THE UNIVERSITY OF HUDDERSFIELD

School of Computing and Engineering

CMI3507/CMI7507 Assignment II

TITLE OF THE ASSIGNMENT- Prediction of students' performance using random forest regression

Name: Lisat Kalathoor Mathew
Student ID: U2263800
Date: December 21, 2022

Abstract

The study and evaluation of student performance and maintaining the level of education have become major issues in all educational institutions in recent years. The purpose of this study was to identify the elements that influence students' academic performance during their years of study and contribute to their decision to drop out. This work suggests an improved random forest classifier technique. This method seeks to increase classification and prediction accuracy.

1. Introduction

Through the creation of a few computer-automated "learning" algorithms, machine learning examines the already-existing data to uncover any hidden laws, which are then used to forecast and evaluate future data. The traditional mainframe has found it challenging to keep up with the demands of the industrial sector due to the rapid rise of the mobile Internet era, the generation of large amounts of data, and improvements in the industry for assessing speed and cost requirements. The technology of distributed computing is created as a result. By breaking down a work that requires a lot of computing power into smaller tasks that can be processed by multiple computer nodes, distributed computing technologies can finally summarise the calculation results to get the results (Ren et al., 2017).

Random forests are a supervised learning algorithm. It can be used for both classification and regression. It is also the most versatile and user-friendly algorithm. Random forests create decision trees on randomly selected data samples, get forecasts from each tree, then vote for the best alternative. To provide accurate models, a ranking of the importance of perceptive variables, and sharp coverage on a record-by-record basis for deep data understanding, Random Forests is a bagging tool. It does this by combining the power of multiple diverse analyses, organisational strategies, and ensemble learning. Its advantages include the ability to spot outliers and anomalies in knowledge-based data, show nearby clusters, forecast future events, make necessary predictions in a characteristic way, find data patterns, swap missing values with imputations, and provide perceptive graphics (Mythili & Shanavas, 2014).

2. Questions

2.1 Question 1

The attached dataset of student achievement is shown in Figure 1. There are 34 columns and 1044 rows in this dataset. Here, I double-checked the column names and form. Then I looked at the descriptive statistics, where details about the dataset, such as the datatypes of each column—some are objects, while the others are integers—were provided, as well as the describe function, which gave information such as the mean, standard deviation, count, maximum value, etc. Additionally, I looked at the missing values in the data and replaced two columns with null values using the fillna () function.

I then deleted those columns that were not necessary for the analysis and verified that each column had a unique value. I then used the replace () function to convert the object to numbers.

```
student_data=student_mat.append(student_por)
student_data
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	fatherd
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3	4	1	1	3	6	5	6	6	NaN
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	3	1	1	3	4	5	5	6	NaN
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3	2	2	3	3	10	7	8	10	NaN
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	2	1	1	5	2	15	14	15	NaN
4	GP	F	16	U	GT3	T	3	3	other	other	...	3	2	1	2	5	4	6	10	10	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	MS	F	19	R	GT3	T	2	3	services	other	...	4	2	1	2	5	4	10	11	10	no
645	MS	F	18	U	LE3	T	3	1	teacher	services	...	3	4	1	1	1	4	15	15	16	no
646	MS	F	18	U	GT3	T	1	1	other	other	...	1	1	1	1	5	6	11	12	9	no
647	MS	M	17	U	LE3	T	3	1	services	services	...	4	5	3	4	2	6	10	10	10	no
648	MS	M	18	R	LE3	T	3	2	services	other	...	4	1	3	4	5	4	10	11	11	no

1044 rows × 34 columns

Fig.1 Appended dataset

```
student_data.shape
```

```
(1044, 34)
```

```
student_data.columns
```

```
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', 'fatherd'],
      dtype='object')
```

Fig.2 shape and column names

```
student_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1044 entries, 0 to 648
Data columns (total 34 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      1044 non-null   object
 1   sex         1044 non-null   object
 2   age         1044 non-null   int64
 3   address     1044 non-null   object
 4   famsize     1044 non-null   object
 5   Pstatus     1044 non-null   object
 6   Medu        1044 non-null   int64
 7   Fedu        1044 non-null   int64
 8   Mjob        1044 non-null   object
 9   Fjob        1044 non-null   object
 10  reason      1044 non-null   object
 11  guardian    1044 non-null   object
 12  traveltime  1044 non-null   int64
 13  studytime   1044 non-null   int64
 14  failures    1044 non-null   int64
 15  schoolsup   1044 non-null   object
 16  famsup      1044 non-null   object
 17  paid        395 non-null    object
 18  activities  1044 non-null   object
 19  nursery     1044 non-null   object
 20  higher      1044 non-null   object
 21  internet    1044 non-null   object
 22  romantic    1044 non-null   object
 23  famrel      1044 non-null   int64
 24  freetime    1044 non-null   int64
 25  goout       1044 non-null   int64
```

Fig.3 information

```
student_data.describe()
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	
count	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.000000	1044.0
mean	16.726054	2.603448	2.387931	1.522989	1.970307	0.264368	3.935824	3.201149	3.156130	1.494253	2.284483	3.5
std	1.239975	1.124907	1.099938	0.731727	0.834353	0.656142	0.933401	1.031507	1.152575	0.911714	1.285105	1.4
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	3.0
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.0
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.0
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.0

Fig.4 describe function

```
student_data.isna().sum()
```

```
school          0
sex             0
age             0
address         0
famsize         0
Pstatus         0
Medu            0
Fedu            0
Mjob            0
Fjob            0
reason          0
guardian        0
traveltime      0
studytime       0
failures        0
schoolsup       0
famsup          0
paid          649
activities      0
nursery         0
higher          0
internet        0
romantic        0
famrel          0
freetime        0
goout           0
Dalc            0
Walc            0
health          0
absences        0
G1              0
G2              0
G3              0
fatherd       395
dtype: int64
```

Fig.5 missing values

I then create a few graphs. In Fig. 6, a stacked bar plot shows that most urban children are assigned to a MS, whereas the proportion of rural kids attending both institutions is about equal. I also investigated sex and school and found that both urban and rural areas had females attending GP schools.

```
sa=pd.crosstab(student_data1['address'],student_data1['school'])
sa
```

school	0	1
address		
0	141	144
1	631	128

```
sa.plot(kind='bar',stacked=True)
plt.xlabel('address')
plt.ylabel('school')
plt.show()
```

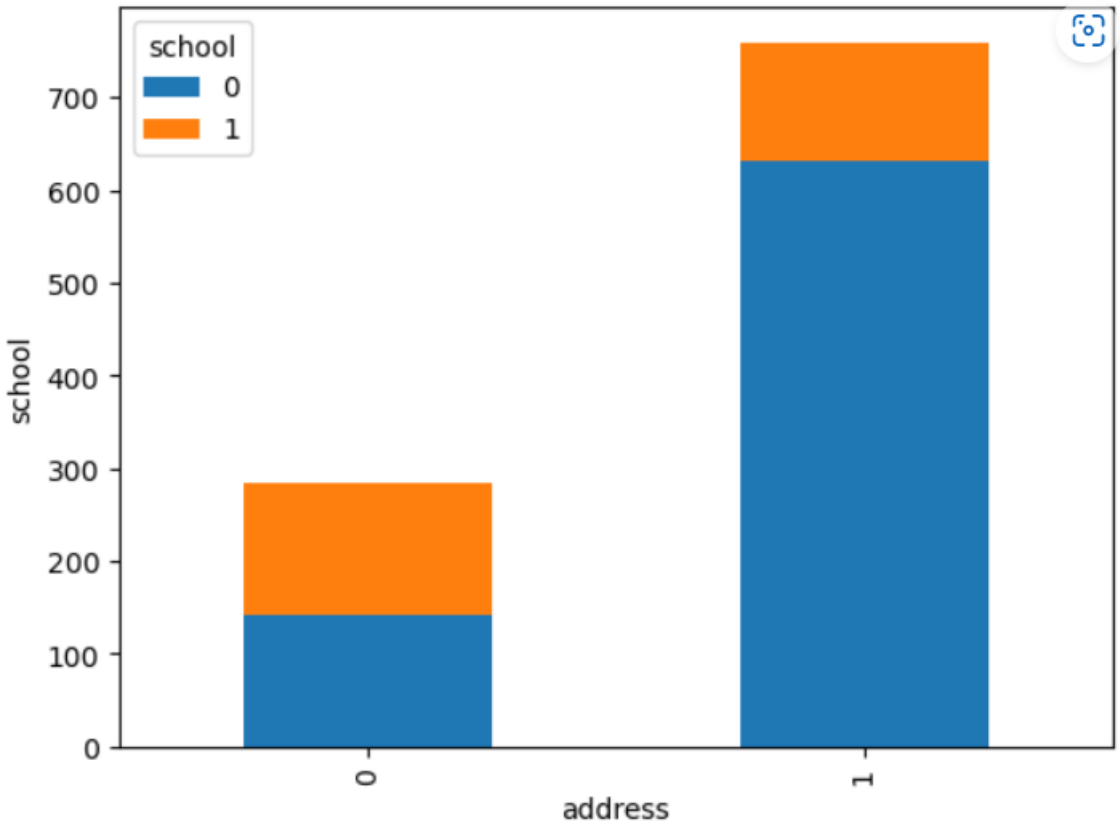


Fig.6 Schools vs Address

In fig.7 I used a grouped bar plot where I checked the family size with address and school. Here family size greater than 3 leaves in Urban area.

```
student_data1.groupby('famsize')[['address','school']].sum().plot.bar(color=['blue','red'],figsize=(8,5))
plt.ylabel('ADDRESS AND SCHOOL')
plt.show()
```

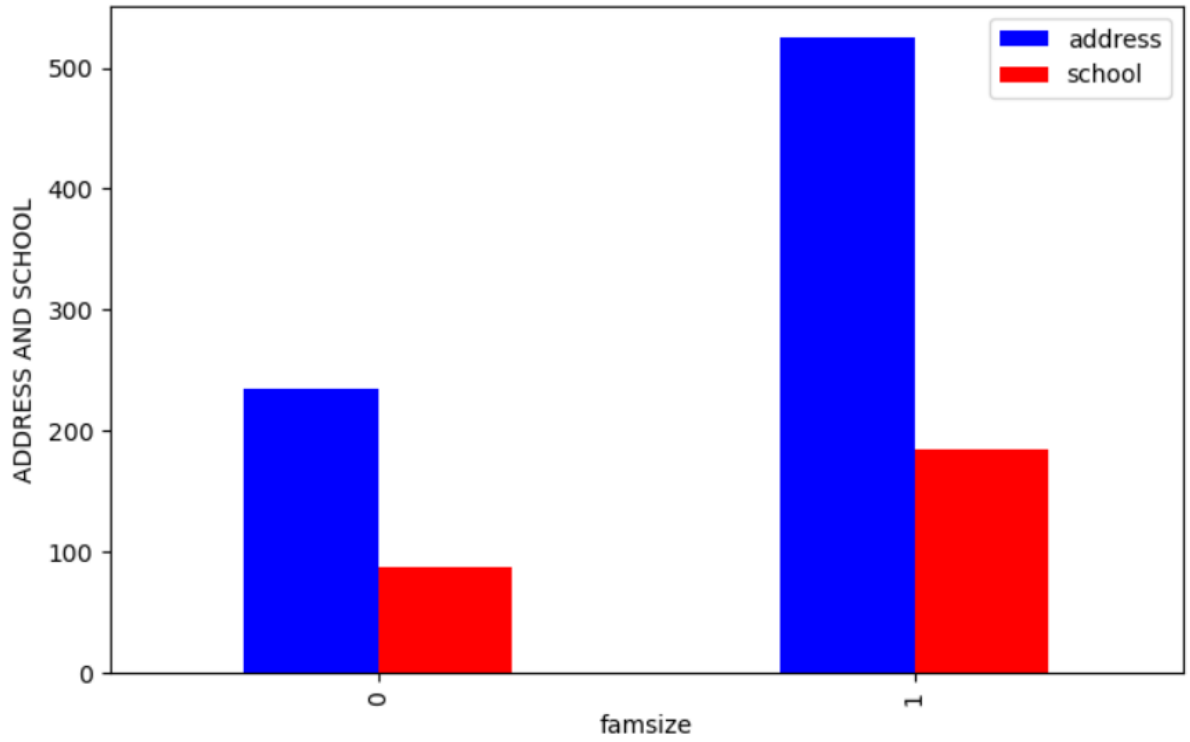


Fig.7 Family size with address and school

Similarly, I checked with internet with sex and address fig.8, where we can see both female and male people leaving in Urban area uses more internet.

```
student_data1.groupby('internet')[['sex','address']].sum().plot.bar(color=['red','green'],figsize=(8,5))
plt.ylabel('INTERNET USAGE')
plt.show()
```

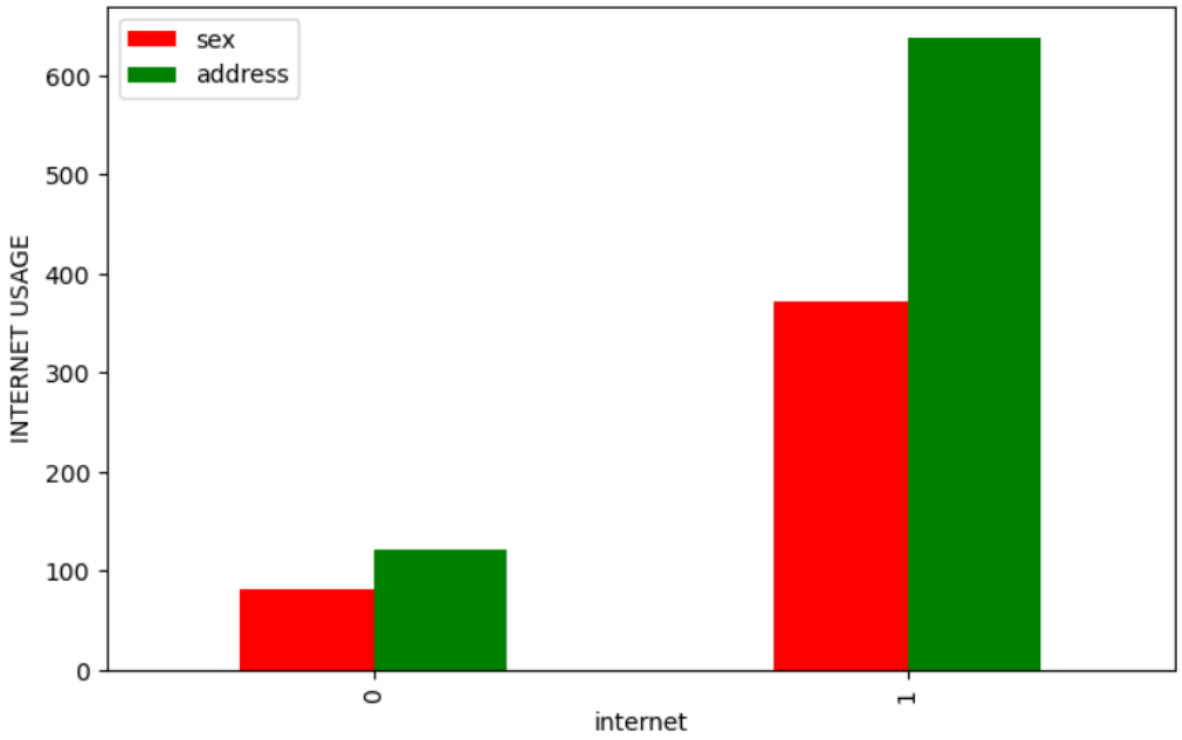


Fig.8 internet usage with address and sex

Then I applied pie chart for internet usage. This can be seen in fig.9, where we can see most of the students use internet.

```
plt.title('internet usage',fontsize=20)
student_data1['internet'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

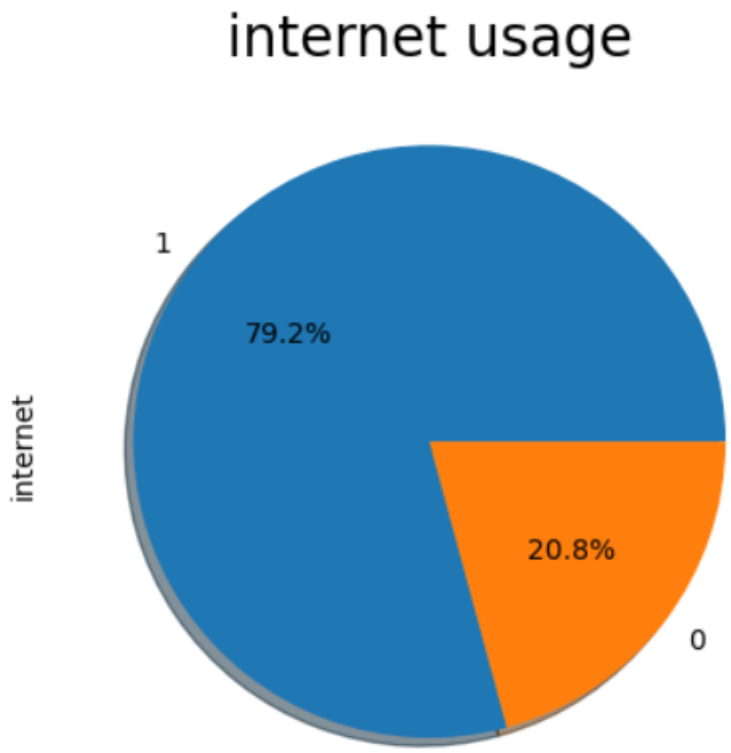


Fig.9 internet usage

Also, I plot to check the education level of parents where we can see mother's have high education level compared to fathers. This we can see in fig.10 and fig.11

```
plt.title('mothers education',fontsize=20)
student_data1['Medu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

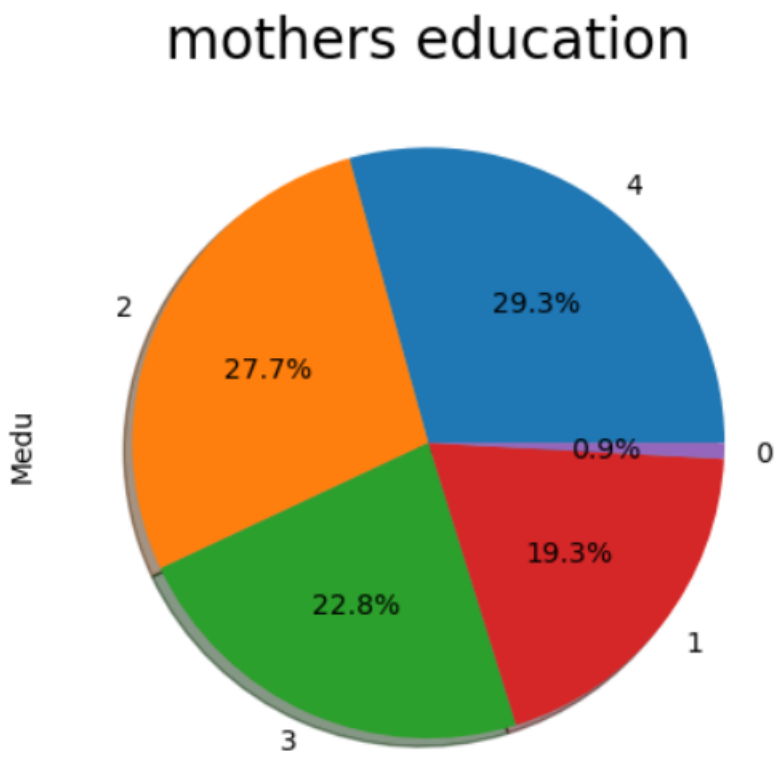


Fig.10 Mother’s education level

```
plt.title('fathers education',fontsize=20)
student_data1['Fedu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

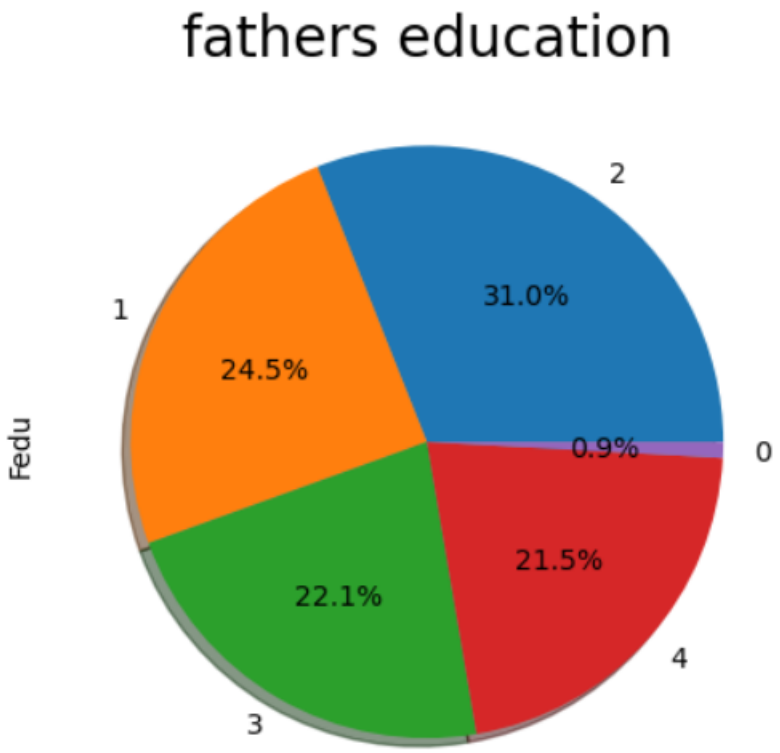


Fig.11 Father's education level

2.1.1 Advanced question 1

Here, I standardised the data using StandardScalar, covariance, and a heatmap on the covariance, which demonstrates that parent's employment has a stronger link with student achievement fig.12.

```
sns.heatmap(student_data1.cov(),vmin=-1,vmax=1,cmap='RdBu',linewidth=1,square=True)
plt.show()
```

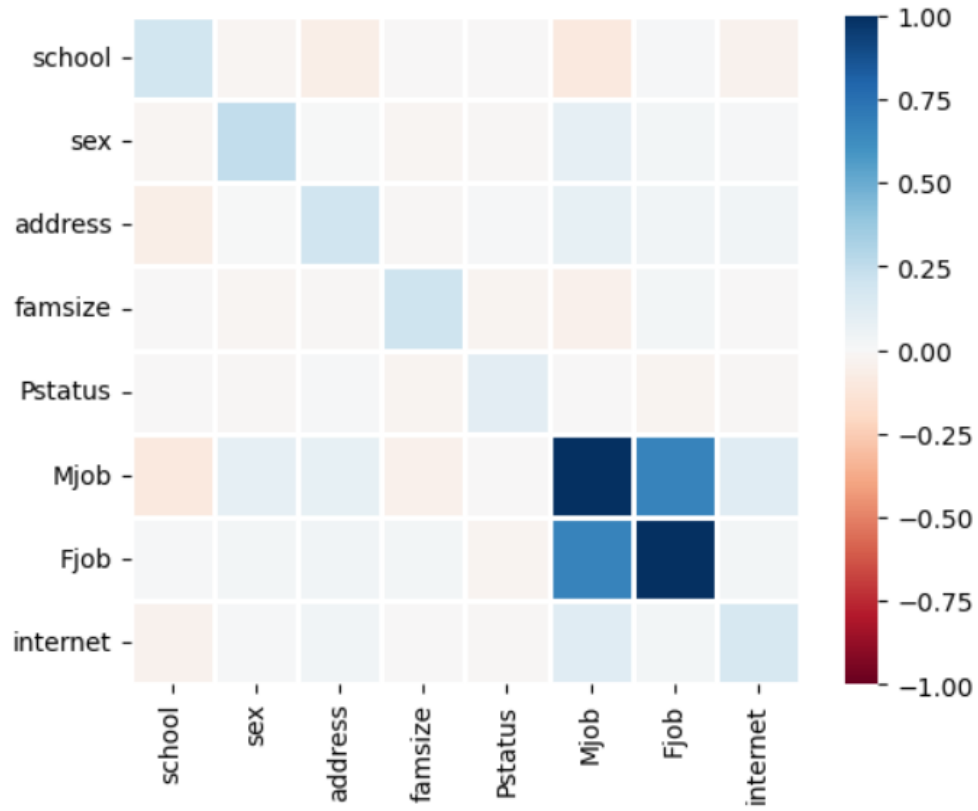


Fig.12 heatmap

2.2 Question 2

In this section I split the dataset using train_test_split from sklearn. model_selection with parameters test size and random_state. Then I fit a model RandomForestRegressor. After this I checked the mean square error which is 2.7

```
mse = mean_squared_error(y_test,y_predict)
mse
```

```
2.709438391281233
```

Fig 13 mean square error

2.2.1 Advanced question 2

Here, in fig.14, I divided the final grade column into three categories. I divided it by using def function as poor, average and well achieving.

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	internet	health	absences	G1	G2	G3
0	0	0	18	1	1	1	4	4	1	4	0	3	6	5	6	poor_achieving
1	0	0	17	1	1	0	1	1	1	0	1	3	4	5	5	poor_achieving
2	0	0	15	1	0	0	1	1	1	0	1	3	10	7	8	average_achieving
3	0	0	15	1	1	0	4	2	2	3	1	5	2	15	14	well_achieving
4	0	0	16	1	1	0	3	3	0	0	0	5	4	6	10	average_achieving
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	1	0	19	0	1	0	2	3	3	0	1	5	4	10	11	average_achieving
645	1	0	18	1	0	0	3	1	4	3	1	1	4	15	15	well_achieving
646	1	0	18	1	1	0	1	1	0	0	0	5	6	11	12	average_achieving
647	1	1	17	1	0	0	3	1	3	3	1	2	6	10	10	average_achieving
648	1	1	18	0	0	0	3	2	3	0	1	5	4	10	11	average_achieving

1044 rows × 16 columns

Fig.14 G3 column divided into 3 categories

Then I split the data and fit a model RandomForestClassifier and checked the accuracy of the data which is 86% fig.15.

```
accuracy_score(y1_test,y1_predict)
```

```
0.8660287081339713
```

Fig.15 accuracy score

The results of the study show that a few important variables, including sex, family size, Pstatus, parental education, and parental employment, usage of internet, might have a negative impact on students' academic performance. Also, the dataset accuracy is 86%, which shows it is good for prediction.

3. Literature Review

Data Mining (DM), a promising and new subject in computer science, is the one with the largest volume. The process of removing valuable information from a sizable database is known as data mining. Large databases were cleaned using data mining techniques to extract the most relevant data

that might be hidden. Therefore, these strategies also offer the ability to forecast the results of future observations (Ahmed & Sadiq, 2018). A random forest is an ensemble classifier that builds a collection of distinct decision trees based on the concept of randomization. The output categories of Random Forest are decided by the mode of the outcomes of the decision tree classification, which contains numerous decision tree classifiers(Ren et al., 2017).

One area that need more explanation to assist users in getting more information to make the best selection is the education sector. The Random Forest technique has been explained and applied in this research to analyse student performance using a dataset (Ahmed & Sadiq, 2018). In his work, the dataset from the students was leveraged to generate meaningful information using the Random Forest classification technique. High classification accuracy provided by such an algorithm is insufficient. Additionally, consumers should find this algorithm easier to understand and benefit from early decision-making assistance. The Random Forest classification algorithm has provided clarification on this. The new data has been tested on all trees in the forest to determine students' performance predictions, and the record can be labelled with the label that has received the most support. The Random Forest algorithm has been clarified by using the trees that were pulled from it using the tree construction algorithms. Also, he says that Random Forest generates K numbers of trees with various attributes each time, without pruning, by selecting the attributes at random. Unlike Random Forest, where the test data is checked on all the generated trees, Decision Tree simply tests one constructed tree, and the most common output is then assigned to that instance. In general, a forest with more trees will have more strong creatures. The similar concept applies to Random Forest Classifier; if a forest contains more trees, Random Forest methods will provide the highest accuracy. Additionally, Random Forest can accept missing values, does not care how many trees are in the forest, never overfits, and never considers the number of trees in the forest and they deal with categorical values.

Web page is another field where Random Forest (RF) is used. Web page classification is the automated tagging of a document with a predetermined subject category. Given that the online is a vast repository of different types of information, including photos, videos, and other types of content, automatic web page classification is one of the most important strategies for web mining. Additionally, categorization web pages are required to meet user needs. Each category's classification of web pages is done only by hand, which takes a lot of time and work. More researchers are concentrating about web page categorization technology to solve this manually classified difficulty. Web pages can be efficiently categorised by the RF classifier according to their appropriate class without the use of additional feature selection techniques (Aung et al., 2009). According to him, web pages can represent diverse data sources in a variety of ways and have many different dimensionalities. Numerous classification techniques were presented by researchers to classify this data source into appropriate classes. A general approach of classification is a decision tree. Decision tree classifiers are simple to comprehend and can be used to solve issues in the real world. Because it just classifies two classes, it is quicker. It can handle categorical and numerical data processing. However, it is only applicable to output attributes that have one. The RF classifier can identify several categories since it consists of numerous decision tree classifiers. Multiple categories must be categorised because of the number of web pages on the website.

(Belgiu & Drăguţ, 2016) says remote sensing has proven useful in a variety of industries, the success of any picture categorization depends on several criteria, including the selection of an appropriate classification method. Since supervised classifiers are more reliable than model-based methods, they are commonly employed. These classifiers can recognise these acquired properties in the unclassified data. They can also learn the characteristics of target classes from training samples. Also, his goal for

the work was to provide an overview of the RF classifier's application in remote sensing, paying particular attention to its parameterization and sensitivity to changes in sampling techniques, the size and representativeness of training sample sets, and data noise.

According to (Jayaprakash et al., 2020), RF can concentrate on grouping at-risk students according to numerous determining criteria. This research work assessed the effectiveness of several models created utilising classification algorithms in addition to identifying student performance. The Naive Bayes algorithm and other ensemble methods were surpassed by the Random Forest algorithm when iteration and bag size were increased. Also, (Ahmed & Sadiq, 2018) says that, Random Forest approach can be used as a white-box algorithm to improve classification accuracy for any dataset.

Reference

Ren, Q., Cheng, H., & Han, H. (2017). Research on machine learning framework based on random forest algorithm. AIP conference proceedings,

Ahmed, N. S., & Sadiq, M. H. (2018). Clarify of the random forest algorithm in an educational field. 2018 international conference on advanced science and engineering (ICOASE),

Aung, W. T., Myanmar, Y., & Hla, K. H. M. S. (2009). Random forest classifier for multi-category classification of web pages. 2009 IEEE Asia-Pacific Services Computing Conference (APSCC),

Belgiu, M., & Drăguţ, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS journal of photogrammetry and remote sensing*, *114*, 24-31.

Jayaprakash, S., Krishnan, S., & Jaiganesh, V. (2020). Predicting students academic performance using an improved random forest classifier. 2020 international conference on emerging smart computing and informatics (ESCI),

Mythili, M., & Shanavas, A. M. (2014). An Analysis of students' performance using classification algorithms. *IOSR Journal of Computer Engineering*, *16*(1), 63-69.

Appendix 1

Importing Libraries

```
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.stats import norm
import seaborn as sns
import numpy as np
import pandas as pd
```

Reading the two dataset and its shape

```
student_mat=pd.read_csv('studentmat.csv')
student_mat
student_mat.shape
```

```
student_por=pd.read_csv('studentpor.csv')
student_por
student_por.shape
```

Appending the two dataset

```
student_data=student_mat.append(student_por)
student_data
```

Reading the shape and column names

```
student_data.shape
student_data.columns
```

Descriptive Statistics

```
student_data.info()
student_data.describe()
student_data['G1'].describe()
student_data['G2'].describe()
student_data['G3'].describe()
student_data['sex'].value_counts()
student_data['age'].value_counts()
```

Checking missing values and cleaning it

```
student_data.isna().sum()
student_data.fillna(0,axis=1,inplace=True)
student_data.isna().sum().sum()
```

Dropping some columns

```
student_data1=student_data.drop(['reason','guardian','traveltime','studytime','failures',
                                 'schoolsup','famsup','paid','activities','nursery','higher',
                                 'romantic','famrel','freetime','goout','Dalc','Walc','fatherd'],axis=1)
student_data1
```

Checking the unique values of columns

```
student_data1['school'].unique()
student_data1['sex'].unique()
student_data1['address'].unique()
student_data1['famsize'].unique()
```

```
student_data1['Pstatus'].unique()

student_data1['Mjob'].unique()

student_data1['Fjob'].unique()

student_data1['internet'].unique()
```

Replacing the object to integers

```
student_data1.sex=student_data1.sex.replace({'M':1,'F':0}v)

student_data1.school=student_data1.school.replace({'MS':1,'GP':0})

student_data1.address=student_data1.address.replace({'U':1,'R':0})

student_data1.famsize=student_data1.famsize.replace({'GT3':1,'LE3':0})

student_data1.Pstatus=student_data1.Pstatus.replace({'A':1,'T':0})

student_data1.Mjob=student_data1.Mjob.replace({'health':2,'at_home':1,'services':3,'teacher':4,'other':0})

student_data1.Fjob=student_data1.Fjob.replace({'health':2,'at_home':1,'services':3,'teacher':4,'other':0})

student_data1.internet=student_data1.internet.replace({'yes':1,'no':0})

student_data1
```

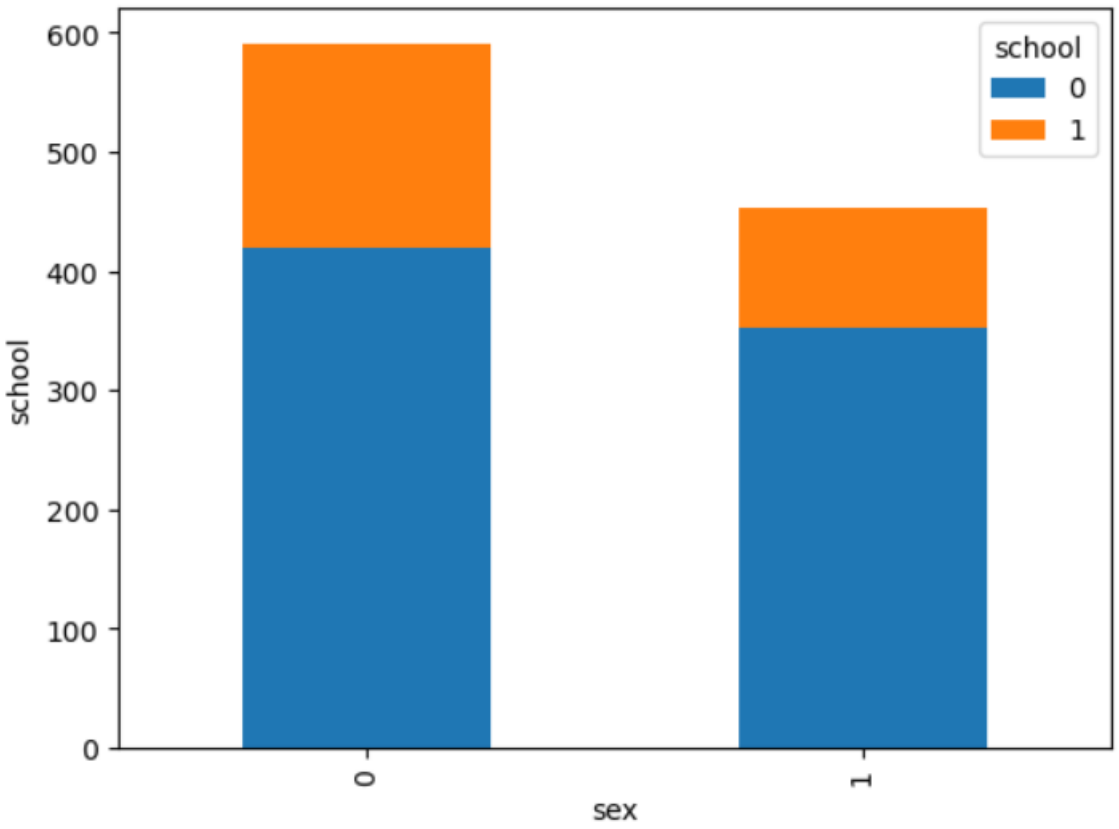
	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	internet	health	absences	G1	G2	G3
0	0	0	18	1	1	1	4	4	1	4	0	3	6	5	6	6
1	0	0	17	1	1	0	1	1	1	0	1	3	4	5	5	6
2	0	0	15	1	0	0	1	1	1	0	1	3	10	7	8	10
3	0	0	15	1	1	0	4	2	2	3	1	5	2	15	14	15
4	0	0	16	1	1	0	3	3	0	0	0	5	4	6	10	10
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	1	0	19	0	1	0	2	3	3	0	1	5	4	10	11	10
645	1	0	18	1	0	0	3	1	4	3	1	1	4	15	15	16
646	1	0	18	1	1	0	1	1	0	0	0	5	6	11	12	9
647	1	1	17	1	0	0	3	1	3	3	1	2	6	10	10	10
648	1	1	18	0	0	0	3	2	3	0	1	5	4	10	11	11

1044 rows × 16 columns

Objects replaced to integer

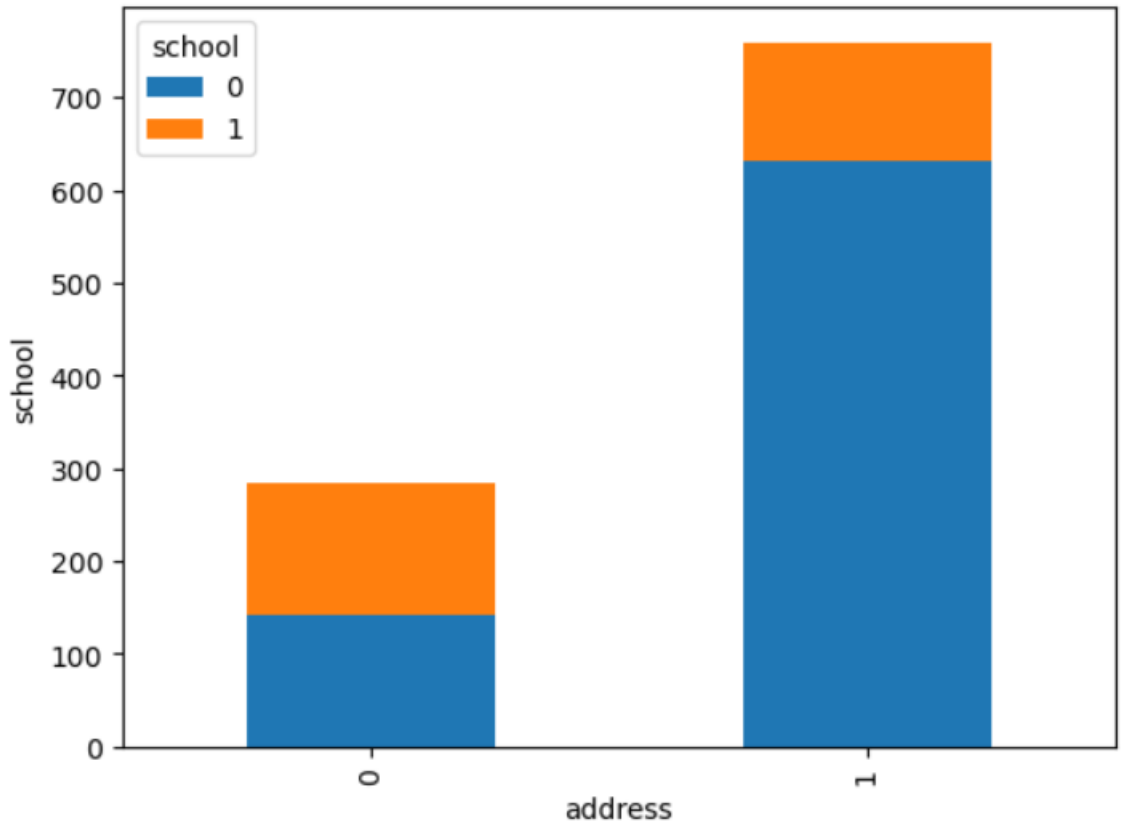
Plotting some columns

```
ss=pd.crosstab(student_data1['sex'],student_data1['school'])
ss
ss.plot(kind='bar',stacked=True)
plt.xlabel('sex')
plt.ylabel('school')
plt.show()
```



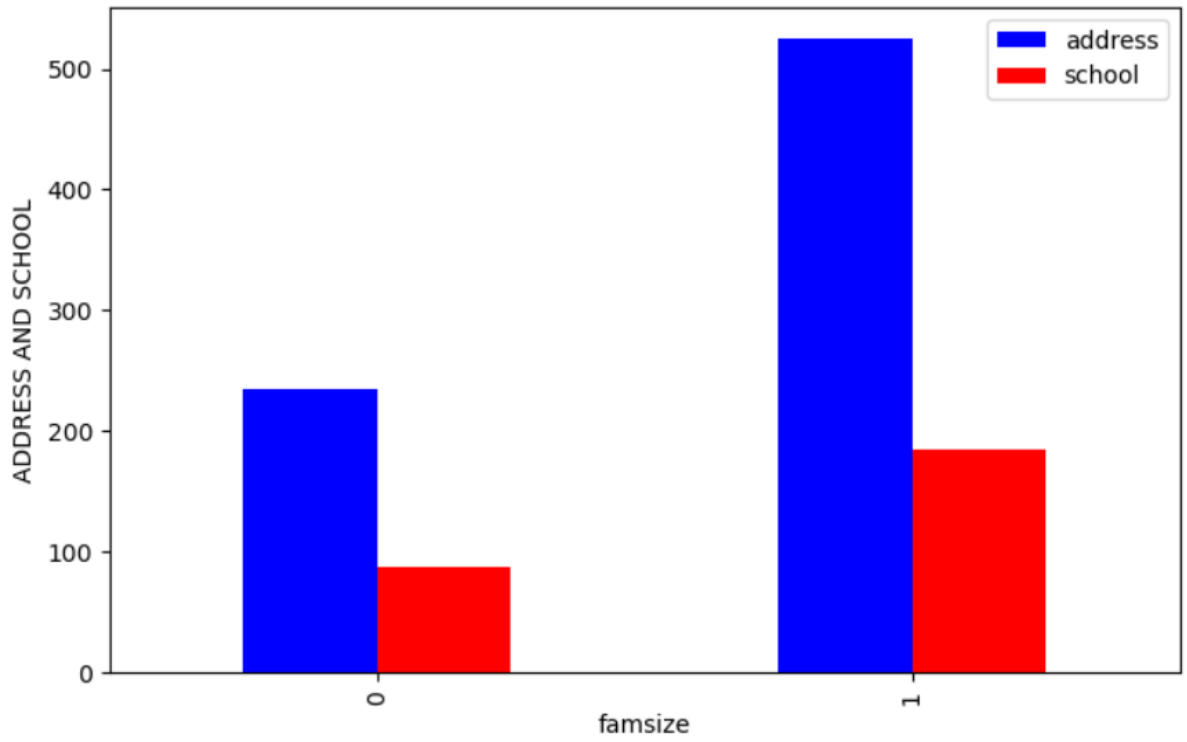
Sex vs school

```
sa=pd.crosstab(student_data1['address'],student_data1['school'])
sa
sa.plot(kind='bar',stacked=True)
plt.xlabel('address')
plt.ylabel('school')
plt.show()
```



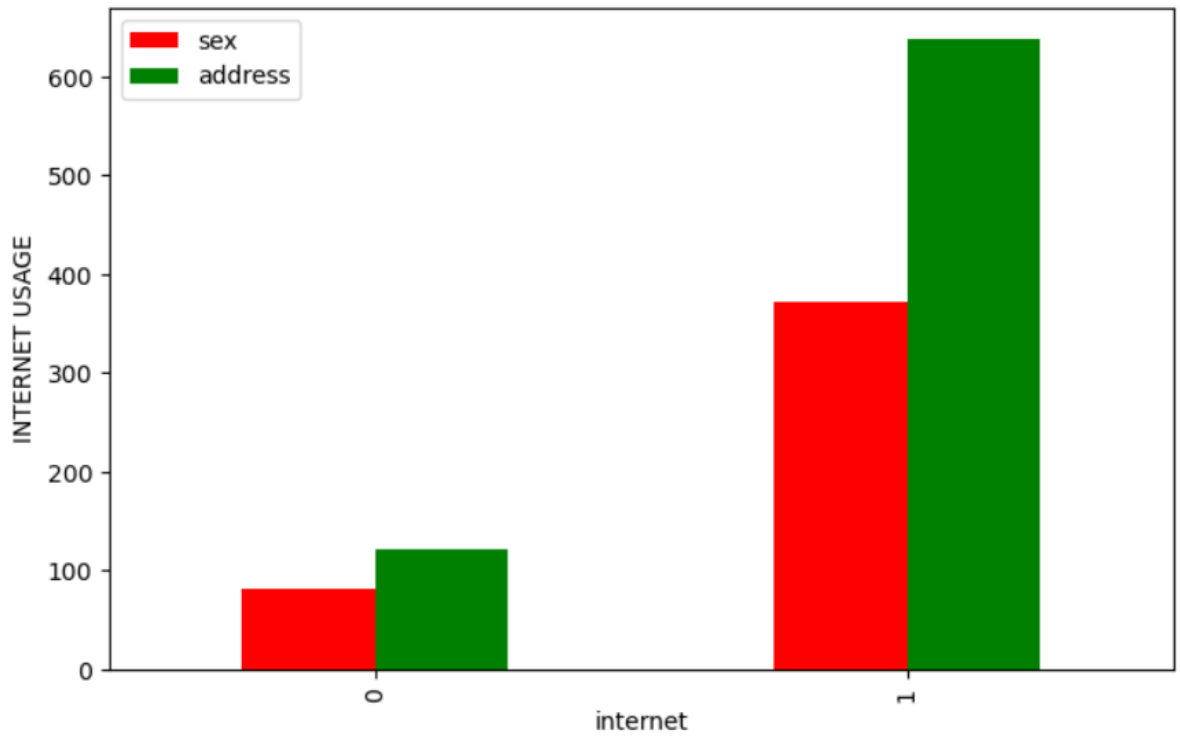
Address vs school

```
student_data1.groupby('famsize')[['address','school']].sum().plot.bar(color=['blue','red'],figsize=(8,5)
)

plt.ylabel('ADDRESS AND SCHOOL')

plt.show()
```



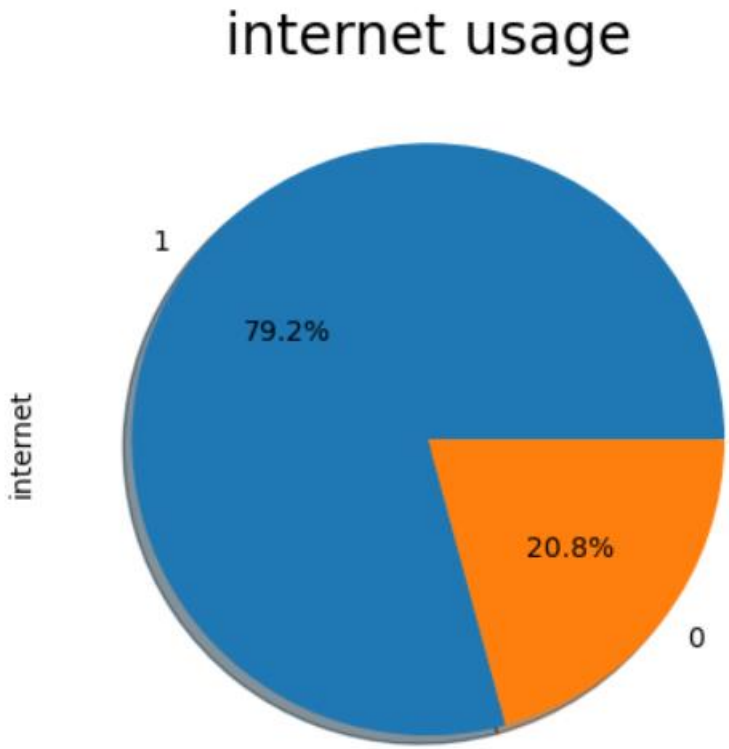
Grouped bar plot for address and school

```
student_data1.groupby('internet')[['sex','address']].sum().plot.bar(color=['red','green'],figsize=(8,5))
plt.ylabel('INTERNET USAGE')
plt.show()
```



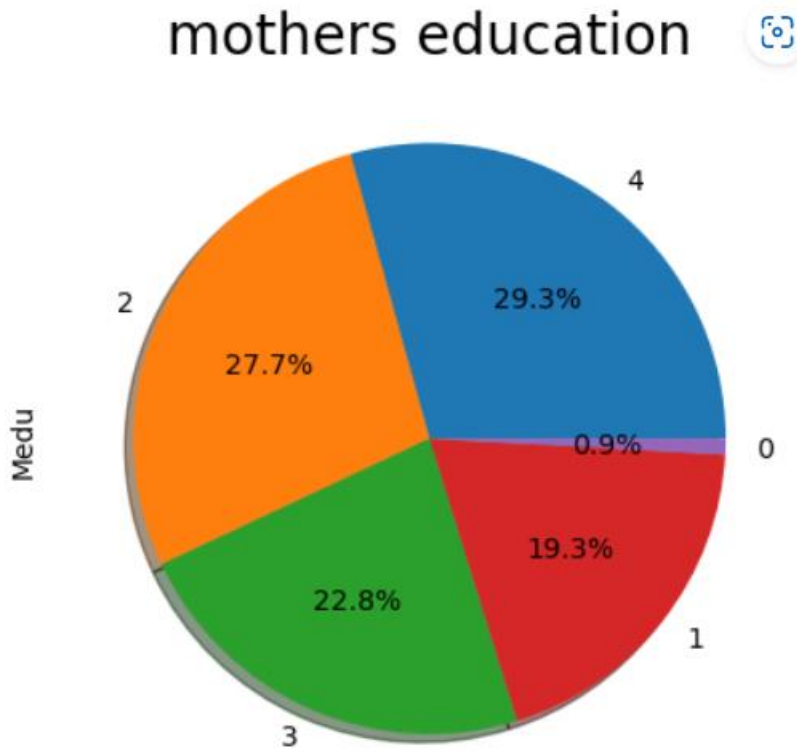
Internet usage

```
plt.title('internet usage',fontsize=20)
student_data1['internet'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

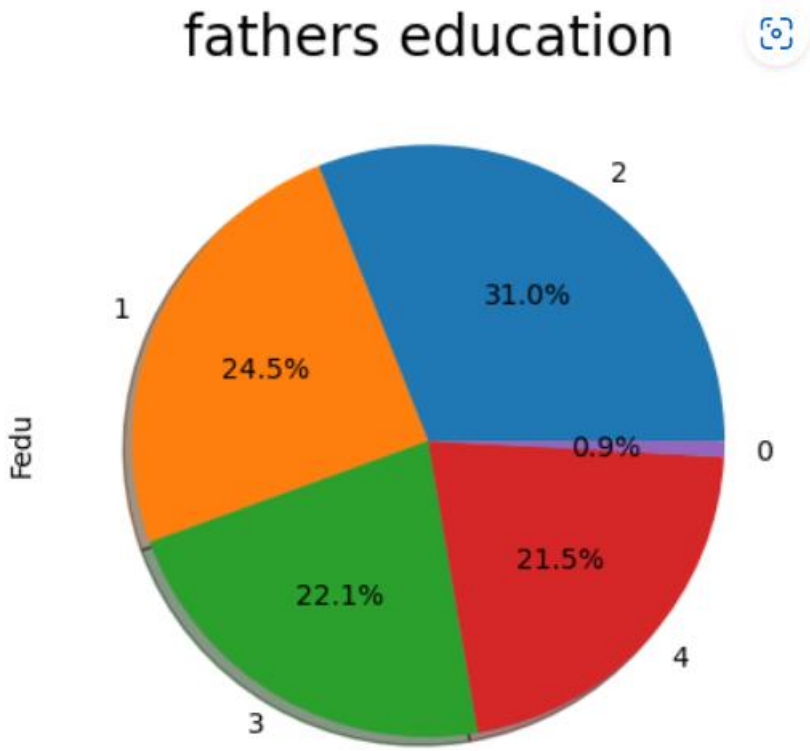


pie plot for internet

```
plt.title('mothers education',fontsize=20)
student_data1['Medu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

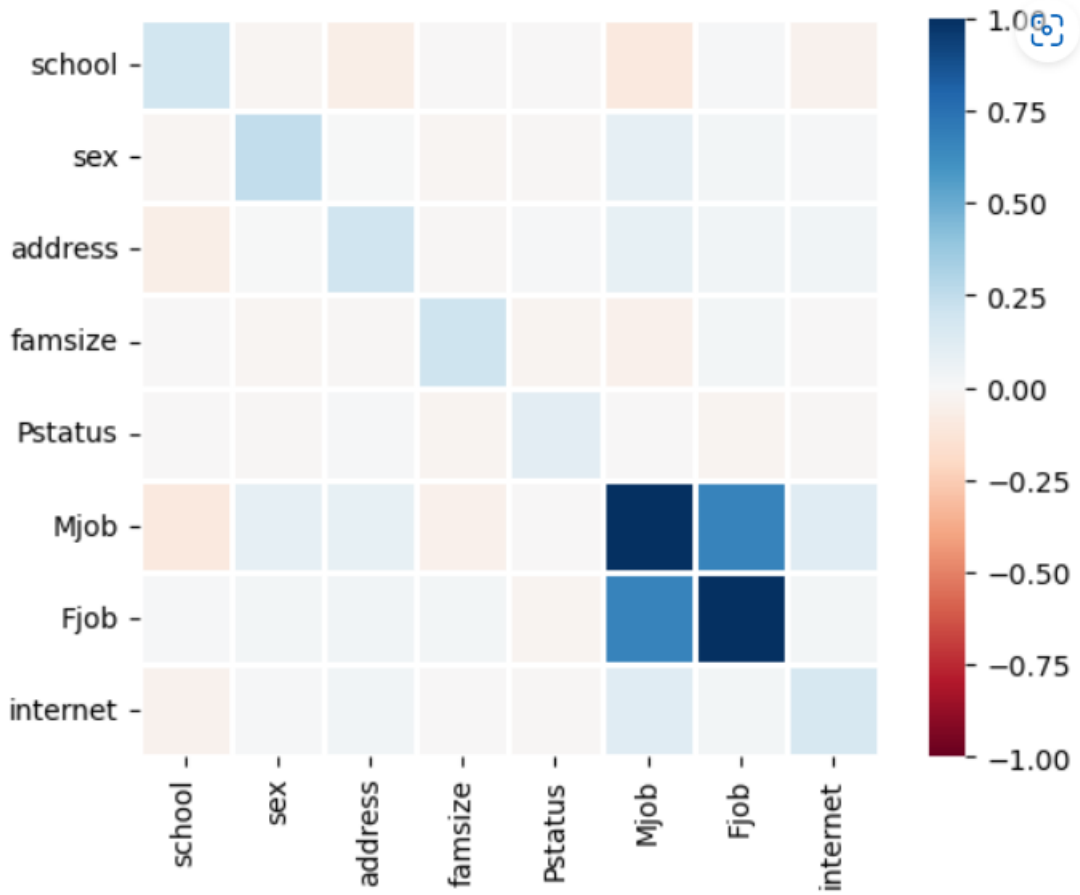


```
plt.title('fathers education',fontsize=20)
student_data1['Fedu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```



Factor analysis

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(student_data1)
scaled_data = pd.DataFrame(scaled_data, columns=student_data1.columns)
scaled_data.head(3)

student_data1.cov()
sns.heatmap(student_data1.cov(),vmin=-1,vmax=1,cmap='RdBu',linewidth=1,square=True)
plt.show()
```



Heatmap on covariance

Random Forest Regression

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
x=student_data1.drop(['G3'],axis=1)
y=student_data1['G3']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
model = RandomForestRegressor(random_state=1)
model.fit(x_train,y_train)
y_predict = model.predict(x_test)
```

y_predict

from sklearn.metrics import mean_squared_error

mse = mean_squared_error(y_test,y_predict)

mse

```
mse = mean_squared_error(y_test,y_predict)
mse
```

```
2.709438391281233
```

Dividing the G3 column into 3 categories

```
def G3(grade):
    if grade<7:
        return 'poor_achieving'
    elif grade<14:
        return 'average_achieving'
    else:
        return 'well_achieving'
student_data1['G3']=student_data1['G3'].apply(G3)
```

student_data1

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	internet	health	absences	G1	G2	G3
0	0	0	18	1	1	1	4	4	1	4	0	3	6	5	6	poor_achieving
1	0	0	17	1	1	0	1	1	1	0	1	3	4	5	5	poor_achieving
2	0	0	15	1	0	0	1	1	1	0	1	3	10	7	8	average_achieving
3	0	0	15	1	1	0	4	2	2	3	1	5	2	15	14	well_achieving
4	0	0	16	1	1	0	3	3	0	0	0	5	4	6	10	average_achieving
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
644	1	0	19	0	1	0	2	3	3	0	1	5	4	10	11	average_achieving
645	1	0	18	1	0	0	3	1	4	3	1	1	4	15	15	well_achieving
646	1	0	18	1	1	0	1	1	0	0	0	5	6	11	12	average_achieving
647	1	1	17	1	0	0	3	1	3	3	1	2	6	10	10	average_achieving
648	1	1	18	0	0	0	3	2	3	0	1	5	4	10	11	average_achieving

1044 rows × 16 columns

Importing RandomForestClassifier and splitting the data

from sklearn.ensemble import RandomForestClassifier

x1=student_data1.drop(['G3'],axis=1)

y1=student_data1['G3']

x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.2)

Fitting the model

model1=RandomForestClassifier(n_estimators=20)

model1.fit(x1_train,y1_train)

y1_predict=model1.predict(x1_test)

y1_predict

Checking accuracy score

from sklearn.metrics import accuracy_score

accuracy_score(y1_test,y1_predict)

```
accuracy_score(y1_test,y1_predict)
```

```
0.8660287081339713
```

Applying confusion matrix

from sklearn.metrics import confusion_matrix

performance=confusion_matrix(y1_test,y1_predict)

performance

```
array([[152,   4,   1],
       [  7,   4,   0],
       [ 10,   0,  31]], dtype=int64)
```

Appendix 1.1 Question 1 Code

Appending the two dataset

student_data=student_mat.append(student_por)

```
student_data
```

Reading the shape and column names

```
student_data.shape
```

```
student_data.columns
```

Descriptive Statistics

```
student_data.info()
student_data.describe()
student_data['G1'].describe()
student_data['G2'].describe()
student_data['G3'].describe()
student_data['sex'].value_counts()
student_data['age'].value_counts()
```

Checking missing values and cleaning it

```
student_data.isna().sum()
student_data.fillna(0,axis=1,inplace=True)
student_data.isna().sum().sum()
```

Dropping some columns

```
student_data1=student_data.drop(['reason','guardian','traveltime','studytime','failures',
                                 'schoolsup','famsup','paid','activities','nursery','higher',
                                 'romantic','famrel','freetime','goout','Dalc','Walc','fatherd'],axis=1)
student_data1
```

Checking the unique values of columns

```
student_data1['school'].unique()

student_data1['sex'].unique()

student_data1['address'].unique()

student_data1['famsize'].unique()

student_data1['Pstatus'].unique()

student_data1['Mjob'].unique()

student_data1['Fjob'].unique()

student_data1['internet'].unique()
```

Replacing the object to integers

```
student_data1.sex=student_data1.sex.replace({'M':1,'F':0}v)

student_data1.school=student_data1.school.replace({'MS':1,'GP':0})

student_data1.address=student_data1.address.replace({'U':1,'R':0})

student_data1.famsize=student_data1.famsize.replace({'GT3':1,'LE3':0})

student_data1.Pstatus=student_data1.Pstatus.replace({'A':1,'T':0})

student_data1.Mjob=student_data1.Mjob.replace({'health':2,'at_home':1,'services':3,'teacher':4,'other':0})

student_data1.Fjob=student_data1.Fjob.replace({'health':2,'at_home':1,'services':3,'teacher':4,'other':0})

student_data1.internet=student_data1.internet.replace({'yes':1,'no':0})
```

Plotting some columns

Stacked bar plot

```
ss=pd.crosstab(student_data1['sex'],student_data1['school'])

ss

ss.plot(kind='bar',stacked=True)

plt.xlabel('sex')
```

```
plt.ylabel('school')
plt.show()

sa=pd.crosstab(student_data1['address'],student_data1['school'])
sa
sa.plot(kind='bar',stacked=True)
plt.xlabel('address')
plt.ylabel('school')
plt.show()
```

Grouped plot

```
student_data1.groupby('famsize')[['address','school']].sum().plot.bar(color=['blue','red'],figsize=(8,5)
)
plt.ylabel('ADDRESS AND SCHOOL')
plt.show()

student_data1.groupby('internet')[['sex','address']].sum().plot.bar(color=['red','green'],figsize=(8,5))
plt.ylabel('INTERNET USAGE')
plt.show()
```

Pie plot

```
plt.title('internet usage',fontsize=20)
student_data1['internet'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()

plt.title('mothers education',fontsize=20)
student_data1['Medu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()

plt.title('fathers education',fontsize=20)
student_data1['Fedu'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True)
plt.show()
```

Appendix 1.2 Advanced Question 1 Code

Factor analysis

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

scaled_data = scaler.fit_transform(student_data1)
scaled_data = pd.DataFrame(scaled_data, columns=student_data1.columns)
scaled_data.head(3)

student_data1.cov()
sns.heatmap(student_data1.cov(),vmin=-1,vmax=1,cmap='RdBu',linewidth=1,square=True)
plt.show()
```

Appendix 1.3 Question 2 Code

Random Forest Regression

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
x=student_data1.drop(['G3'],axis=1)
y=student_data1['G3']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
model = RandomForestRegressor(random_state=1)
```

```
model.fit(x_train,y_train)
y_predict = model.predict(x_test)
y_predict
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test,y_predict)
mse
```

Appendix 1.4 Advanced Question 2 Code

Dividing the G3 column into 3 categories

```
def G3(grade):
    if grade<7:
        return 'poor_achieving'
    elif grade<14:
        return 'average_achieving'
    else:
        return 'well_achieving'
student_data1['G3']=student_data1['G3'].apply(G3)
student_data1
```

Importing RandomForestClassifier and splitting the data

```
from sklearn.ensemble import RandomForestClassifier
x1=student_data1.drop(['G3'],axis=1)
y1=student_data1['G3']
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.2)
```

Fitting the model

```
model1=RandomForestClassifier(n_estimators=20)
model1.fit(x1_train,y1_train)
y1_predict=model1.predict(x1_test)
y1_predict
```

Checking accuracy score

```
from sklearn.metrics import accuracy_score

accuracy_score(y1_test,y1_predict)
```

Applying confusion matrix

```
from sklearn.metrics import confusion_matrix

performance=confusion_matrix(y1_test,y1_predict)

performance
```

```
array([[152,   4,   1],
       [  7,   4,   0],
       [ 10,   0,  31]], dtype=int64)
```