VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
VIT Chennai
Vandalur - Kelambakkam Road, Chennai - 600 127

Final Report

Programme: BTech

Course: CSE2004-DBMS

Slot: D2

Faculty: Prof M. Premalatha

Component: J

Title: **STUDENT MARKS PREDICTION**

Team Member(s):

DOLLY AGARWALA (20BCE1863)
NAYAN KHEMKA (20BCE1884)
HARSH CHAUHAN (20BCE1886)

Abstract

This project attempts to predict grade of students for Maths and Portuguese subjects based on data collected from two Portuguese schools using classification and prediction models. The goal is to go beyond knowing what has happened to providing a best assessment of what will happen in the future. Predictive models use known results to develop (or train) a model that can be used to predict values for different or new data. Modelling provides results in the form of predictions that represent a probability of the target variable based on estimated significance from a set of input variables.

Keywords: Data Mining, Machine Learning, Regression, Prediction, Data Visualisation

Introduction

The ultimate goal of any educational institution is offering the best educational experience and knowledge to the students. Identifying the students who need extra support and taking the appropriate actions to enhance their performance plays an important role in achieving that goal. In this research, four machine learning techniques have been used to build a classifier that can predict the performance of the students The models have been compared using the ROC index performance measure and the classification accuracy. In addition, different measures have been computed such as the classification error, precision, recall, and the F measure.

1. **Data Set Description**:

   Source: - https://archive.ics.uci.edu/ml/datasets/student+performance

   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).

   Attribute Information: # Attributes for both student-mat.csv (Math course) and student-por.csv (Portuguese language course) datasets:

   1) school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)
   2) sex - student's sex (binary: 'F' - female or 'M' - male)
   3) age - student's age (numeric: from 15 to 22)
   4) address - student's home address type (binary: 'U' - urban or 'R' - rural)
   5) famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)
   6) Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)
   7) Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
   8) Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 â€“ 5th to 9th grade, 3 â€“ secondary education or 4 â€“ higher education)
   9) Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

10) Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')
11) reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')
12) guardian - student's guardian (nominal: 'mother', 'father' or 'other')
13) traveltime - home to school travel time (numeric: 1 - 1 hour)
14) studytime - weekly study time (numeric: 1 - 10 hours)
15) failures - number of past class failures (numeric: n if 1<=n
16) schoolsup - extra educational support (binary: yes or no)
17) famsup - family educational support (binary: yes or no)
18) paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
19) activities - extra-curricular activities (binary: yes or no)
20) nursery - attended nursery school (binary: yes or no)
21) higher - wants to take higher education (binary: yes or no) 22) internet - Internet access at home (binary: yes or no)
23) romantic - with a romantic relationship (binary: yes or no)
24) famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
25) freetime - free time after school (numeric: from 1 - very low to 5 - very high)
26) goout - going out with friends (numeric: from 1 - very low to 5 - very high)
27) Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
28) Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
29) health - current health status (numeric: from 1 - very bad to 5 - very good)
30) absences - number of school absences (numeric: from 0 to 93)
31) G1 - first period grade (numeric: from 0 to 20)
32) G2 - second period grade (numeric: from 0 to 20)
33) G3 - final grade (numeric: from 0 to 20)

Data Pre-processing:

Both datasets are processed to check

1. Null Values
2. Duplicates &
3. Invalid values

but fortunately, there are no such irregularities in the dataset, implying that data is already clean and processed. Since both datasets have same set of attributes and has similar kind of data, they are merged vertically, so as to make the dataset effective and increase the dataset, this process supposedly key aspect of data pre-processing.

Modification of Dataset:

A new column called ‘FinalGrade’ is asserted which reflects ‘Grade3’ and a broader level view of ‘Grade3’. The column is inferred broadly as five categories under given conditions as follows:

- 'Excellent'[(data.G3 >= 18) & (data.G3 <= 20)] ● 'Good' [(data.G3 >= 15) & (data.G3 <= 17)]
- 'Satisfactory' [(data.G3 >= 11) & (data.G3 <= 14)]
- 'Poor' [(data.G3 >= 6) & (data.G3 <= 10)]
- 'Failure' [(data.G3 >= 0) & (data.G3 <= 5)]

New columns s_no, m_id and f_id are made to form keys.

Feature Extraction:

Exploring and observing the data through visualizations resulted in some major conclusions, which made primary contribution to extract and structure important features.

And it is stored in a new file named **‘Features’**

2. Consider the schema alone and normalize it till BCBF using schema decomposition

FEATURES DATABASE

features(s_no, school, sex, age, address, famsize, Pstatus, M_id, Medu, F_id, Fedu, Mjob, Fjob, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, G1, G2, G3, subject, FinalGrade, Regularity, Grade1, Grade2)

1NF – Identify Key, FDs

s_no -> school, sex, age, address, famsize, Pstatus, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, G1, G2, G3, subject, Regularity

M_id -> Medu, Mjob

F_id -> Fedu, Fjob

G1 -> Grade1

G2 -> Grade2

G3 -> FinalGrade

Primary Key Selected: **sno, M_id, F_id**

Features is already in 1NF

2NF – No Partial Dependency – If table has Partial Dependency decompose it.

There is a partial dependency since:

i. Medu, Mjob depend on M_id only
ii. Fedu, Fjob depend on F_id only
iii. And rest of the attributes depend only on s_no

Hence 'features' is not in 2NF and is decomposed to:

Students (**s_no** (primary key), school, sex, age, address, famsize, Pstatus, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, G1, G2, G3, subject, Regularity, Grade1, Grade2, FinalGrade)
Mothers (**M_id** (primary key), Medu, Mjob)
Fathers (**F_id** (primary key), Fedu, Fjob)
Parents (**s_no** (foreign key), **M_id** (foreign key), **F_id** (foreign key))

3NF – There should be No Transitive Dependency – No Non key to Non key dependency, if present decompose.

The tables Mothers, Fathers, Parents have no transitive dependency but Students table has partial dependency since (G1 -> Grade1, G2 -> Grade2, G3 -> FinalGrade) and G1, G2 and G3 are non-keys in the functional dependencies.
Hence students table needs to be decomposed.

Students (**s_no** (primary key), school, sex, age, address, famsize, Pstatus, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, subject, Regularity)
Grades (**Grade** (primary key), Remark)
Stud_Grades (**s_no** (foreign key), **G1** (foreign key), **G2** (foreign key), **G3** (foreign key))

Where all G1, G2 and G3 refer to the same primary key of Grades table.

So now tables after normalised to 3NF, which also are in **BCNF** as well since all pks have only 1 attribute, are:

Students (**s_no** (primary key), school, sex, age, address, famsize, Pstatus, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, subject, Regularity)
Grades (**Grade** (primary key), Remark)
Mothers (**M_id** (primary key), Medu, Mjob)
Fathers (**F_id** (primary key), Fedu, Fjob)
Parents (**s_no** (foreign key), **M_id** (foreign key), **F_id** (foreign key))
Stud_Grades (**s_no** (foreign key), **G1** (foreign key), **G2** (foreign key), **G3** (foreign key))

3. Draw the ER diagram for the final decomposed schema stating the key attributes, mapping cardinalities, participation constraint, and so on

The final decomposed schema to the BCNF form is:

Students:
s_no (primary key), school, sex, age, address, famsize, Pstatus, reason, guardian, traveltime, studytime, failures, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic, famrel, freetime, gout, Dalc, Walc, health, absences, subject, Regularity

Grades:
Grade (primary key), Remark

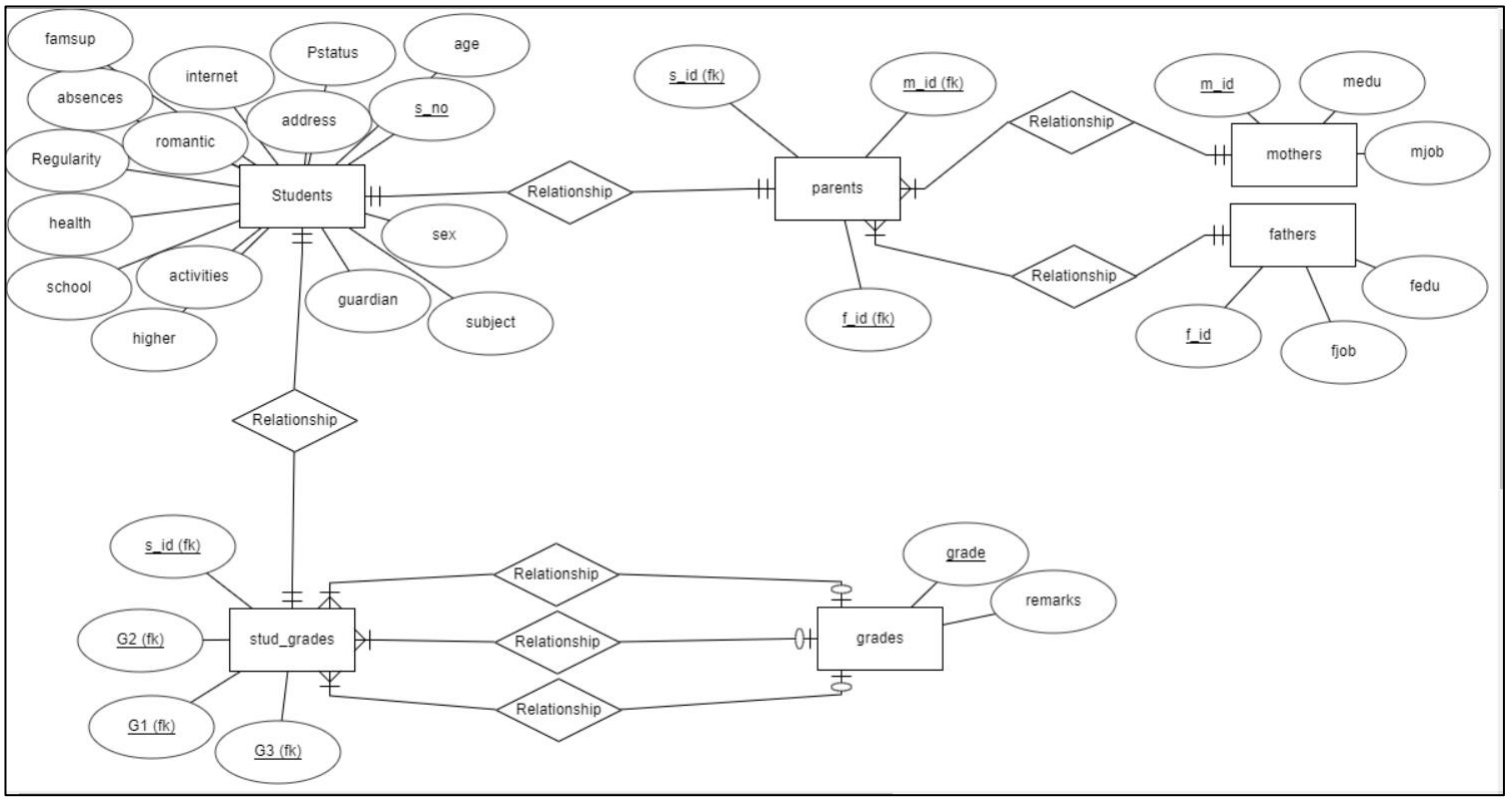
Mothers:
M_id (primary key), Medu, Mjob

Fathers:
F_id (primary key), Fedu, Fjob

Parents:
s_no (foreign key), **M_id** (foreign key), **F_id** (foreign key)

Stud_Grades:
s_no (foreign key), **G1** (foreign key), **G2** (foreign key), **G3** (foreign key)

The ERD for the above schema:



There is **one-to-one** relationship and **total** participation between '*Students*' and '*parents*'.
There is **one-to-one** relationship and **total** participation between '*mothers*' and '*parents*'.
There is **one-to-one** relationship and **total** participation between '*fathers*' and '*parents*'.

There is **one-to-one** relationship and **total** participation between '*students*' and '*stud_grades*'.
There are **three many-to-one** relationships between '*stud_grades*' and '*grades*', where the '*grades*' table have **partial** participation.

4. Methodology and Algorithm used:

This project presents main characteristics of data sets in concise form. It covers following two basic data mining techniques:
• REGRESSION: Regression analysis is a statistical method to model the relationship between a dependent (target) and independent (predictor) variables with one or more independent variables. More specifically, Regression analysis helps us to understand how the value of the dependent variable is changing corresponding to an independent variable when other independent variables are held fixed. It predicts continuous/real values such as temperature, age, salary, price, etc.

• PREDICTION: The word prediction in machine learning refers to the output of a trained model, representing the most likely value that will be obtained for a given input. Prediction in machine learning has a variety of applications, from chatbot development to recommendation systems. The model is trained with historical data, and then predicts a selected property of the data for new inputs. Prediction is used in lots of different areas, since it allows us to make highly accurate guesses about many things, such as predicting what the stock markets will do on any given day, predict results in sports, or even help the medical industry predict diseases. The algorithms for prediction are classified as supervised learning algorithms since they need a training dataset with correct examples to learn from them.

• CLASSIFICATION: As the name suggests, Classification is the task of "classifying things" into sub categories. But, by a machine. If that doesn't sound like much, imagine your computer being able to differentiate between you and a stranger. Between a potato and a tomato. Between an A grade and a F. In Machine Learning and Statistics, Classification is the problem of identifying to which of a set of categories (sub populations), a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Algorithms used: -

i. LOGISTIC REGRESSION: Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.
Sometimes logistic regressions are difficult to interpret; the Intellects Statistics tool easily allows you to conduct the analysis, then in plain English interprets the output. Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution function of logistic distribution. Thus, it treats the same set of problems as probit regression using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard logistic distribution of errors and the second a standard normal distribution of errors.

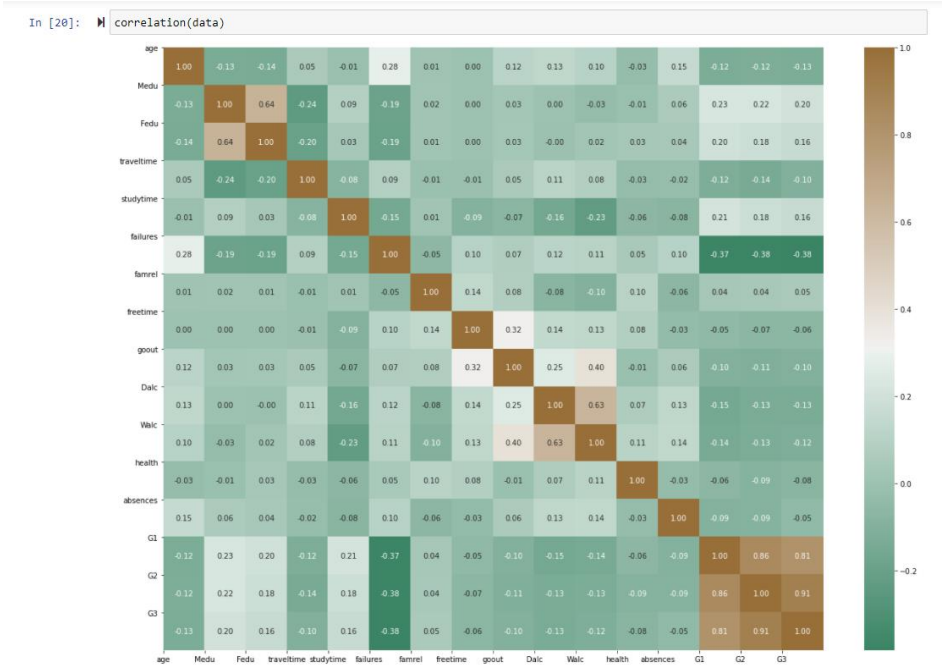
ii. XG BOOST: XGBoost is an open-source software library which provides the gradient boosting framework for C++, Java, Python, R, and Julia. It works on Linux, Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". Other than running on a single machine, it also supports the distributed processing frameworks Apache Hadoop, Apache Spark, and Apache Flink. It has gained much popularity and attention recently as it was the algorithm of choice for many winning teams of a number of machine learning competitions.

iii. SUPPORT VECTOR MACHINE: Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well

*After Data Visualisation and implementation of the given models we decide to use **XGBoost** model for our predictive analysis because it gives us the most accurate value for the Final Grade Prediction in accordance with our dataset.*

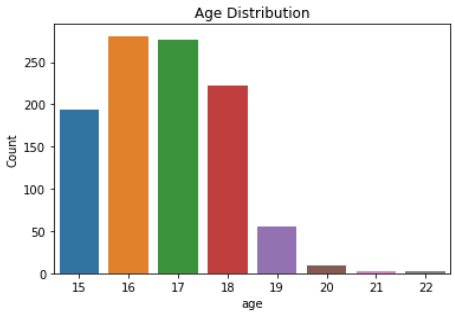
5. Implementation

First, we visualize our data set to see which features impact our prediction the most. And here are few of the results of our data visualization.



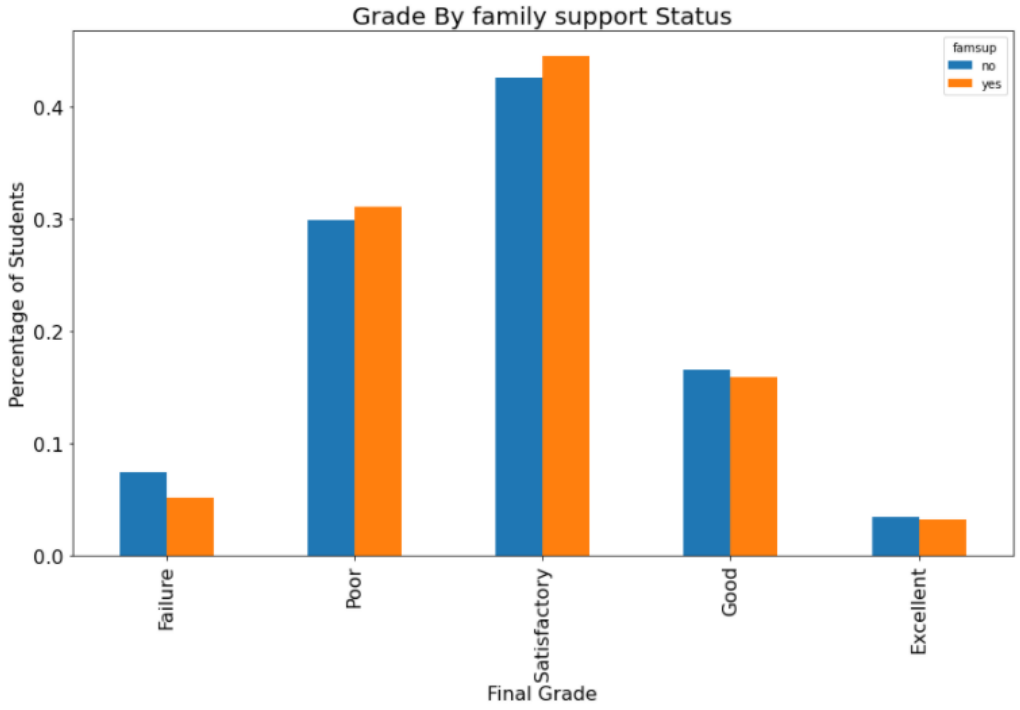

```
In [27]:
f, ax = plt.subplots()
figure = sns.countplot(x = 'age', data=data, order=[15,16,17,18,19,20,21,22])
ax = ax.set(ylabel="Count", xlabel="age")
figure.grid(False)
plt.title('Age Distribution')
#plt.savefig('age_plot.png', bbox_inches='tight')
```

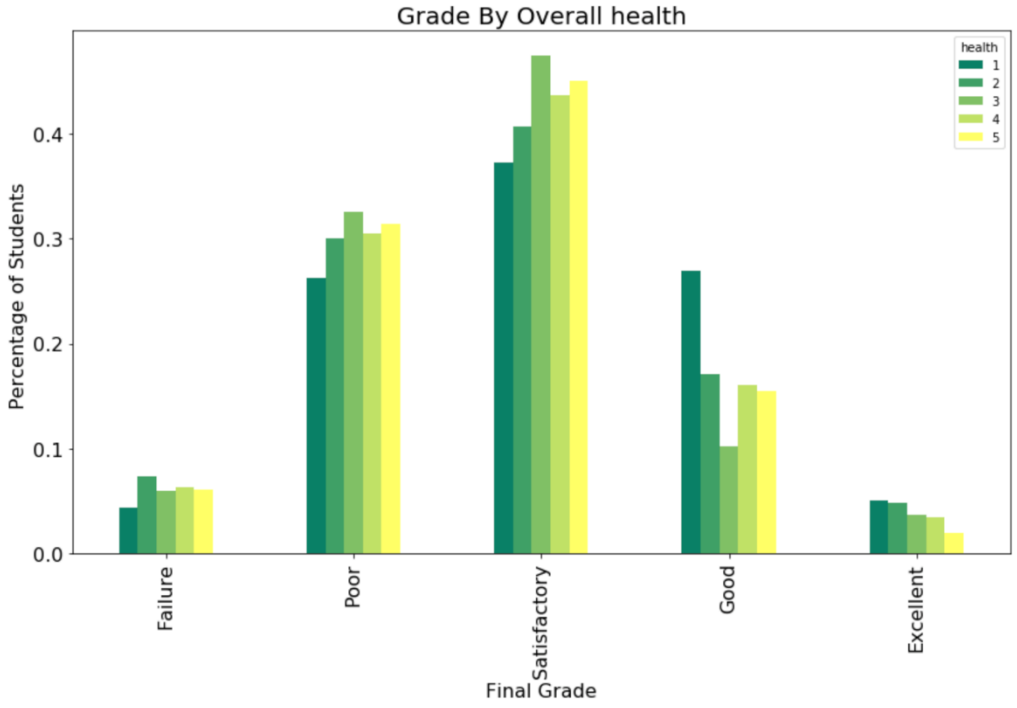
```
Out[27]: Text(0.5, 1.0, 'Age Distribution')
```

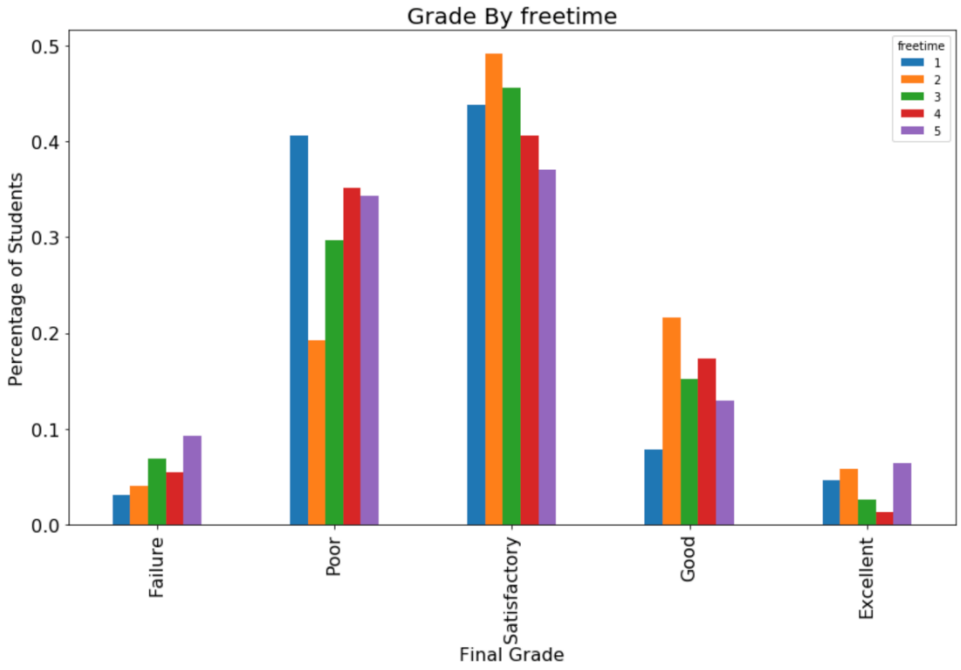


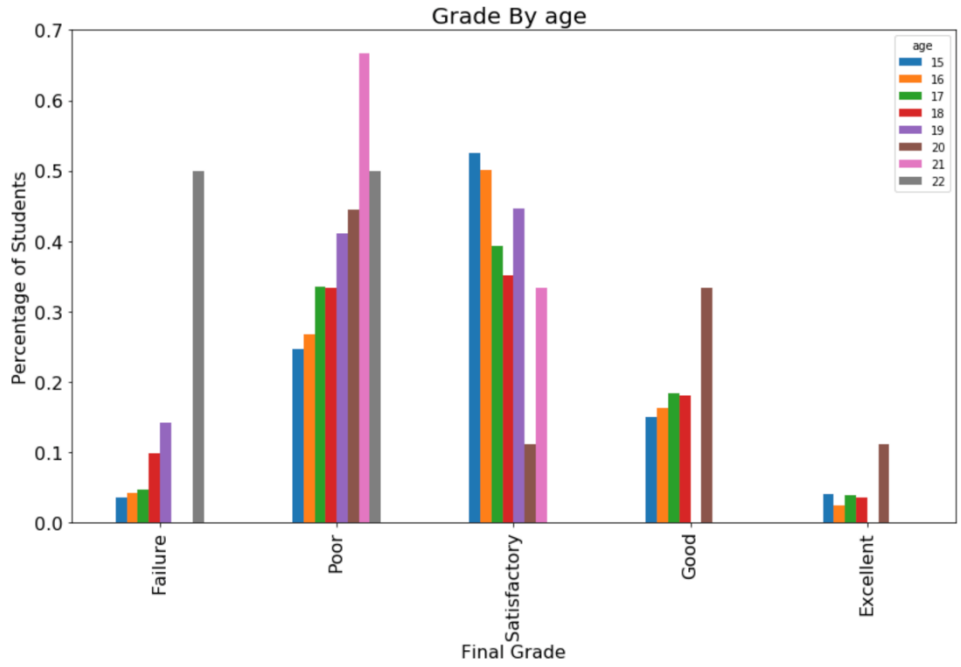
In [52]:

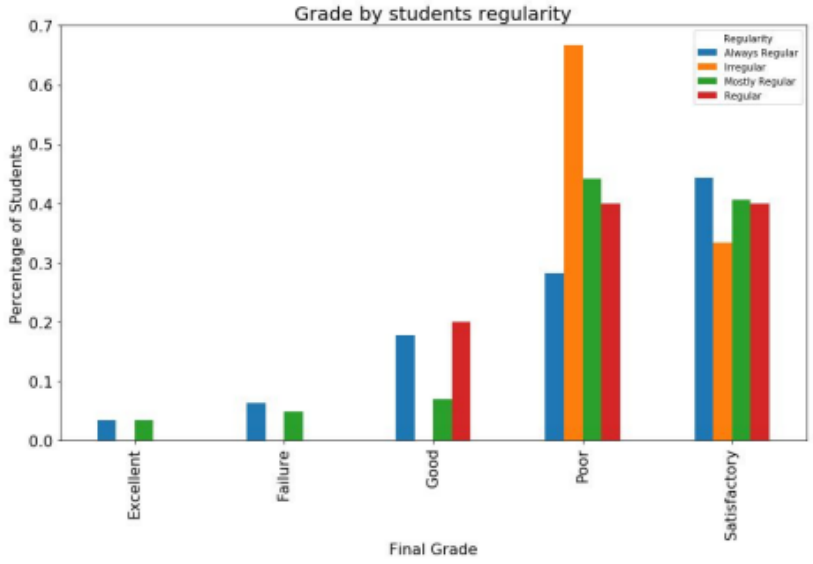
```
# relationship status
perc = (lambda col: col/col.sum())
index = ['Failure','Poor','Satisfactory','Good','Excellent']
relationship_index = pd.crosstab(index=data.FinalGrade, columns=data.famsup)
romantic_index = relationship_index.apply(perc).reindex(index)
romantic_index.plot.bar(fontsize=16, figsize=(14,8))
plt.title('Grade By family support Status', fontsize=20)
plt.ylabel('Percentage of Students', fontsize=16)
plt.xlabel('Final Grade', fontsize=16)
# plt.savefig('Grade_family support.png', bbox_inches='tight')
plt.show()
```

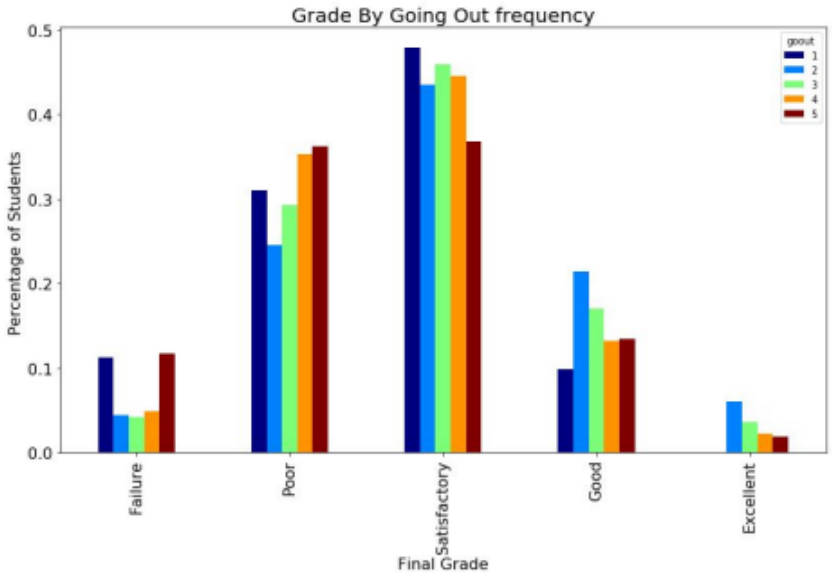


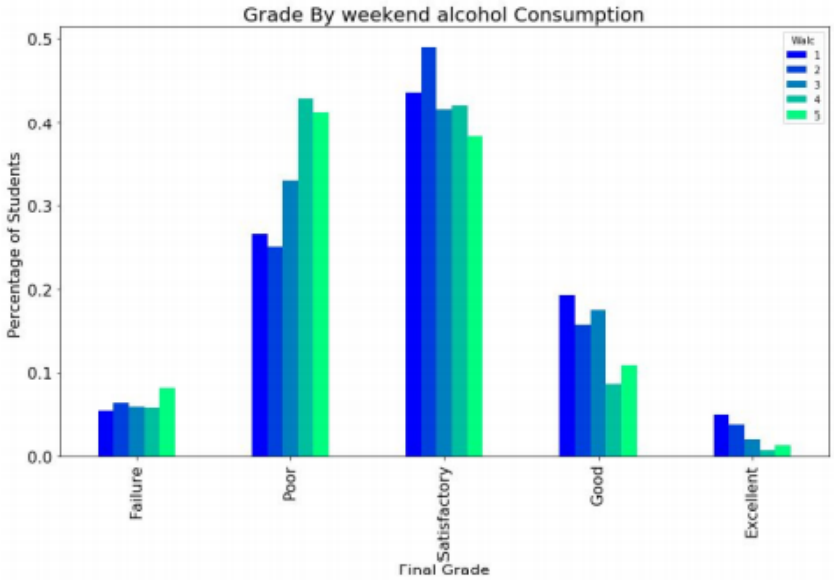



Grade By freetime
freetime
1
2
3
4
5
0.5
0.4
0.3
0.2
0.1
0.0
Percentage of Students
Failure
Poor
Satisfactory
Good
Excellent
Final Grade


Grade By age
age
15
16
17
18
19
20
21
22
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Percentage of Students
Failure
Poor
Satisfactory
Good
Excellent
Final Grade


Grade by students regularity
Regularity
Always Regular
Irregular
Mostly Regular
Regular
Percentage of Students
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Excellent
Failure
Good
Poor
Satisfactory
Final Grade


Grade By Going Out frequency
goout
1
2
3
4
5
Percentage of Students
0.0
0.1
0.2
0.3
0.4
0.5
Failure
Poor
Satisfactory
Good
Excellent
Final Grade


Grade By weekend alcohol Consumption
Walc
1
2
3
4
5
Percentage of Students
0.0
0.1
0.2
0.3
0.4
0.5
Failure
Poor
Satisfactory
Good
Excellent
Final Grade

We add few columns like regularity grade1, grade2 and final grade. And we extract it in a file called features.csv

Final Grade

In [58]:
```
#absences - number of school absences (numeric: from 0 to 93)
data.head()
data['Regularity'] = 'na'
data.loc[(data.absences >= 0) & (data.absences <= 9), 'Regularity'] = 'Always Regular'
data.loc[(data.absences >= 10) & (data.absences <= 29), 'Regularity'] = 'Mostly Regular'
data.loc[(data.absences >= 30) & (data.absences <= 49), 'Regularity'] = 'Regular'
data.loc[(data.absences >= 50) & (data.absences <= 79), 'Regularity'] = 'Irregular'
data.loc[(data.absences >= 80)& (data.absences <= 93), 'Regularity'] = 'Highly Irregular'
data.head()
```

Out[58]:

ol	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	Dalc	Walc	health	absences	G1	G2	G3	subject	FinalGrade	Regularity
GP	F	18	U	GT3	A	4	4	at_home	teacher	...	1	1	3	4	0	11	11	Portuguese	Satisfactory	Always Regular
GP	F	17	U	GT3	T	1	1	at_home	other	...	1	1	3	2	9	11	11	Portuguese	Satisfactory	Always Regular
GP	F	15	U	LE3	T	1	1	at_home	other	...	2	3	3	6	12	13	12	Portuguese	Satisfactory	Always Regular
GP	F	15	U	GT3	T	4	2	health	services	...	1	1	5	0	14	14	14	Portuguese	Satisfactory	Always Regular
GP	F	16	U	GT3	T	3	3	other	other	...	1	2	5	0	11	13	13	Portuguese	Satisfactory	Always Regular

36 columns

In [60]:
```
# 31 G1 - first period grade (numeric: from 0 to 20)
# 31 G2 - second period grade (numeric: from 0 to 20)
# 32 G3 - final grade (numeric: from 0 to 20, output target)
data['Grade1'] = 'na'
data.loc[(data.G1 >= 18) & (data.G1 <= 20), 'Grade1'] = 'Excellent'
data.loc[(data.G1 >= 15) & (data.G1 <= 17), 'Grade1'] = 'Good'
data.loc[(data.G1 >= 11) & (data.G1 <= 14), 'Grade1'] = 'Satisfactory'
data.loc[(data.G1 >= 6) & (data.G1 <= 10), 'Grade1'] = 'Poor'
data.loc[(data.G1 >= 0) & (data.G1 <= 5), 'Grade1'] = 'Failure'

data['Grade2'] = 'na'
data.loc[(data.G2 >= 18) & (data.G2 <= 20), 'Grade2'] = 'Excellent'
data.loc[(data.G2 >= 15) & (data.G2 <= 17), 'Grade2'] = 'Good'
data.loc[(data.G2 >= 11) & (data.G2 <= 14), 'Grade2'] = 'Satisfactory'
data.loc[(data.G2 >= 6) & (data.G2 <= 10), 'Grade2'] = 'Poor'
data.loc[(data.G2 >= 0) & (data.G2 <= 5), 'Grade2'] = 'Failure'

data.head(5)
```

Out[60]:

e	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	health	absences	G1	G2	G3	subject	FinalGrade	Regularity	Grade1	Grade2
3	U	GT3	A	4	4	at_home	teacher	...	3	4	0	11	11	Portuguese	Satisfactory	Always Regular	Failure	Satisfactory
7	U	GT3	T	1	1	at_home	other	...	3	2	9	11	11	Portuguese	Satisfactory	Always Regular	Poor	Satisfactory
5	U	LE3	T	1	1	at_home	other	...	3	6	12	13	12	Portuguese	Satisfactory	Always Regular	Satisfactory	Satisfactory
5	U	GT3	T	4	2	health	services	...	5	0	14	14	14	Portuguese	Satisfactory	Always Regular	Satisfactory	Satisfactory
5	U	GT3	T	3	3	other	other	...	5	0	11	13	13	Portuguese	Satisfactory	Always Regular	Satisfactory	Satisfactory

5

After extracting our dataset into features.csv we implement few models on it such as

1. Logistic Regression

Logistic Regression

In [18]:
```
def logistic_regression_model(x_train,y_train,x_val,y_val):
    lr =  LogisticRegression()
    lr.fit(x_train,y_train)
    y_pred = lr.predict(x_val)
    y_predict = lr.predict_proba(x_val)
    print("Log_Loss: ",log_loss(y_val,y_predict))
    print("Accuracy_Score: ",accuracy_score(y_val,y_pred))
    confusionmatrix(y_val,y_pred)
    Fscore(y_val,y_pred)
    recall(y_val,y_pred)
    report(y_val,y_pred)
    fbeta(y_val,y_pred)
    return lr
```

2. SVM model

SVM model

In [24]:
```
def SVM_Model(X_train,Y_train,X_test,y_val):
    svc = SVC()
    svc.fit(X_train, Y_train)
    Y_pred = svc.predict(X_test)
    print("SVM Train data Score" , ":" , svc.score(X_train, y_train)
          , "," ,"Validation data Score" ,":" , svc.score(X_test, y_val))
    confusionmatrix(y_val,Y_pred)
    Fscore(y_val,Y_pred)
    recall(y_val,Y_pred)
    report(y_val,Y_pred)
    fbeta(y_val,Y_pred)
    return svc
```

3. XGBoost Model

XGBoost Model

In [33]:
```
def XGBoost(x_train,y_train,x_val,y_val):
    model = XGBClassifier()
    model = XGBClassifier(learning_rate=0.1,n_estimators=80,eval_metric='mlogloss')
    mf = model.fit(x_train,y_train)
    y_pred=model.predict(x_val)
    y_predict = mf.predict_proba(x_val)
    print("XGBoost Train data Score" , ":" , mf.score(x_train, y_train)
          , "," ,"Validation data Score" ,":" , mf.score(x_val, y_val))
    print("Log_Loss: ",log_loss(y_val,y_predict))
    confusionmatrix(y_val,y_pred)
    Fscore(y_val,y_pred)
    recall(y_val,y_pred)
    report(y_val,y_pred)
    fbeta(y_val,y_pred)


    # plot feature importance
    fig, ax = plt.subplots(figsize=(10, 20))
    plot_importance(model, ax=ax)
#     plt.savefig('Feature_Engineering.png', bbox_inches='tight')
    plt.show()
    return model
```

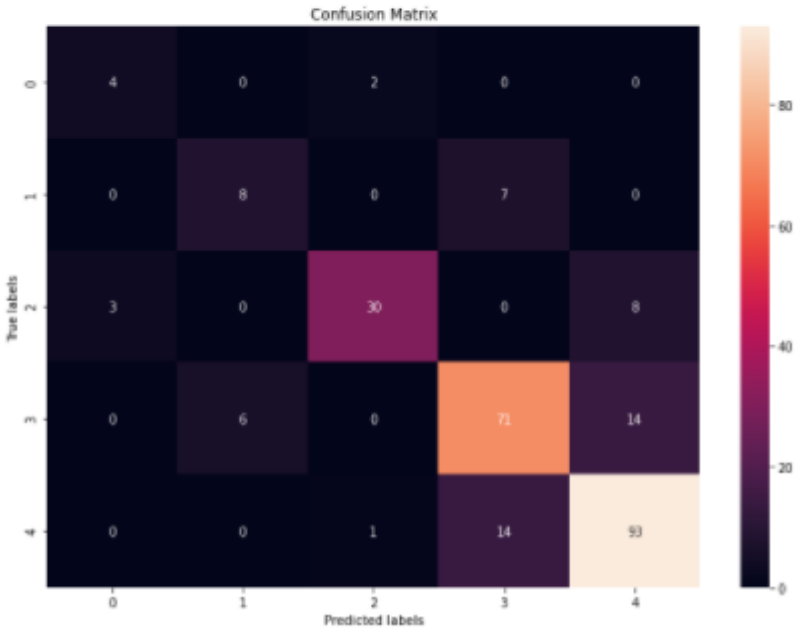
6. Results and Discussion

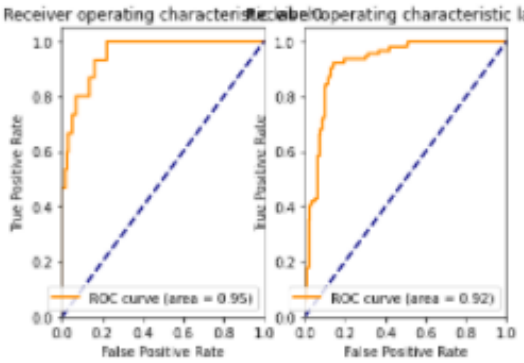
Here are all the results of the model we applied one by one to predict the final grade. The data is trained and tested with all three algorithms.

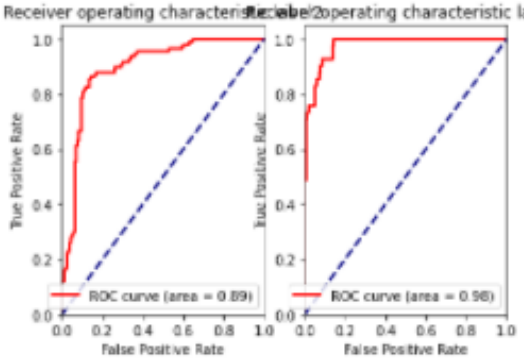
We have tried to find out accuracy score, accuracy percentage, fscore, classification report and ROC plot of each of these models to find out which model is best suited for our given data set to predict the final grade

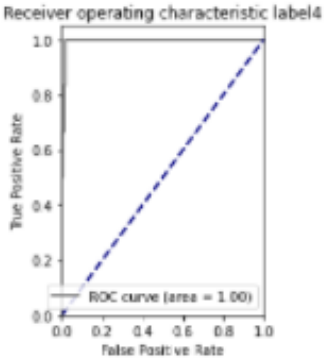
1. LOGISTICAL REGRESSION

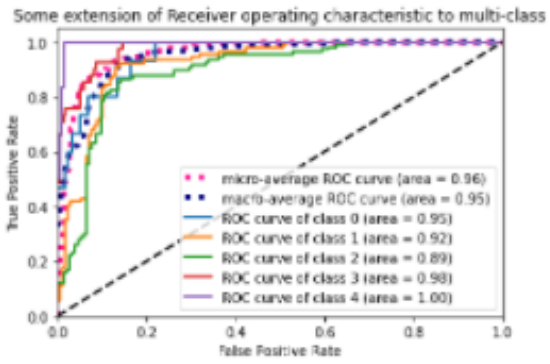
```
In [19]: model =logistic_regression_model(x_train,y_train,x_val,y_val)
         ROC_plot(x_train,x_val,model)

Log_Loss:  0.5050073793516586
Accuracy_Score:  0.789272030651341
percentage of sensitivity = 71.46076416808124
percentage of precision = 72.64765669113496
Accuracy percentage = 91.57088122605363
f score = 0.7175913131536511
percentage of recall score = 0.7146076416808124
Classification Report
              precision    recall  f1-score   support

     Failure       0.57      0.67      0.62         6
        Poor       0.57      0.53      0.55        15
Satisfactory       0.91      0.73      0.81        41
        Good       0.77      0.78      0.78        91
   Excellent       0.81      0.86      0.83       108

    accuracy                           0.79       261
   macro avg       0.73      0.71      0.72       261
weighted avg       0.79      0.79      0.79       261

Fbeta score = 0.7221500100125986
Area Under the Curve with label 0 is 0.9528455284552846
Area Under the Curve with label 1 is 0.9219133807369102
Area Under the Curve with label 2 is 0.8937908496732027
Area Under the Curve with label 3 is 0.9768292682926829
Area Under the Curve with label 4 is 0.9954248366013072
```




Receiver operating characteristic label0
Receiver operating characteristic label1
True Positive Rate
False Positive Rate
ROC curve (area = 0.95)
ROC curve (area = 0.92)


Receiver operating characteristic label2
Receiver operating characteristic label3
True Positive Rate
False Positive Rate
ROC curve (area = 0.89)
ROC curve (area = 0.98)

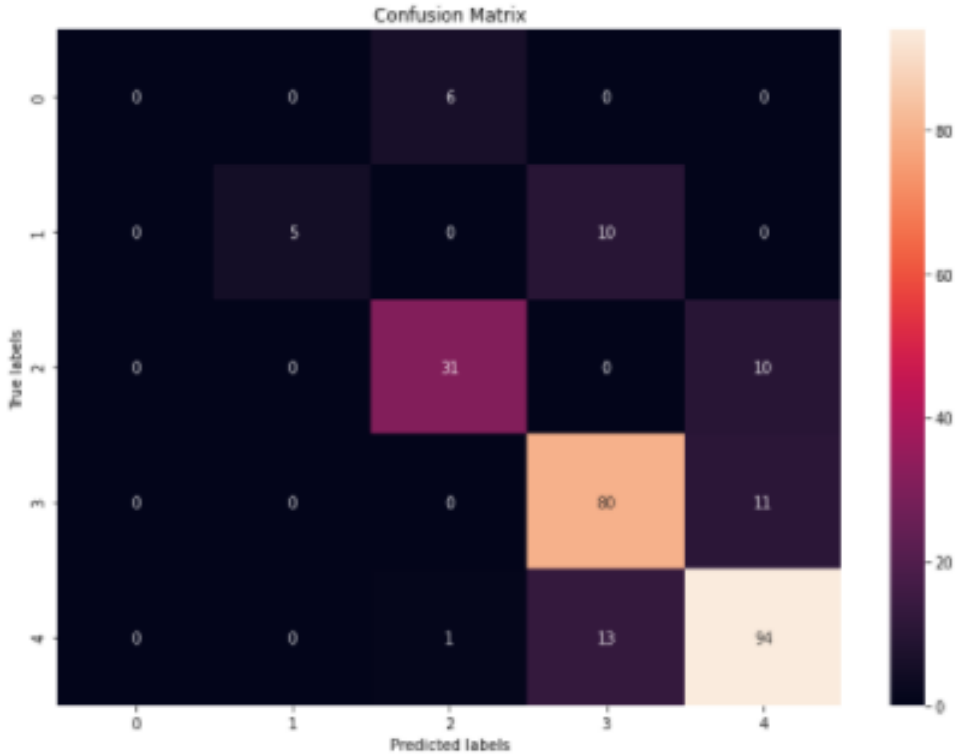

Receiver operating characteristic label4
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)


Some extension of Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
micro-average ROC curve (area = 0.96)
macro-average ROC curve (area = 0.95)
ROC curve of class 0 (area = 0.95)
ROC curve of class 1 (area = 0.92)
ROC curve of class 2 (area = 0.89)
ROC curve of class 3 (area = 0.98)
ROC curve of class 4 (area = 1.00)

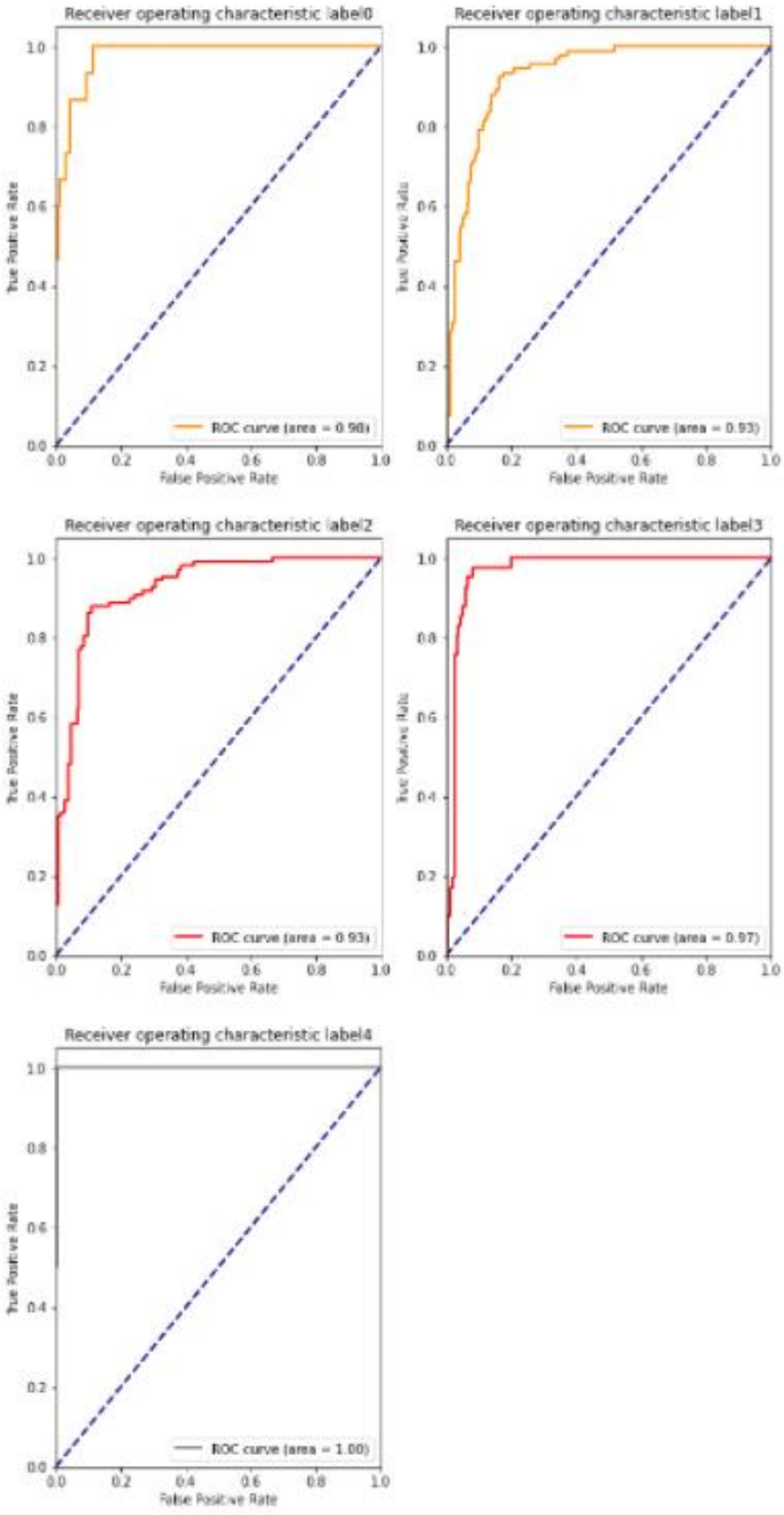
2. SVM MODEL

In [25]:
```
model = SVM_Model(x_train,y_train,x_val,y_val)
ROC_plot(x_train,x_val,model)
```

```
SVM Train data Score : 0.8122605363984674 , Validation data Score : 0.8045977011494253
percentage of sensitivity = 56.778442876003844
percentage of precision = nan
Accuracy percentage = 92.18390804597702
f score = 0.5905203443955551
percentage of recall score = 0.5677844287600384
Classification Report
              precision    recall  f1-score   support

     Failure       0.00      0.00      0.00         6
        Poor       1.00      0.33      0.50        15
Satisfactory       0.82      0.76      0.78        41
        Good       0.78      0.88      0.82        91
   Excellent       0.82      0.87      0.84       108

    accuracy                           0.80       261
   macro avg       0.68      0.57      0.59       261
weighted avg       0.79      0.80      0.79       261

Fbeta score = 0.6280175879077761
Area Under the Curve with label 0 is 0.9766937669376694
Area Under the Curve with label 1 is 0.9281835811247576
Area Under the Curve with label 2 is 0.9253207455821835
Area Under the Curve with label 3 is 0.9667405764966741
Area Under the Curve with label 4 is 0.9980392156862745
```



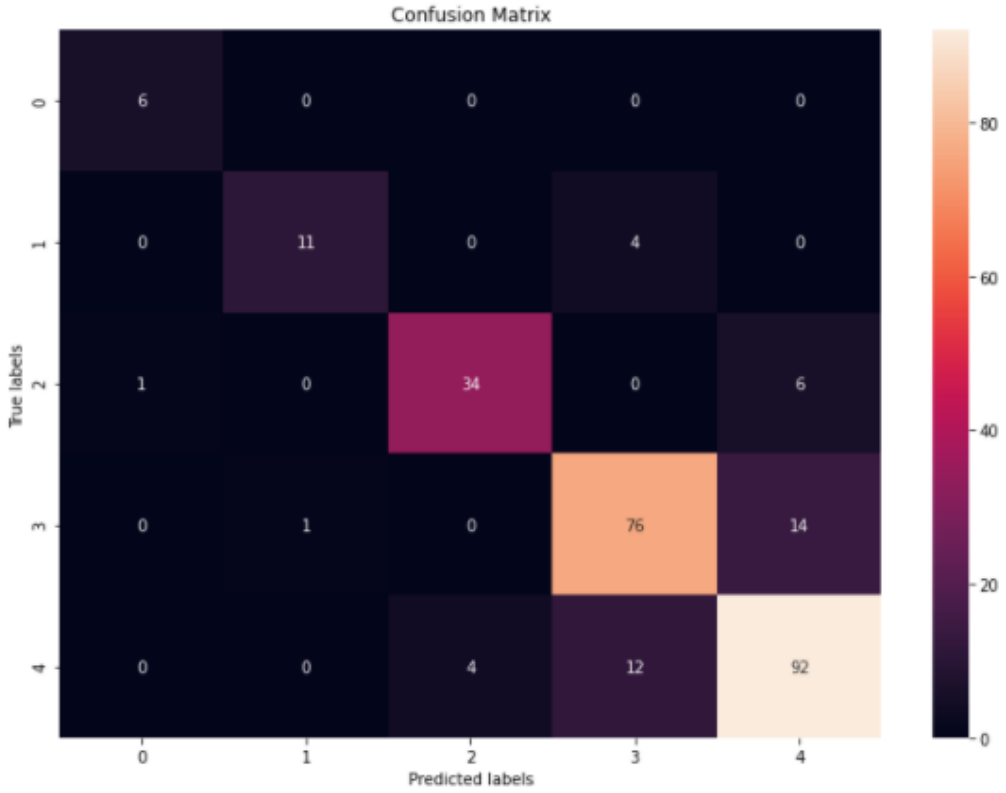
Predicted labels

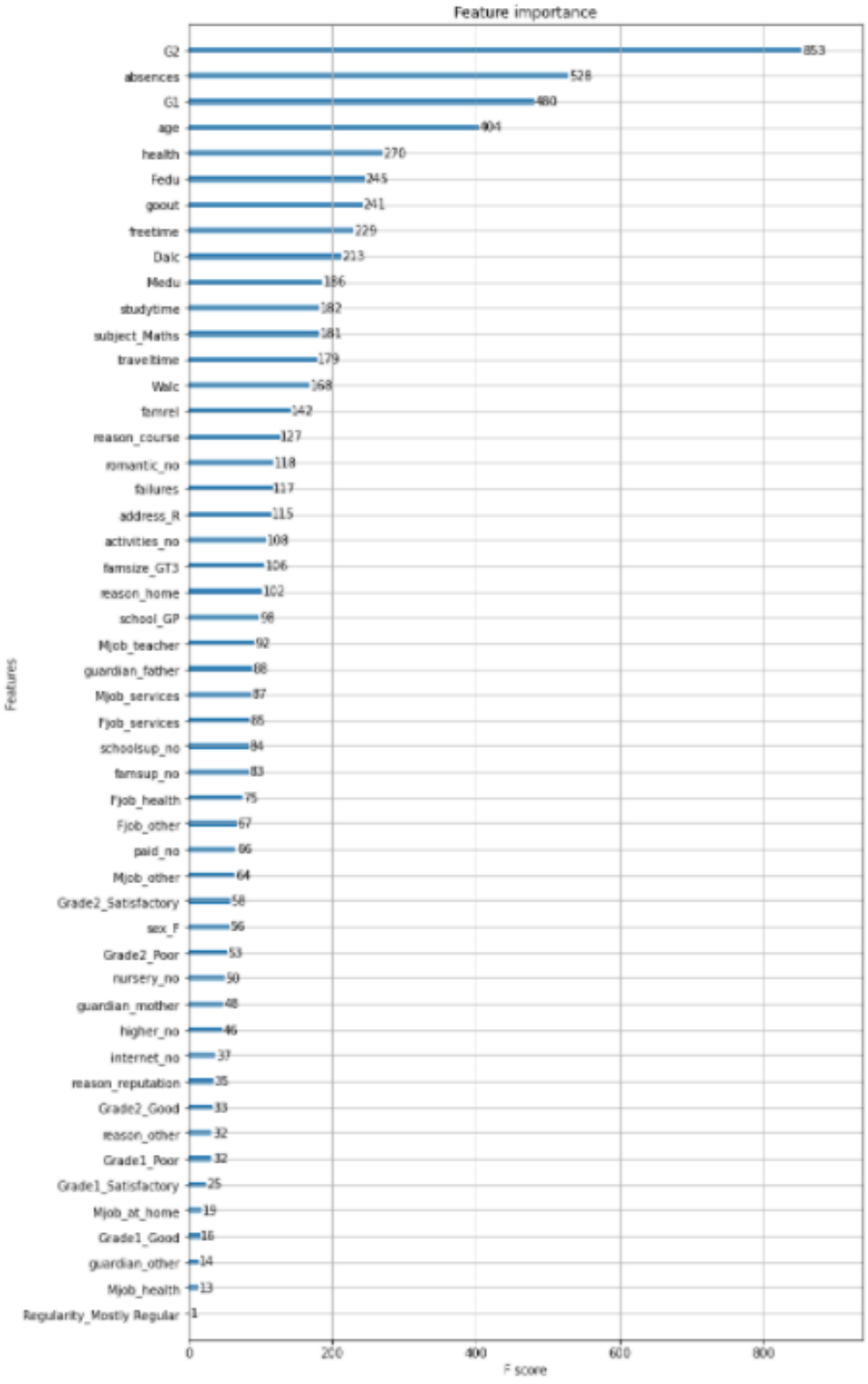


3. XGBOOST

In [34]:
```
model = XGBoost(x_train,y_train,x_val,y_val)
# ROC_plot(x_train,x_val,model)
```

```
XGBoost Train data Score : 1.0 , Validation data Score : 0.8390804597701149
Log_Loss:  0.44798505828908547
percentage of sensitivity = 84.99236626065894
percentage of precision = 86.32123787730195
Accuracy percentage = 93.5632183908046
f score = 0.8531231921644871
percentage of recall score = 0.8499236626065894
Classification Report
              precision    recall  f1-score   support

     Failure       0.86      1.00      0.92         6
        Poor       0.92      0.73      0.81        15
Satisfactory       0.89      0.83      0.86        41
        Good       0.83      0.84      0.83        91
   Excellent       0.82      0.85      0.84       108

    accuracy                           0.84       261
   macro avg       0.86      0.85      0.85       261
weighted avg       0.84      0.84      0.84       261

Fbeta score = 0.8582845338944878
```




Feature importance
G2
absences
G1
age
health
Fedu
goout
freetime
Dalc
Medu
studytime
subject_Maths
traveltime
Walc
famrel
reason_course
romantic_no
failures
address_R
activities_no
famsize_GT3
reason_home
school_GP
Mjob_teacher
guardian_father
Mjob_services
Fjob_services
schoolsup_no
famsup_no
Fjob_health
Fjob_other
paid_no
Mjob_other
Grade2_Satisfactory
sex_F
Grade2_Poor
nursery_no
guardian_mother
higher_no
internet_no
reason_reputation
Grade2_Good
reason_other
Grade1_Poor
Grade1_Satisfactory
Mjob_at_home
Grade1_Good
guardian_other
Mjob_health
Regularity_Mostly Regular
853
528
480
404
270
245
241
229
213
186
182
181
179
168
142
127
118
117
115
108
106
102
98
92
88
87
85
84
83
75
67
66
64
58
56
53
50
48
46
37
35
33
32
32
25
19
16
14
13
1
Features
0
200
400
600
800
F score

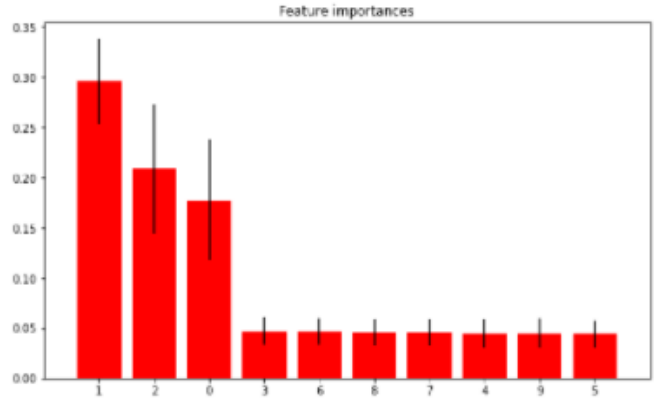
We build a classification task using 3 informative features. Then we build a forest and compute the feature importances and then we plot it.

Features Importance

In [38]:
```
def FeatureImportance():
    # Build a classification task using 3 informative features
    X, y = make_classification(n_samples=1000,
                               n_features=10,
                               n_informative=3,
                               n_redundant=0,
                               n_repeated=0,
                               n_classes=2,
                               random_state=0,
                               shuffle=False)

    # Build a forest and compute the feature importances
    forest = ExtraTreesClassifier(n_estimators=250,random_state=0)

    forest.fit(X, y)
    importances = forest.feature_importances_
    std = np.std([tree.feature_importances_ for tree in forest.estimators_],
                 axis=0)
    indices = np.argsort(importances)[::-1]

    # Print the feature ranking
    print("Feature ranking:")

    for f in range(X.shape[1]):
        print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))

    # Plot the feature importances of the forest
    plt.figure()
    plt.title("Feature importances")
    plt.bar(range(X.shape[1]), importances[indices],
           color="r", yerr=std[indices], align="center")
    plt.xticks(range(X.shape[1]), indices)
    plt.xlim([-1, X.shape[1]])
    plt.show()
```

In [39]:
```
FeatureImportance()
```

```
Feature ranking:
1. feature 1 (0.295902)
2. feature 2 (0.208351)
3. feature 0 (0.177632)
4. feature 3 (0.047121)
5. feature 6 (0.046303)
6. feature 8 (0.046013)
7. feature 7 (0.045575)
8. feature 4 (0.044614)
9. feature 9 (0.044577)
10. feature 5 (0.043912)
```



Finally we choose XGBoost Model

#

Train data Score : 0.9386973180076629

#

Validation data Score : 0.8850574712643678

After analysing our results from above we can see that xgboost has a train data score of 0.9387 and validation score of 0.8851. So we choose xgboost model for our prediction of final grade.

7. **Conclusion**

   All these results can be taken into consideration and even more reliable and more accurate algorithms can be used. Then the project will be more powerful to depend upon and even more efficient to depend upon. Here is the conclusion that we draw from our project for the dataset and the final model that we used:

Analysis of models: -

This table represents accuracies of different models when predicted on test data Model

Model	FE1	FE2	FE3	FE4	FE5
LR	0.8103	0.8120	0.8161	0.8165	0.8182
RF	0.82	0.83	0.82	0.81	0.81
SVM	0.841	0.843	0.842	0.841	0.844
DT	0.801	0.801	0.803	0.802	0.83
ADA	0.791	0.7912	0.792	0.7911	0.7934
XG	0.86	0.87	0.87	0.87	0.88

From the result, it is clear that XGBoost has maximum accuracy. Hence it is the best suited classifier to predict the final grade for the given dataset.