student_data_insights

September 6, 2022

1 Data Analysis on student performance dataset

1.0.1 Problem Statement

```
Predict student performance in secondary education (high school)
```

1.0.2 Dataset Information

This data approach student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographic, social and other school related features.

1.0.3 Attributes Information

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 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - 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 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)

14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)

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)

These grades are related with the course subject, Math or Portuguese:

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)

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

Libraries used :

1. For Data Analysis : pandas, numpy, matplotlib, seaborn
2. For Predictive Model : sklearn, statsmodels

2 Loading dataset into pandas dataframe

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from matplotlib.pyplot import figure
```

```python
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```python
data  = pd.read_csv('data/stu_train.csv')
```

[4]:
```python
data
```

[4]:
```
      Id school sex  age address famsize Pstatus  Medu  Fedu      Mjob  ...  \
0      1     GP   F   18       U     GT3       A     4     4   at_home  ...
1      2     GP   F   17       U     GT3       T     1     1   at_home  ...
2      3     GP   F   15       U     LE3       T     1     1   at_home  ...
3      4     GP   F   15       U     GT3       T     4     2    health  ...
4      5     GP   F   16       U     GT3       T     3     3     other  ...
..   ...    ...  ..  ...     ...     ...     ...   ...   ...       ...  ...
513  514     MS   F   16       U     GT3       T     3     1     other  ...
514  515     MS   F   16       U     GT3       T     3     2  services  ...
515  516     MS   F   18       U     LE3       T     1     1     other  ...
516  517     MS   F   16       R     GT3       T     4     4    health  ...
517  518     MS   F   16       R     LE3       T     1     2     other  ...

    famrel freetime goout  Dalc  Walc  health absences  G1  G2  G3
0        4        3     4     1     1       3        4   0  11  11
1        5        3     3     1     1       3        2   9  11  11
2        4        3     2     2     3       3        6  12  13  12
3        3        2     2     1     1       5        0  14  14  14
4        4        3     2     1     2       5        0  11  13  13
..     ...      ...   ...   ...   ...     ...      ...  ..  ..  ..
513      3        1     3     1     3       1        0   8   6   8
514      3        1     3     1     4       3        2   7   6   7
515      2        3     5     1     4       3        8   9   8  10
516      4        3     3     2     3       2        0  14  16  16
517      5        4     5     1     4       2        0  14  14  15

[518 rows x 34 columns]
```

3 Checking for NaN values in dataframe

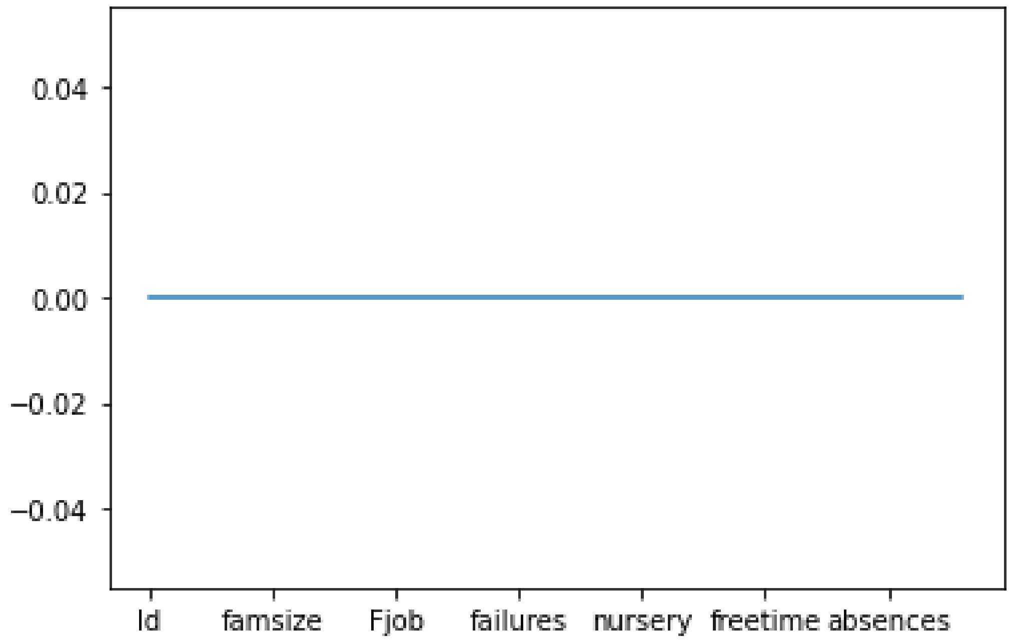
[5]:
```python
data.isnull().sum()
```

[5]:
```
Id          0
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          0
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
dtype: int64
```

[6]:
```
data.isnull().sum().plot(kind='line')
```

[6]:
```
<AxesSubplot:>
```



Found No NaN values, so we can procede with our analysis

4 Differentiating categorical and Numerical variables

Categorical variables : The variables with classify the attribute into different groups

Numerical variables : The variables that provide a numerical value to the attribute

[7]:
```
categorical_vars = data.dtypes[data.dtypes == 'object']
numerical_vars = data.dtypes[data.dtypes == 'int64']

categorical_vars, numerical_vars
```

[7]:
```
(school        object
 sex           object
 address       object
 famsize       object
 Pstatus       object
 Mjob          object
 Fjob          object
 reason        object
 guardian      object
 schoolsup     object
 famsup        object
 paid          object
 activities    object
```

```
nursery        object
higher         object
internet       object
romantic       object
dtype: object,
Id              int64
age             int64
Medu            int64
Fedu            int64
traveltime      int64
studytime       int64
failures        int64
famrel          int64
freetime        int64
goout           int64
Dalc            int64
Walc            int64
health          int64
absences        int64
G1              int64
G2              int64
G3              int64
dtype: object)
```

5 Plots for numerical and categorical variables

[45]:
```
data.columns
```

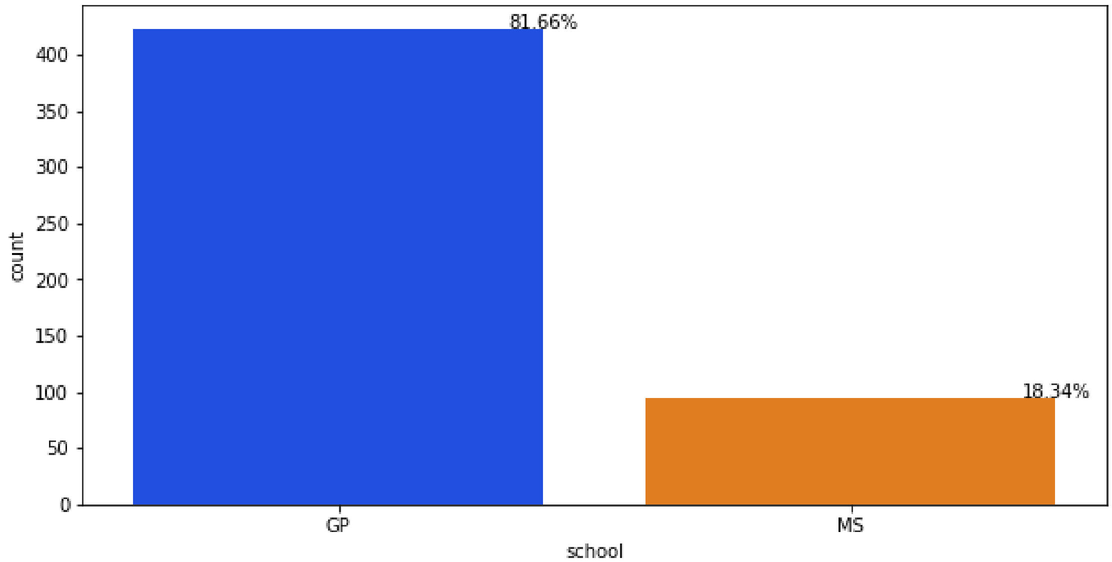
[45]:
```
Index(['Id', 'school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu',
       'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2', 'G3'],
      dtype='object')
```

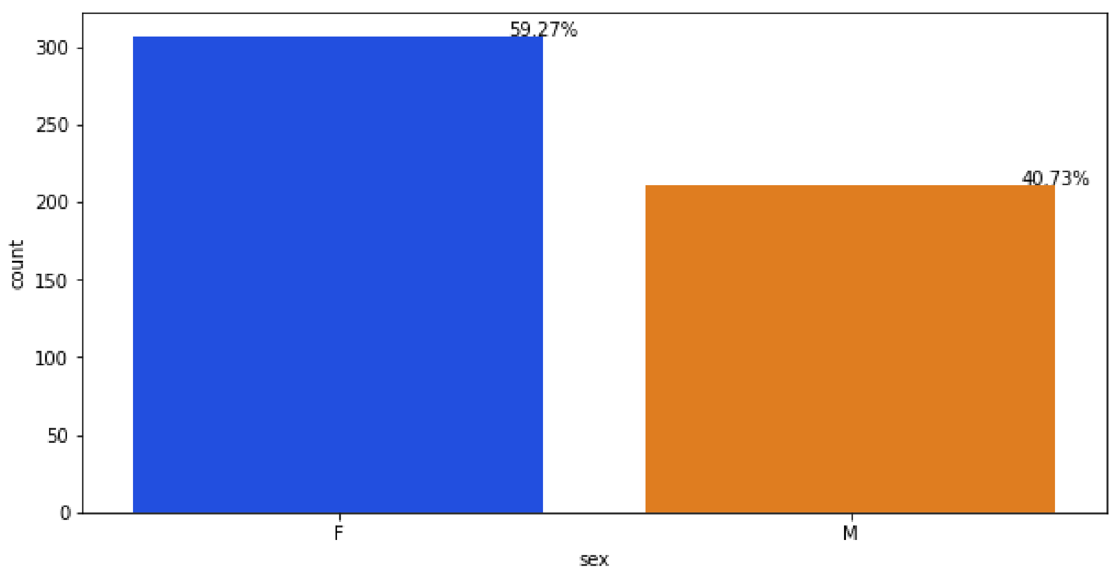
- We can observe that there are 34 columns in the dataset

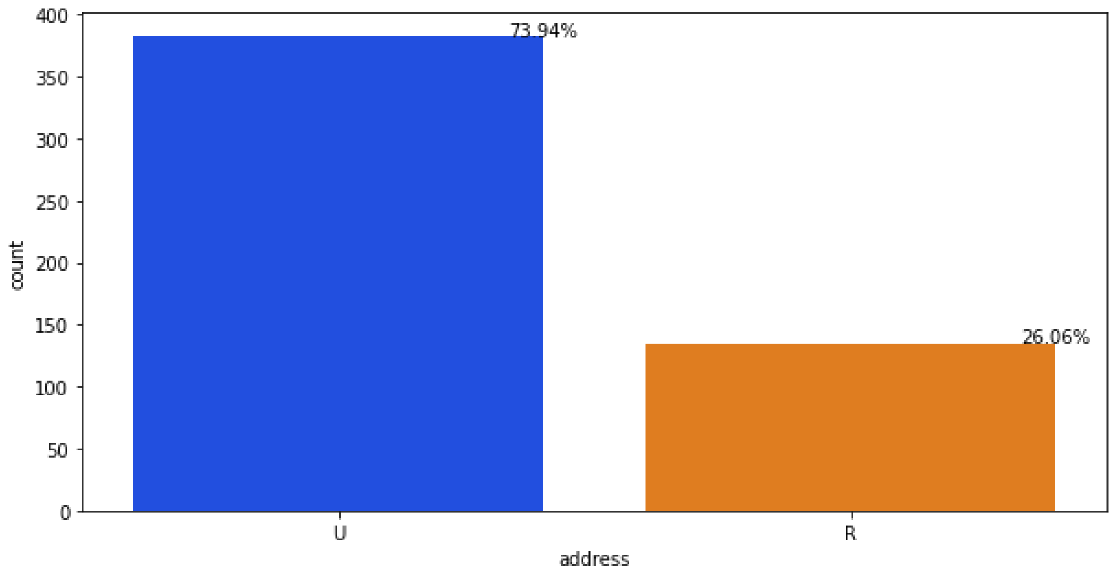
5.1 Categorical data plots

[9]:
```
for i in categorical_vars.index:
    plt.figure(figsize= (10,5))
    graph = sns.countplot(x = i, data = data, palette= 'bright')
    total = float(len(data))
    for p in graph.patches:
        pct = f"{(100 * p.get_height()/total):.2f}%"
        x = p.get_x() + p.get_width()
```

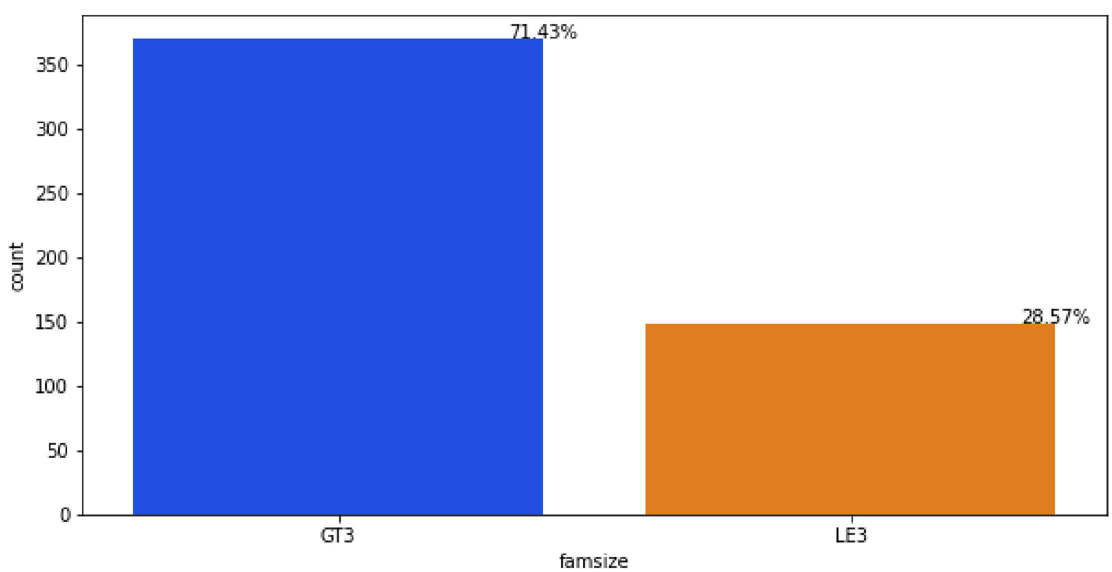
```
        y = p.get_height()
        graph.annotate(pct, (x, y),ha='center')

    plt.show()
```



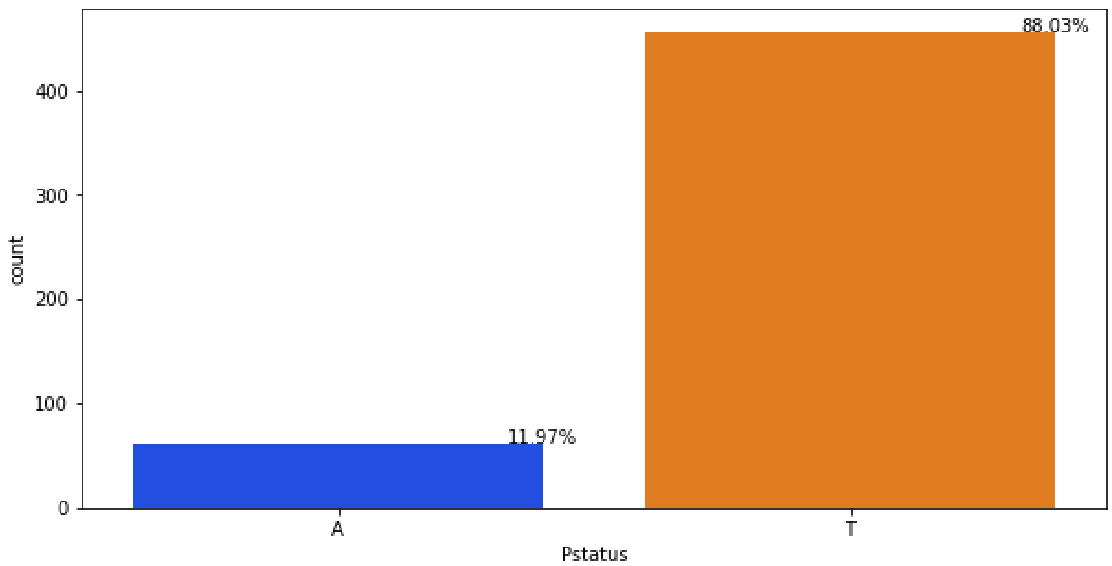




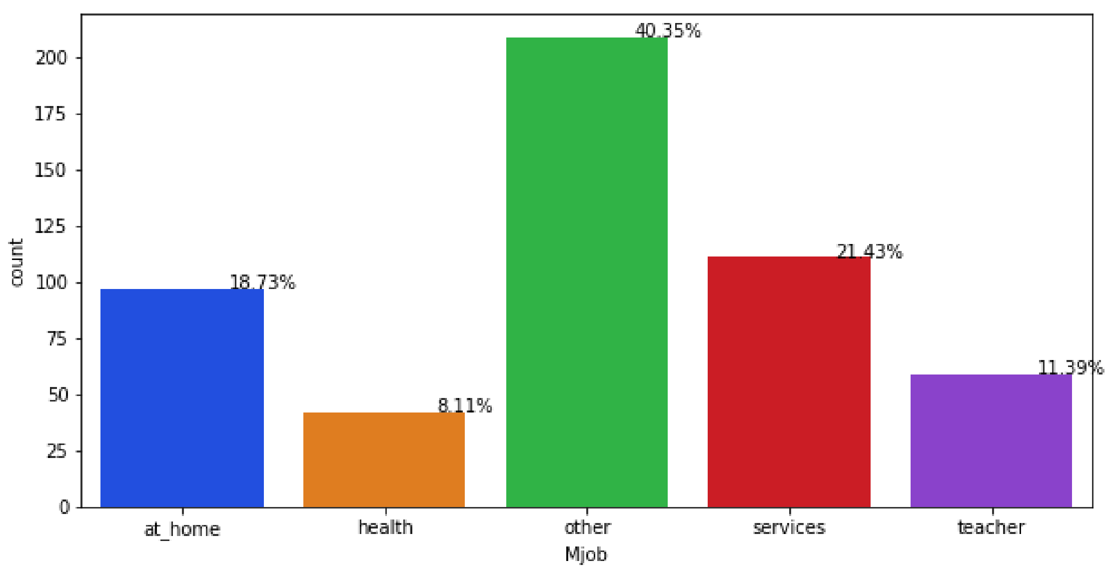
400
350
300
250
200
150
100
50
0
count
73.94%
26.06%
U
R
address



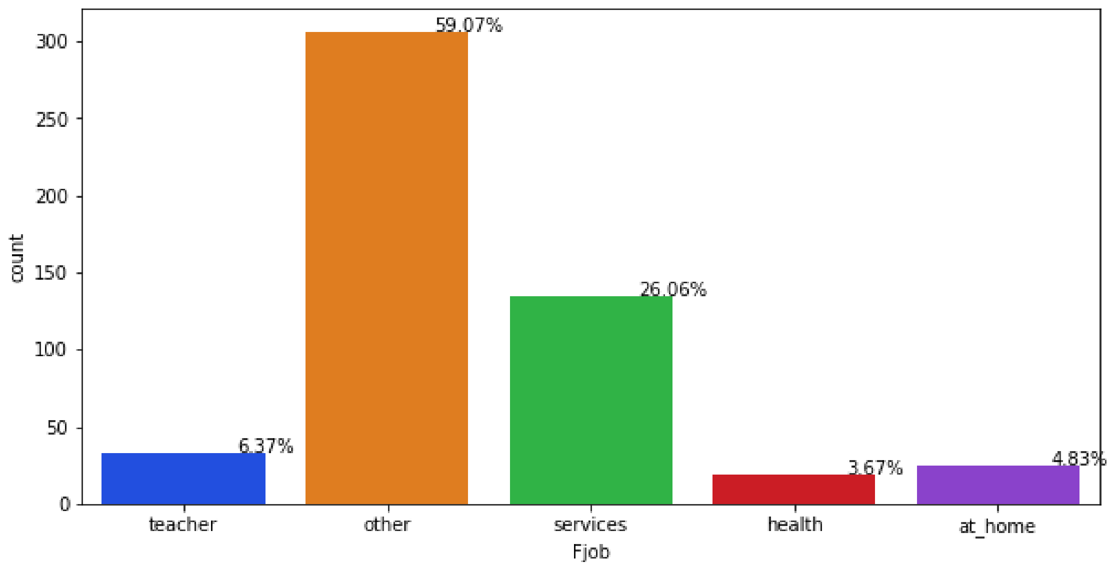
350
300
250
200
150
100
50
0
count
71.43%
28.57%
GT3
LE3
famsize



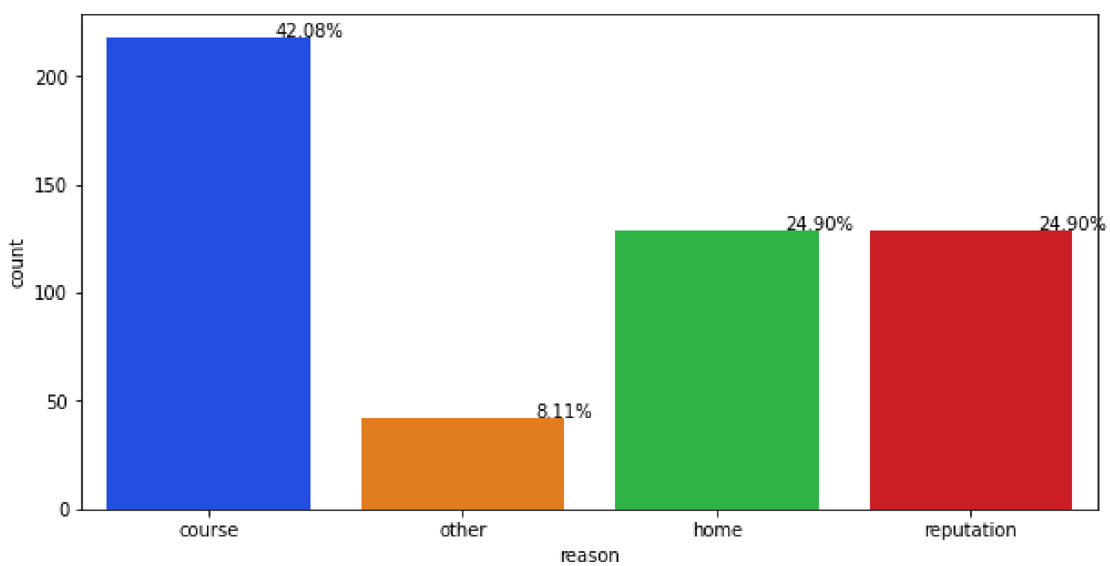
88.03%
11.97%
400
300
200
100
0
count
A
T
Pstatus



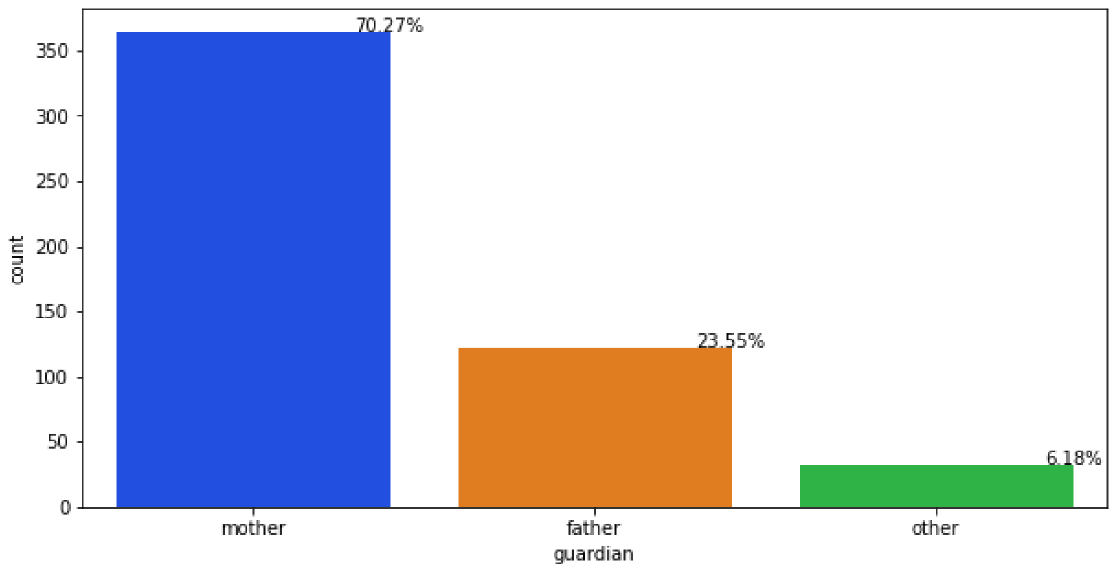
40.35%
21.43%
18.73%
11.39%
8.11%
200
175
150
125
100
75
50
25
0
count
at_home
health
other
services
teacher
Mjob

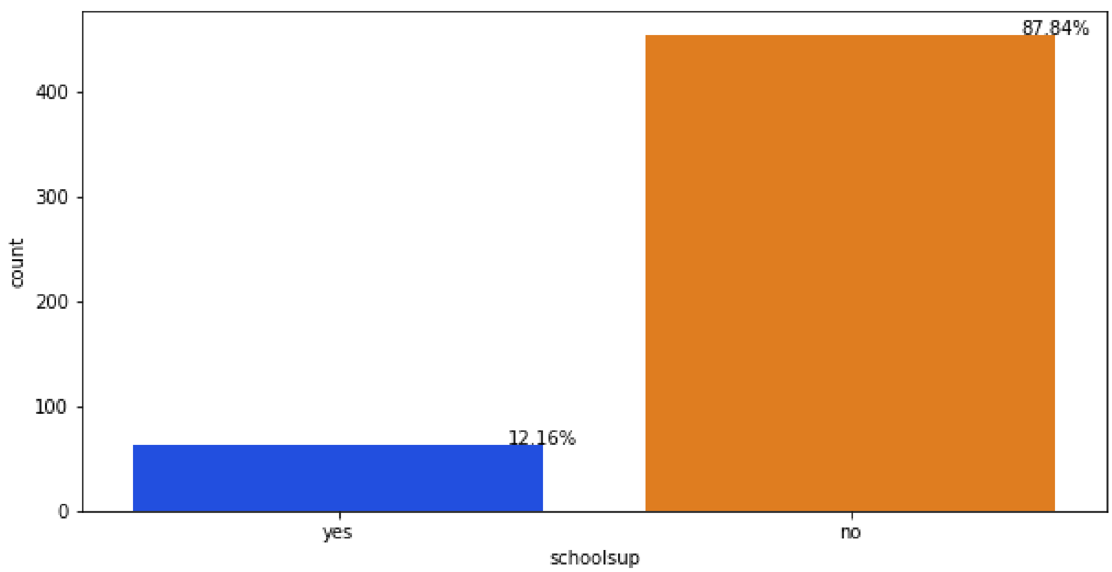


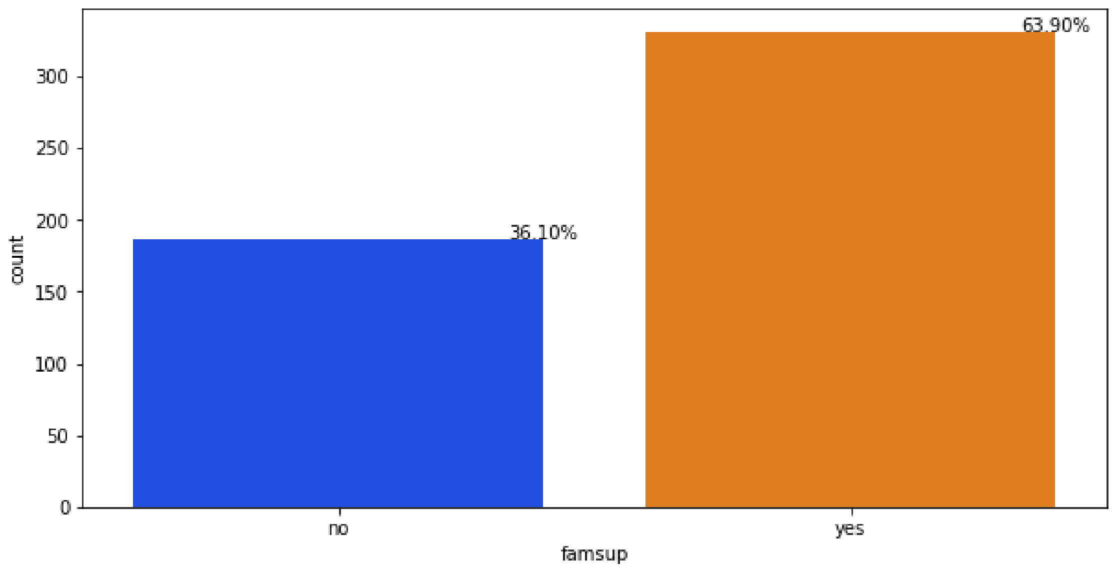
count
Fjob
teacher
other
services
health
at_home
6.37%
59.07%
26.06%
3.67%
4.83%
0
50
100
150
200
250
300

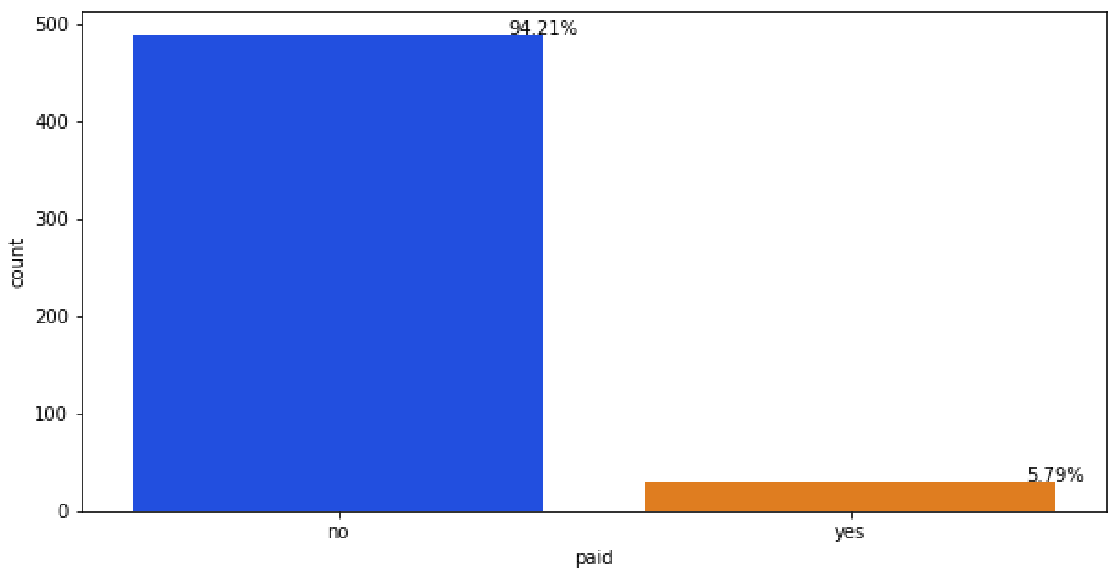


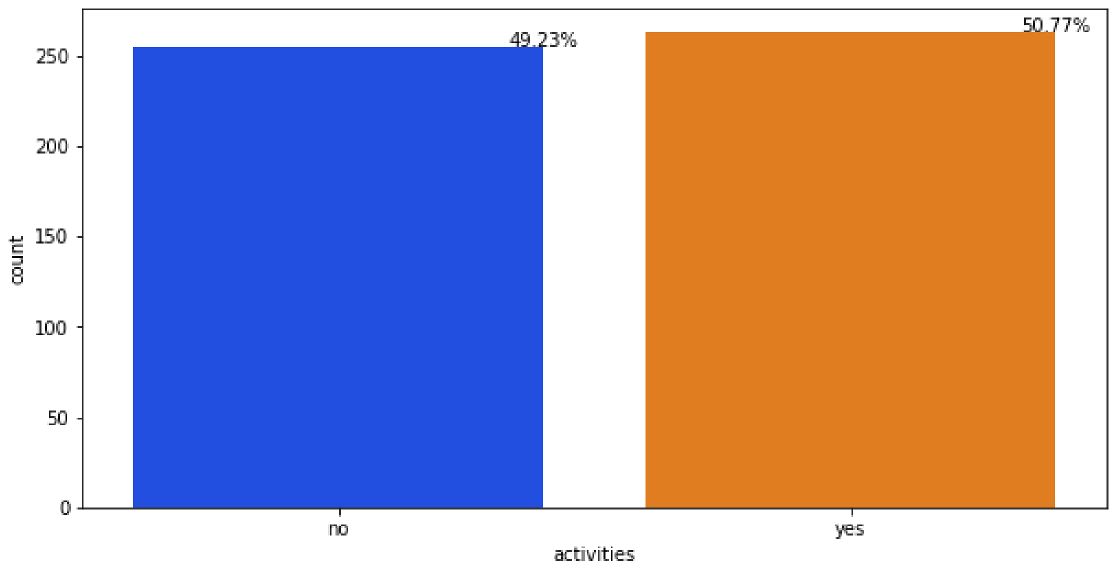
count
reason
course
other
home
reputation
42.08%
8.11%
24.90%
24.90%
0
50
100
150
200

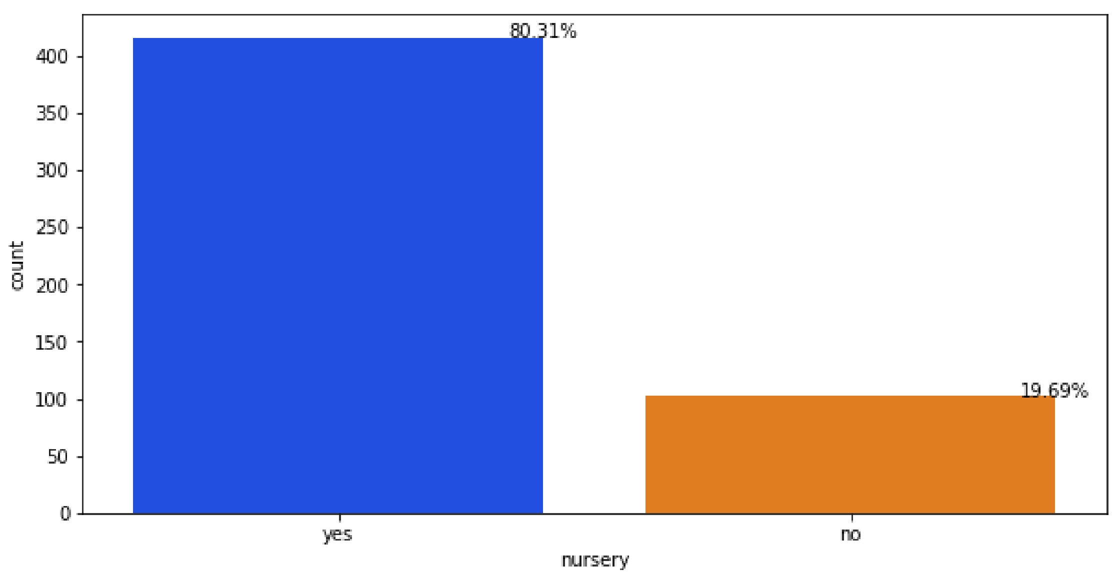

70.27%
23.55%
6.18%
count
350
300
250
200
150
100
50
0
mother
father
other
guardian

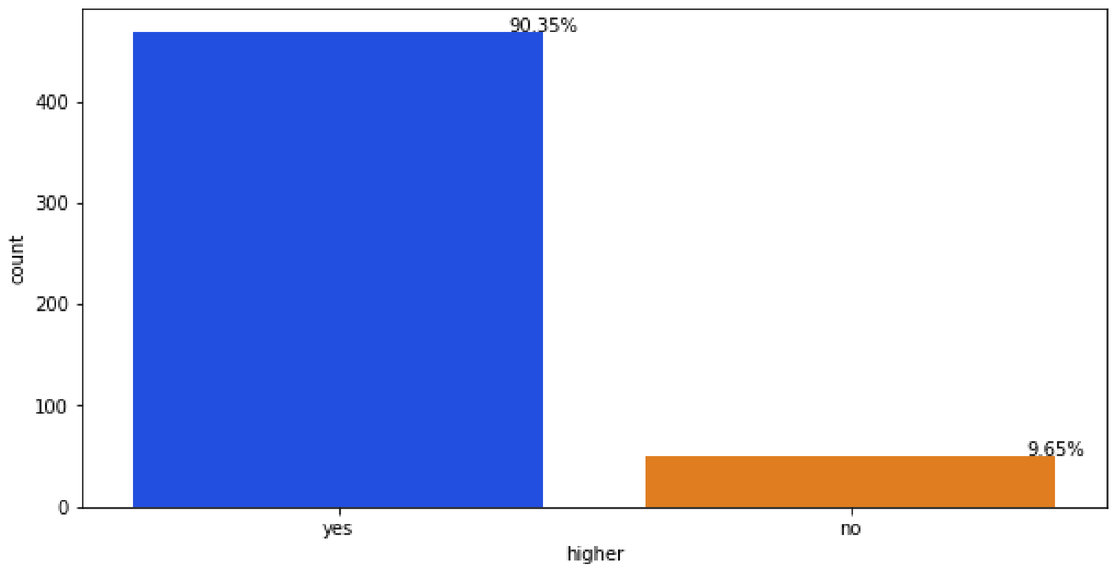

87.84%
12.16%
count
400
300
200
100
0
yes
no
schoolsup

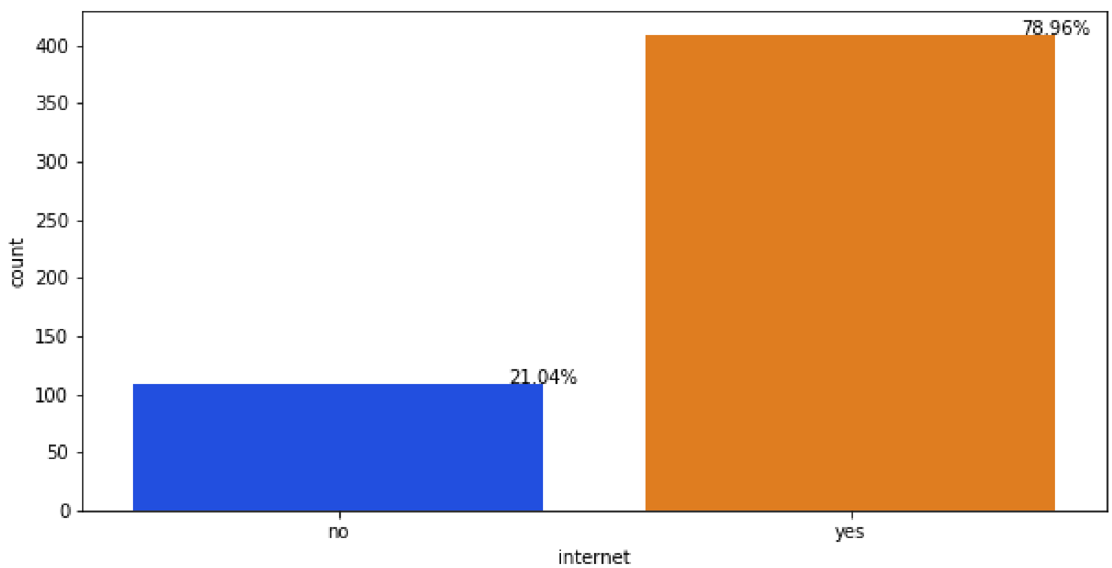

63.90%
36.10%
count
0
50
100
150
200
250
300
no
yes
famsup

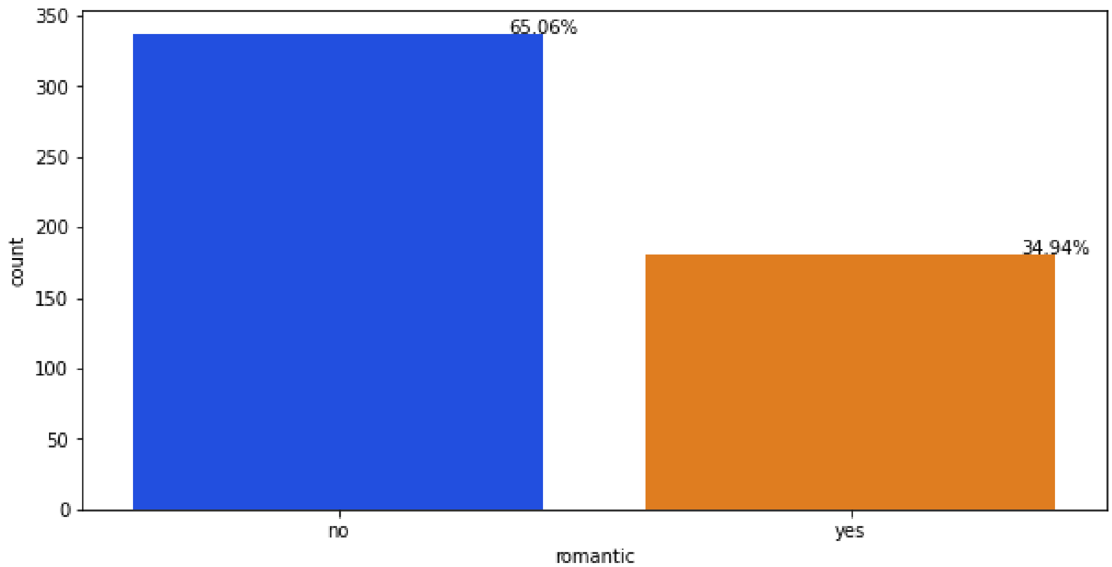

94.21%
5.79%
count
0
100
200
300
400
500
no
yes
paid


49.23%
50.77%
250
200
150
100
50
0
count
no
yes
activities


80.31%
19.69%
400
350
300
250
200
150
100
50
0
count
yes
no
nursery


90.35%
9.65%
400
300
200
100
0
count
yes
no
higher


78.96%
21.04%
400
350
300
250
200
150
100
50
0
count
no
yes
internet



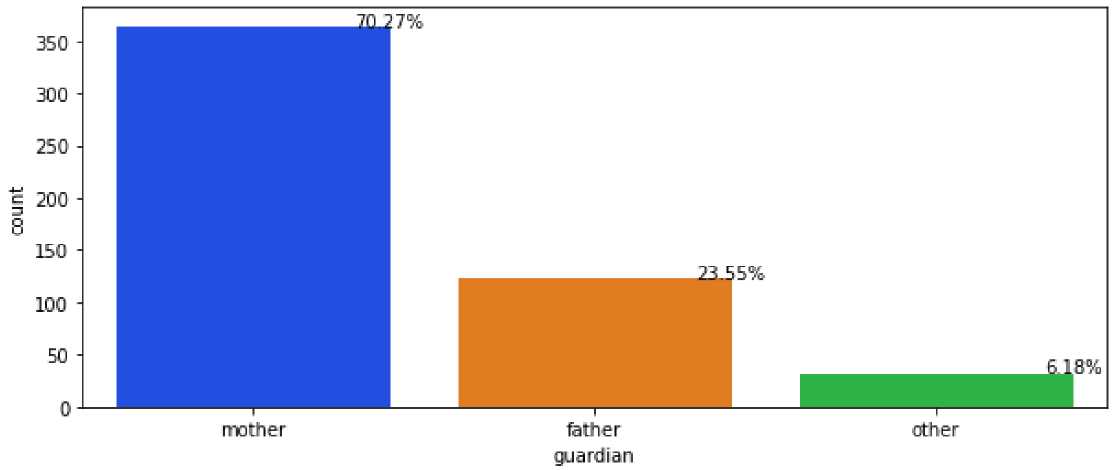
5.2 Nuemrical data plots

[10]:
```
data.hist(figsize=(20,18))
plt.show()
```

6 Detailed Observations

Guardians data - Most students (~70%) have mother as a guardian

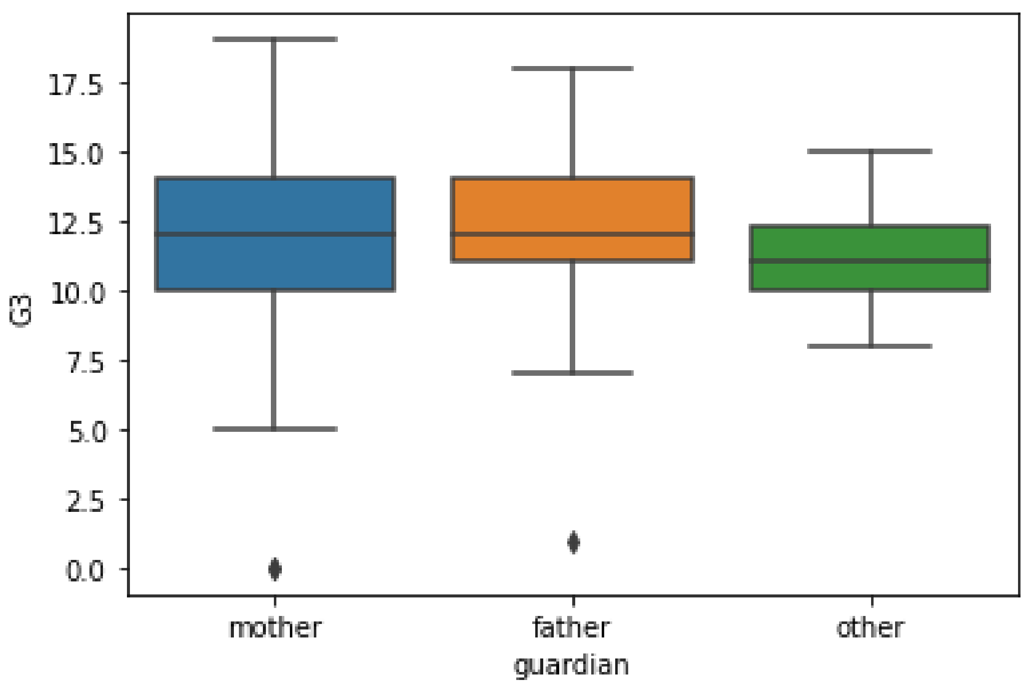
[11]:
```
plt.figure(figsize= (10,4))
graph = sns.countplot(x = 'guardian', data = data, palette= 'bright')
total = float(len(data))
for p in graph.patches:
    pct = f"{(100 * p.get_height()/total):.2f}%"
    x = p.get_x() + p.get_width()
    y = p.get_height()
    graph.annotate(pct, (x, y),ha='center')

plt.show()
```



- Students with 'other' guardian have comparatively lower marks than those with 'mother' or 'father'

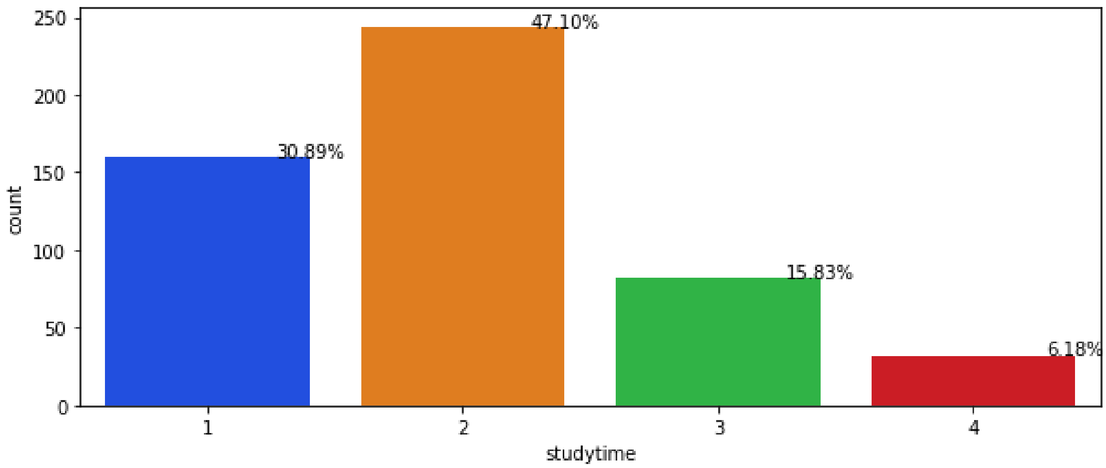
[12]:
```
sns.boxplot(x = 'guardian', y= 'G3', data = data)
```

[12]: <AxesSubplot:xlabel='guardian', ylabel='G3'>



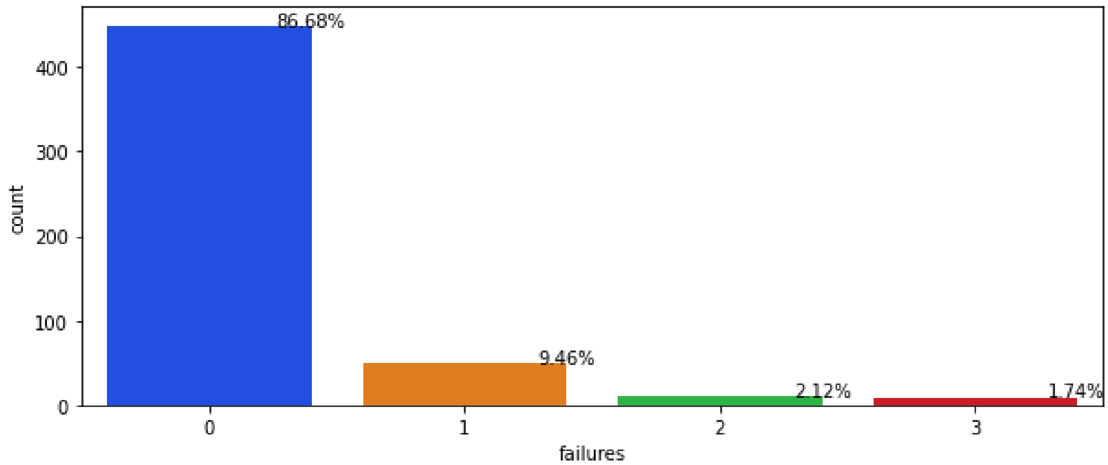
Study hours - Approximately 47% students study for 2-5 hours a week, 30% study for less than 2 hrs

[13]:
```
plt.figure(figsize= (10,4))
graph = sns.countplot(x = 'studytime', data = data, palette= 'bright')
total = float(len(data))
for p in graph.patches:
    pct = f"{(100 * p.get_height()/total):.2f}%"
    x = p.get_x() + p.get_width()
    y = p.get_height()
    graph.annotate(pct, (x, y),ha='center')

plt.show()
```



Failures - About 87% students never failed

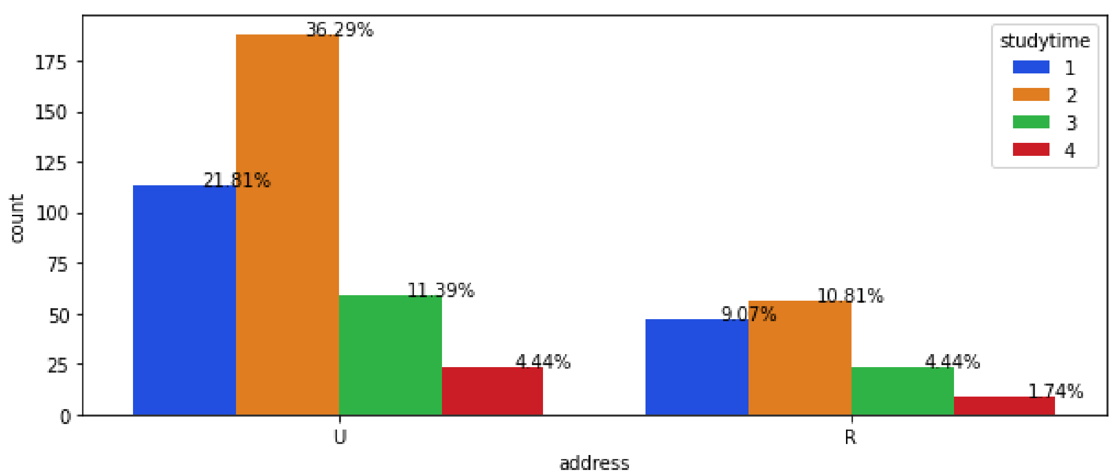
[14]:
```
plt.figure(figsize= (10,4))
graph = sns.countplot(x = 'failures', data = data, palette= 'bright')
total = float(len(data))
for p in graph.patches:
    pct = f"{(100 * p.get_height()/total):.2f}%"
    x = p.get_x() + p.get_width()
    y = p.get_height()
    graph.annotate(pct, (x, y),ha='center')

plt.show()
```



Students in Rural and Urban areas

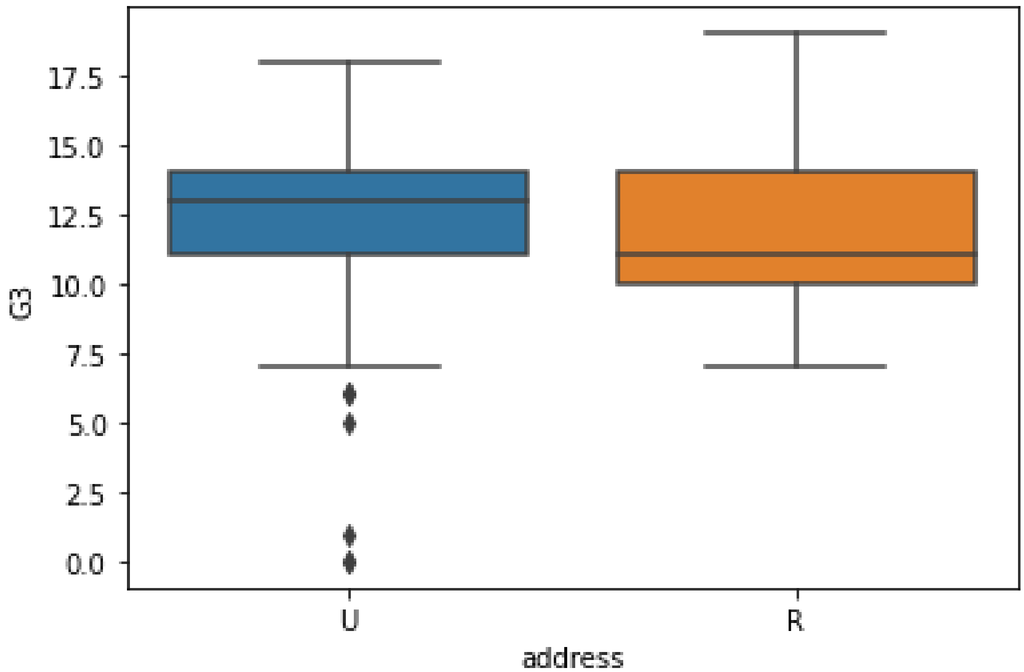
- ~74% students live in Urban areas
- It is clear from the bar plot that more students in urban areas get higher study time
- More students from urabn areas have a commute time of less than 15 mins

[15]:
```
plt.figure(figsize= (10,4))
graph = sns.countplot(x = 'address', data = data, palette= 'bright',hue =␣
↪'studytime')
total = float(len(data))
for p in graph.patches:
    pct = f"{(100 * p.get_height()/total):.2f}%"
    x = p.get_x() + p.get_width()
    y = p.get_height()
    graph.annotate(pct, (x, y),ha='center')

plt.show()
```

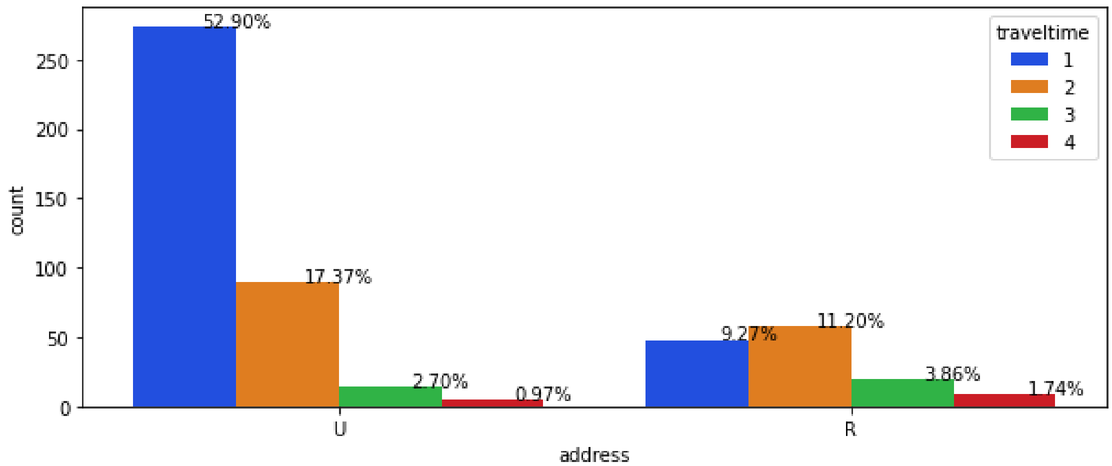


[16]:
```
sns.boxplot(x = 'address', y= 'G3', data = data)
```

[16]: <AxesSubplot:xlabel='address', ylabel='G3'>

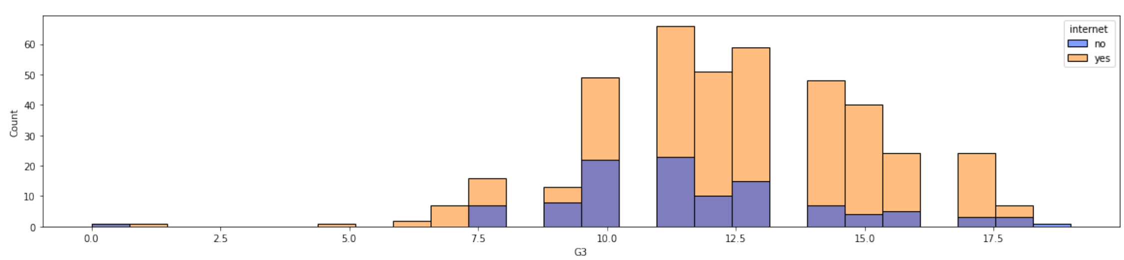


[17]:
```
plt.figure(figsize= (10,4))
graph = sns.countplot(x = 'address', data = data, palette= 'bright',hue =␣
 ↪'traveltime')
total = float(len(data))
for p in graph.patches:
    pct = f"{(100 * p.get_height()/total):.2f}%"
    x = p.get_x() + p.get_width()
    y = p.get_height()
    graph.annotate(pct, (x, y),ha='center')

plt.show()
```



Effect of internet availability on overall grades - More number of students from ‘having internet’ category scored marks closer to the average as compared to those who did not have internet facilities

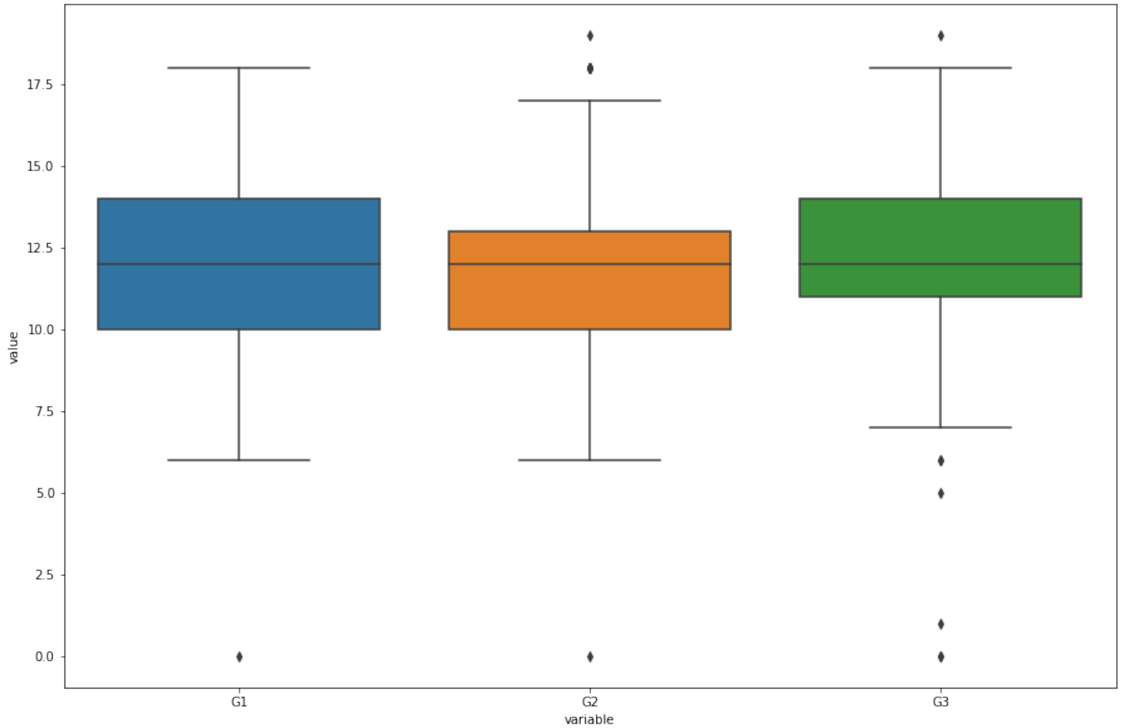
[18]:
```
plt.figure(figsize= (20,4))
graph = sns.histplot(x = 'G3', data = data, palette= 'bright',hue = 'internet')
# total = float(len(data))
# for p in graph.patches:
#     pct = f"{(100 * p.get_height()/total):.2f}%"
#     x = p.get_x() + p.get_width()
#     y = p.get_height()
#     graph.annotate(pct, (x, y),ha='center')

plt.show()
```



Grade variables

- The grade variables present in the above boxplot contain zero grades as well
  - A possible explaination for this can be that zeros represent Dropouts, students with high absences or students that were unable to take an important exam
- The median for G1, G2 and G3 do not seem to differ much. The variation is larger for G1 (term 1 grade)

[19]:
```
plt.figure(figsize=(15,10))
sns.boxplot(x = 'variable', y = 'value', data = pd.melt(data[['G1', 'G2',␣
↪'G3']]))

plt.show()
```



7 Implementing a Predictive Model using Linear Regression

7.1 Importing required libraries

[20]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso, Ridge, RidgeCV, LassoCV, ElasticNet,␣
↪ElasticNetCV, LinearRegression
from sklearn.model_selection import train_test_split
import statsmodels.api
from statsmodels.stats.outliers_influence import variance_inflation_factor as␣
↪vif
```

7.2 Scaling the data to a unit variance

- This is required because the data we have has a wide range

[21]:
```
scaler =  StandardScaler()
```

Defining feature and target variables* - Here we are considering all the numerical variables as features and column 'G3' as target

[22]:
```python
X = data[numerical_vars.index].drop(columns=['Id', 'G3'])
X
```

[22]:
```
     age  Medu  Fedu  traveltime  studytime  failures  famrel  freetime  \
0     18     4     4           2          2         0       4         3
1     17     1     1           1          2         0       5         3
2     15     1     1           1          2         0       4         3
3     15     4     2           1          3         0       3         2
4     16     3     3           1          2         0       4         3
..   ...   ...   ...         ...        ...       ...     ...       ...
513   16     3     1           1          1         0       3         1
514   16     3     2           1          1         0       3         1
515   18     1     1           2          2         0       2         3
516   16     4     4           1          2         0       4         3
517   16     1     2           2          1         0       5         4

     goout  Dalc  Walc  health  absences  G1  G2
0        4     1     1       3         4   0  11
1        3     1     1       3         2   9  11
2        2     2     3       3         6  12  13
3        2     1     1       5         0  14  14
4        2     1     2       5         0  11  13
..     ...   ...   ...     ...       ...  ..  ..
513      3     1     3       1         0   8   6
514      3     1     4       3         2   7   6
515      5     1     4       3         8   9   8
516      3     2     3       2         0  14  16
517      5     1     4       2         0  14  14

[518 rows x 15 columns]
```

[23]:
```python
y = data['G3']
```

[24]:
```python
arr = scaler.fit_transform(X)
```

Finding Variation inflation factor (vif) for each columns Standard acceptable vif value is <10 So we should drop a column which has vif >= 10

[25]:
```python
[vif(arr, i ) for i in range(arr.shape[1])]
```

[25]:
```
[1.253186067218731,
 1.8462728732526232,
 1.7860498706067724,
 1.1079935899811844,
 1.1436681085552456,
```

```
 1.3025671509758214,
 1.1067956037568312,
 1.1617961028014965,
 1.3746667933174581,
 1.7330608576284452,
 1.9680392624900334,
 1.0775006183088571,
 1.1375848490670828,
 4.352149038245545,
 4.5303473211085725]
```

[26]:
```
vif_df = pd.DataFrame()
```

[27]:
```
vif_df['features'] = X.columns
vif_df['vif'] = [vif(arr, i ) for i in range(arr.shape[1])]
```

[28]:
```
vif_df ## drop columns where vif is greater than or equal to 10
```

[28]:

	features	vif
0	age	1.253186
1	Medu	1.846273
2	Fedu	1.786050
3	traveltime	1.107994
4	studytime	1.143668
5	failures	1.302567
6	famrel	1.106796
7	freetime	1.161796
8	goout	1.374667
9	Dalc	1.733061
10	Walc	1.968039
11	health	1.077501
12	absences	1.137585
13	G1	4.352149
14	G2	4.530347

7.3 Splitting the dataset into train and test dataset

[54]:
```
X_train, X_test, y_train, y_test =  train_test_split(arr,y, test_size=0.15,␣
↪random_state=500)
```

Note : We can tune the random state parameter to get better results

7.4 Implementing linear regression model on train dataset

[55]:
```
linear= LinearRegression()
linear.fit(X_train, y_train)
```

[55]:
```
LinearRegression()
```

[68]:
```
linear.score(X_test, y_test) ##checking score
```

[68]:
```
0.91286177751691
```

- We have obtained an accuracy of 91.29%

7.4.1 Testing the prediction model with manually prepared input

[33]:
```
X.iloc[0]
```

[33]:
```
age           18
Medu           4
Fedu           4
traveltime     2
studytime      2
failures       0
famrel         4
freetime       3
goout          4
Dalc           1
Walc           1
health         3
absences       4
G1             0
G2            11
Name: 0, dtype: int64
```

[34]:
```
sample = [[17,3,3,3,7,1,3,5,2,1,1,2,8,1,16]]
```

[35]:
```
linear.predict(sample)
```

[35]:
```
array([48.36650111])
```

- We are getting result > 20 because we have used the scaled data as our input.
- We have to transform the sample input again to obtain the desired result

[36]:
```
test1 = scaler.transform(sample)
```

[37]:
```
linear.predict(test1)
```

[37]:
```
array([14.00857774])
```

- Overall grade for the manually prepared data is 14.00857774

7.5 Running the prediction model on an untested dataset

[38]:
```
final_test_df = pd.read_csv('data/stu_test.csv')
```

[39]:
```
X_test_final = final_test_df[numerical_vars.index].drop(columns=['Id', 'G3'])
```

[40]:
```
y_test_pred = final_test_df['G3']
```

[41]:
```
for i in range(len(y_test_pred)):
    y_test_pred.iloc[i] = linear.predict(scaler.transform([X_test_final.iloc[i].
 ↪values]))
```

[42]:
```
y_test_pred
```

[42]:
```
0       6.194001
1       8.264763
2       8.644678
3      10.440467
4       8.546643
         …
126    11.297710
127    15.654319
128    12.952776
129    10.589719
130    11.918108
Name: G3, Length: 131, dtype: float64
```