 1 | 17 | 1 | 1 | 1 | 3 | 1 | 3 | 3 | ... | 2 | 4 | 5 | 3 | 4 | 2 | 3 | 14 | 16 | 16 |
| 392 | 1 | 1 | 21 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | ... | 5 | 5 | 3 | 3 | 3 | 3 | 3 | 10 | 8 | 7 |
| 393 | 1 | 1 | 18 | 0 | 1 | 1 | 3 | 2 | 3 | 2 | ... | 4 | 4 | 1 | 3 | 4 | 5 | 0 | 11 | 12 | 10 |
| 394 | 1 | 1 | 19 | 1 | 1 | 1 | 1 | 1 | 2 | 0 | ... | 3 | 2 | 3 | 3 | 3 | 5 | 5 | 8 | 9 | 9 |

5 rows × 33 columns

In [ ]:

```
stud.corr()['G3'].sort_values()    # Correlation wrt G3
```

Out[ ]:

```
failures     -0.360415
age          -0.161579
goout        -0.132791
romantic     -0.129970
traveltime   -0.117142
schoolsup    -0.082788
guardian     -0.070109
health       -0.061335
Pstatus      -0.058009
Dalc         -0.054660
Walc         -0.051939
school       -0.045017
famsup       -0.039157
freetime      0.011307
activities    0.016100
absences      0.034247
Fjob          0.042286
famrel        0.051363
nursery       0.051568
famsize       0.081407
studytime     0.097820
internet      0.098483
paid          0.101996
Mjob          0.102082
sex           0.103456
address       0.105756
reason        0.121994
Fedu          0.152457
higher        0.182465
Medu          0.217147
G1            0.801468
G2            0.904868
G3            1.000000
Name: G3, dtype: float64
```

In [ ]:

```
# drop the school and grade columns
stud = stud.drop(['school', 'G1', 'G2'], axis='columns')
```

- Although G1 and G2 which are period grades of a student and are highly correlated to the final grade G3, we drop them. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful because we want to find other factors affect the grade.

In [ ]:

```
# Find correlations with the Grade
most_correlated = stud.corr().abs()['G3'].sort_values(ascending=False)

# Maintain the top 8 most correlation features with Grade
most_correlated = most_correlated[:9]
most_correlated
```

Out[ ]:

```
G3          1.000000
failures    0.360415
Medu        0.217147
higher      0.182465
age         0.161579
Fedu        0.152457
goout       0.132791
romantic    0.129970
reason      0.121994
Name: G3, dtype: float64
```

In [ ]:

```
stud = stud.loc[:, most_correlated.index]
stud.head()
```

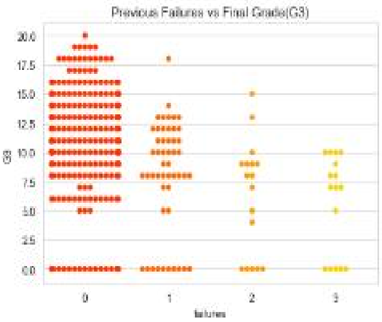
Out[ ]:

| | G3 | failures | Medu | higher | age | Fedu | goout | romantic | reason |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 6 | 0 | 4 | 1 | 18 | 4 | 4 | 0 | 0 |
| 1 | 6 | 0 | 1 | 1 | 17 | 1 | 3 | 0 | 0 |
| 2 | 10 | 3 | 1 | 1 | 15 | 1 | 2 | 0 | 2 |
| 3 | 15 | 0 | 4 | 1 | 15 | 2 | 2 | 1 | 1 |
| 4 | 10 | 0 | 3 | 1 | 16 | 3 | 2 | 0 | 1 |

### Failure Attribute

In [36]:
```
b = sns.swarmplot(x=stud['failures'],y=stud['G3'],palette='autumn')
b.axes.set_title('Previous Failures vs Final Grade(G3)')
```

Out[36]: Text(0.5, 1.0, 'Previous Failures vs Final Grade(G3)')

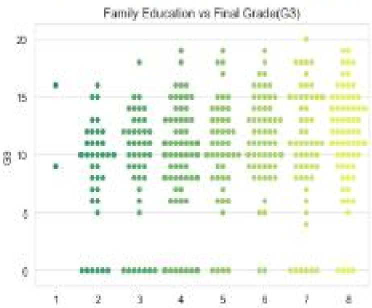



**Observation :** Student with less previous failures usually score higher

### Family Education Attribute ( Fedu + Medu )

In [37]:
```
fa_edu = stud['Fedu'] + stud['Medu']
b = sns.swarmplot(x=fa_edu,y=stud['G3'],palette='summer')
b.axes.set_title('Family Education vs Final Grade(G3)')
```

Out[37]: Text(0.5, 1.0, 'Family Education vs Final Grade(G3)')

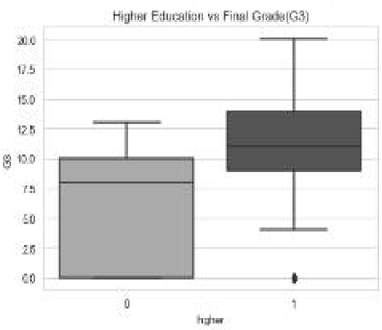



**Observation :** Educated families result in higher grades

### Wish to go for Higher Education Attribute

In [38]:
```
b = sns.boxplot(x=stud['higher'],y=stud['G3'],palette='binary')
b.axes.set_title('Higher Education vs Final Grade(G3)')
```

Out[38]: Text(0.5, 1.0, 'Higher Education vs Final Grade(G3)')

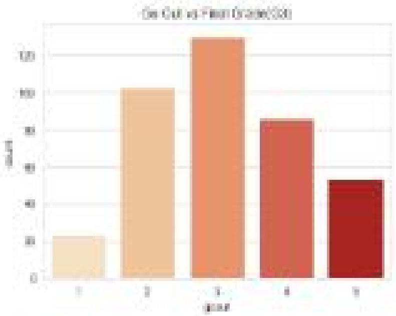



**Observation :** Students who wish to go for higher studies score more

## Going Out with Friends Attribute

```
b = sns.countplot(x=stud['goout'],palette='OrRd')
b.axes.set_title('Go Out vs Final Grade(G3)')
```

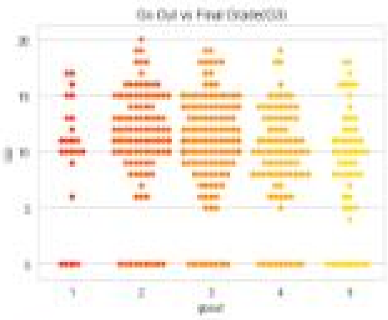
```
Text(0.5, 1.0, 'Go Out vs Final Grade(G3)')
```




**Observation :** The students have an average score when it comes to going out with friends.

```
b = sns.swarmplot(x=stud['goout'],y=stud['G3'],palette='autumn')
b.axes.set_title('Go Out vs Final Grade(G3)')
```

```
Text(0.5, 1.0, 'Go Out vs Final Grade(G3)')
```

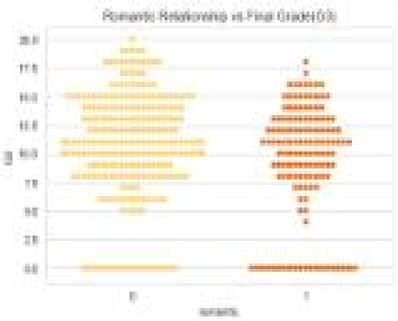



**Observation :** Students who go out a lot score less

## Romantic relationship Attribute

```
b = sns.swarmplot(x=stud['romantic'],y=stud['G3'],palette='YlOrRd')
b.axes.set_title('Romantic Relationship vs Final Grade(G3)')
```

```
Text(0.5, 1.0, 'Romantic Relationship vs Final Grade(G3)')
```



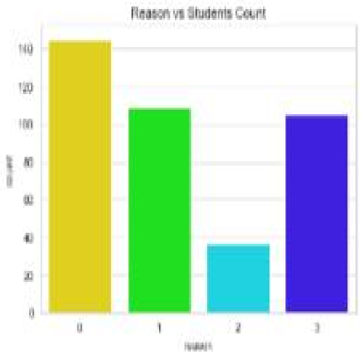

- Here romantic attribute with value 0 means no relationship and value with 1 means in relationship

**Observation :** Students with no romantic relationship score higher

## Reason Attribute

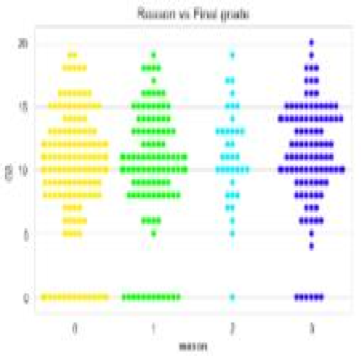
In [42]:
```
b = sns.countplot(x='reason',data=stud,palette='gist_rainbow')    # Reason to choose this school
b.axes.set_title('Reason vs Students Count')
```

Out[42]: Text(0.5, 1.0, 'Reason vs Students Count')




In [43]:
```
b = sns.swarmplot(x='reason', y='G3', data=stud,palette='gist_rainbow')
b.axes.set_title('Reason vs Final grade')
```

Out[43]: Text(0.5, 1.0, 'Reason vs Final grade')




**Observation :** The students have an equally distributed average score when it comes to reason attribute.

# Model Machine learning algorithms

## Machine Learning Algorithms

In [44]:
```
# Standard ML Models for comparison
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR

# Splitting data into training/testing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, median_absolute_error

# Distributions
import scipy
```

In [45]:
```
# splitting the data into training and testing data (75% and 25%)
# we mention the random state to achieve the same split everytime we run the code
X_train, X_test, y_train, y_test = train_test_split(stud, stud['G3'], test_size = 0.25, random_state=42)
```

In [46]:
```
X_train.head()
```

Out[46]:

| | G3 | failures | Medu | higher | age | Fedu | goout | romantic | reason |
|---|---|---|---|---|---|---|---|---|---|
| 16 | 14 | 0 | 4 | 1 | 16 | 4 | 3 | 0 | 3 |
| 66 | 12 | 0 | 4 | 1 | 15 | 4 | 3 | 1 | 3 |
| 211 | 13 | 0 | 4 | 1 | 17 | 4 | 5 | 1 | 1 |
| 7 | 6 | 0 | 4 | 1 | 17 | 4 | 4 | 0 | 1 |
| 19 | 10 | 0 | 4 | 1 | 16 | 3 | 3 | 0 | 1 |

## MAE - Mean Absolute Error & RMSE - Root Mean Square Error

In [47]:
```
# Calculate mae and rmse
def evaluate_predictions(predictions, true):
    mae = np.mean(abs(predictions - true))
    rmse = np.sqrt(np.mean((predictions - true) ** 2))

    return mae, rmse
```

In [48]:
```
# find the median
median_pred = X_train['G3'].median()

# create a list with all values as median
median_preds = [median_pred for _ in range(len(X_test))]

# store the true G3 values for passing into the function
true = X_test['G3']
```

In [49]:
```
# Display the naive baseline metrics
mb_mae, mb_rmse = evaluate_predictions(median_preds, true)
print('Median Baseline  MAE: {:.4f}'.format(mb_mae))
print('Median Baseline RMSE: {:.4f}'.format(mb_rmse))
```

```
Median Baseline  MAE: 3.7879
Median Baseline RMSE: 4.8252
```

In [50]:

```
# Evaluate several ml models by training on training set and testing on testing set
def evaluate(X_train, X_test, y_train, y_test):
    # Names of models
    model_name_list = ['Linear Regression', 'ElasticNet Regression',
                      'Random Forest', 'Extra Trees', 'SVM',
                       'Gradient Boosted', 'Baseline']
    X_train = X_train.drop('G3', axis='columns')
    X_test = X_test.drop('G3', axis='columns')

    # Instantiate the models
    model1 = LinearRegression()
    model2 = ElasticNet(alpha=1.0, l1_ratio=0.5)
    model3 = RandomForestRegressor(n_estimators=100)
    model4 = ExtraTreesRegressor(n_estimators=100)
    model5 = SVR(kernel='rbf', degree=3, C=1.0, gamma='auto')
    model6 = GradientBoostingRegressor(n_estimators=50)

    # Dataframe for results
    results = pd.DataFrame(columns=['mae', 'rmse'], index = model_name_list)

    # Train and predict with each model
    for i, model in enumerate([model1, model2, model3, model4, model5, model6]):
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)

        # Metrics
        mae = np.mean(abs(predictions - y_test))
        rmse = np.sqrt(np.mean((predictions - y_test) ** 2))

        # Insert results into the dataframe
        model_name = model_name_list[i]
        results.loc[model_name, :] = [mae, rmse]

    # Median Value Baseline Metrics
    baseline = np.median(y_train)
    baseline_mae = np.mean(abs(baseline - y_test))
    baseline_rmse = np.sqrt(np.mean((baseline - y_test) ** 2))

    results.loc['Baseline', :] = [baseline_mae, baseline_rmse]

    return results
```

In [51]:

```
results = evaluate(X_train, X_test, y_train, y_test)
results
```

Out[51]:

| | mae | rmse |
|---|---|---|
| Linear Regression | 3.48512 | 4.4326 |
| ElasticNet Regression | 3.60805 | 4.57327 |
| Random Forest | 3.72601 | 4.61621 |
| Extra Trees | 3.7797 | 4.77882 |
| SVM | 3.54927 | 4.58147 |
| Gradient Boosted | 3.57244 | 4.50059 |
| Baseline | 3.78788 | 4.82523 |

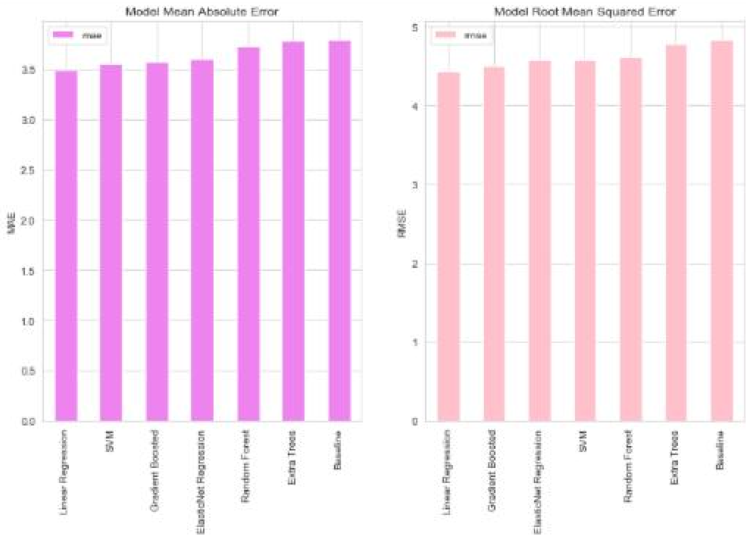
In [52]:

```
plt.figure(figsize=(12, 7))

# Root mean squared error
ax =  plt.subplot(1, 2, 1)
results.sort_values('mae', ascending = True).plot.bar(y = 'mae', color = 'violet', ax = ax)
plt.title('Model Mean Absolute Error')
plt.ylabel('MAE')

# Median absolute percentage error
ax = plt.subplot(1, 2, 2)
results.sort_values('rmse', ascending = True).plot.bar(y = 'rmse', color = 'pink', ax = ax)
plt.title('Model Root Mean Squared Error')
plt.ylabel('RMSE')

plt.show()
```



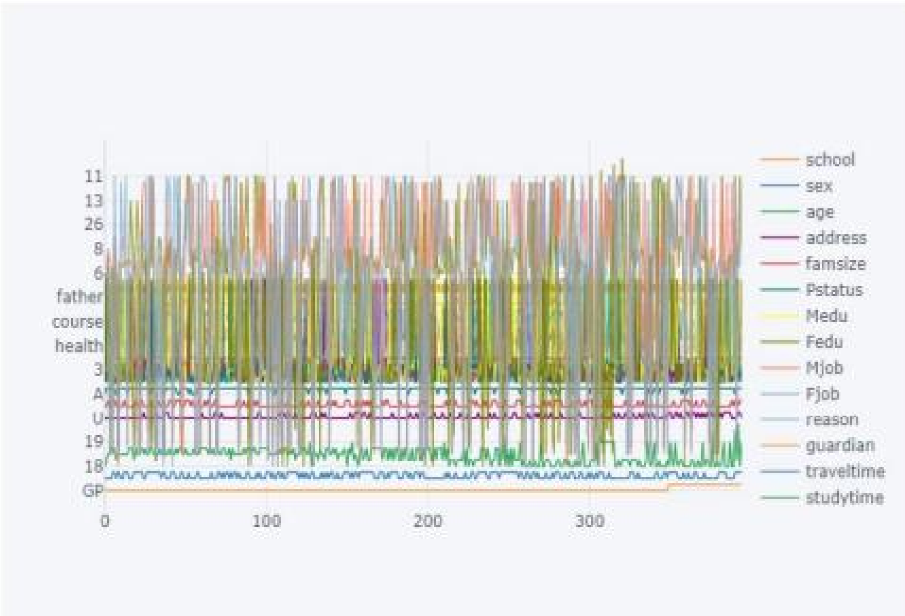

Conclusion: As we see both Model Mean Absolute Error & Model Root Mean Squared Error that the linear regression is performing the best in both cases

# CHAPTER - 5

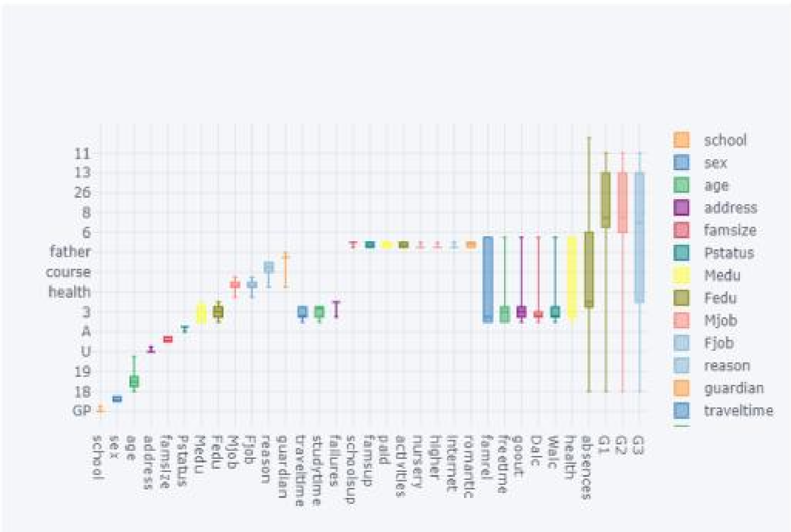
# RESULTS & ANALYSIS

## 5.1 Results

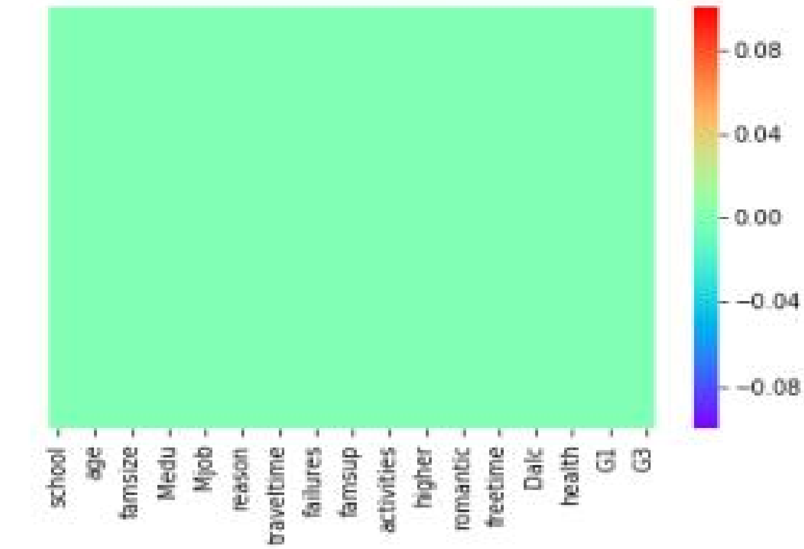
KDE Plot to view all attributes using cufflinks



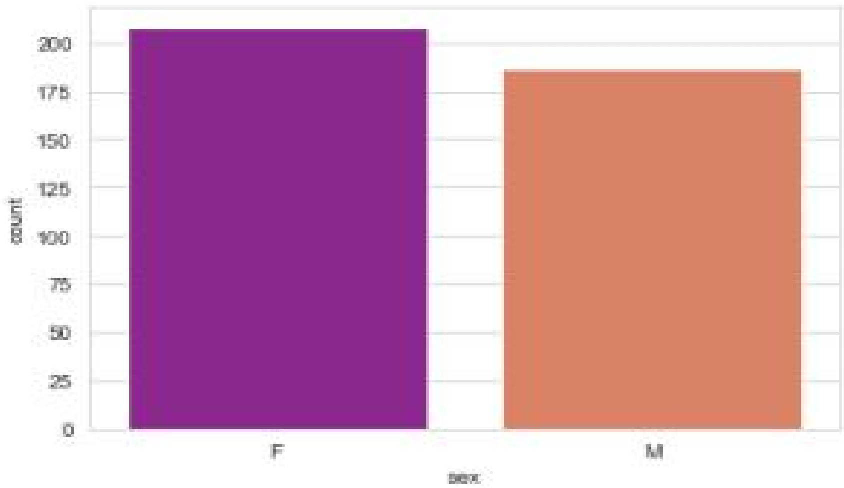

Box Plot to view all attributes using cufflinks



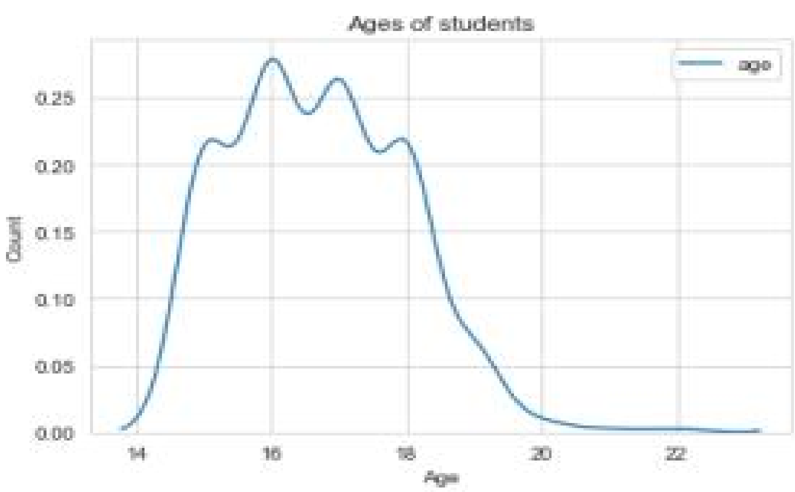
- Pictorial representation of any null data present in the dataset.




- Count Plot for Student Sex Attribute

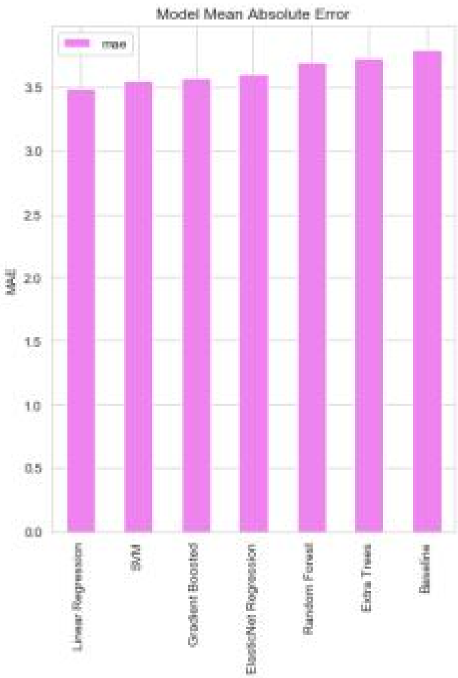


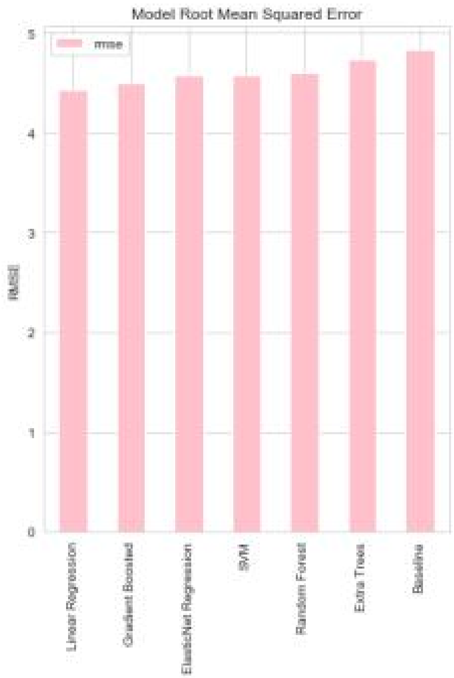

- Kernel Density Estimation for Age of Students.



## 5.2 Conclusion

As we see both MAE & Model RMSE that the Linear Regression is performing the best in both cases.




The following results have been averaged over 5 trials:

| Features Considered | G1 & G2 | G1 & School | School & Absences |
|---|---|---|---|
| Paper Accuracy | 0.919 | 0.838 | 0.706 |
| My Model Accuracy | 0.9165 | 0.8285 | 0.6847 |
| False Pass Rate | 0.096 | 0.12 | 0.544 |
| False Fail Rate | 0.074 | 0.1481 | 0.2185 |

## 5.3 Future works:

- In future, we will extend our database since as we increase the training data, the accuracy of the system will be higher. We also try to make the application as high-end and time-saving for the both school/college management and parents.
- We also plan to develop a system within the webapp in future that will predict and help students to focus on aspects where they are lacking in scoring marks ,thus it improves students academic results.

**PREFERENCES:**

https://www.kaggle.com/datasets/vipoooool/students-dataset