Importing Libraries and datasets

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

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Ridge
R = Ridge()

from sklearn.ensemble import RandomForestRegressor
RF = RandomForestRegressor()

from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

mat = pd.read_csv("/content/student-mat.csv")
por = pd.read_csv("/content/student-por.csv")

data = pd.concat([mat,por])
```

Exploratory Data Analysis

```
data.head()
```

```
  school sex  age address famsize Pstatus  ...  Walc  health absences
G1  G2  G3
0     GP   F   18       U     GT3       A  ...     1       3        6
5   6   6
1     GP   F   17       U     GT3       T  ...     1       3        4
5   5   6
2     GP   F   15       U     LE3       T  ...     3       3       10
7   8  10
3     GP   F   15       U     GT3       T  ...     1       5        2
15  14  15
4     GP   F   16       U     GT3       T  ...     2       5        4
6  10  10

[5 rows x 33 columns]
```

```
data.tail()
```

```
    school sex  age address famsize Pstatus  ...  Walc  health
absences  G1  G2  G3
644     MS   F   19       R     GT3       T  ...     2       5
4  10  11  10
645     MS   F   18       U     LE3       T  ...     1       1
4  15  15  16
646     MS   F   18       U     GT3       T  ...     1       5
6  11  12   9
647     MS   M   17       U     LE3       T  ...     4       2
6  10  10  10
648     MS   M   18       R     LE3       T  ...     4       5
4  10  11  11

[5 rows x 33 columns]
```

```
data.shape
```

```
(1044, 33)
```

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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1044 entries, 0 to 648
Data columns (total 33 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        1044 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
 26  Dalc        1044 non-null   int64
 27  Walc        1044 non-null   int64
 28  health      1044 non-null   int64
 29  absences    1044 non-null   int64
 30  G1          1044 non-null   int64
 31  G2          1044 non-null   int64
 32  G3          1044 non-null   int64
dtypes: int64(16), object(17)
memory usage: 277.3+ KB
```

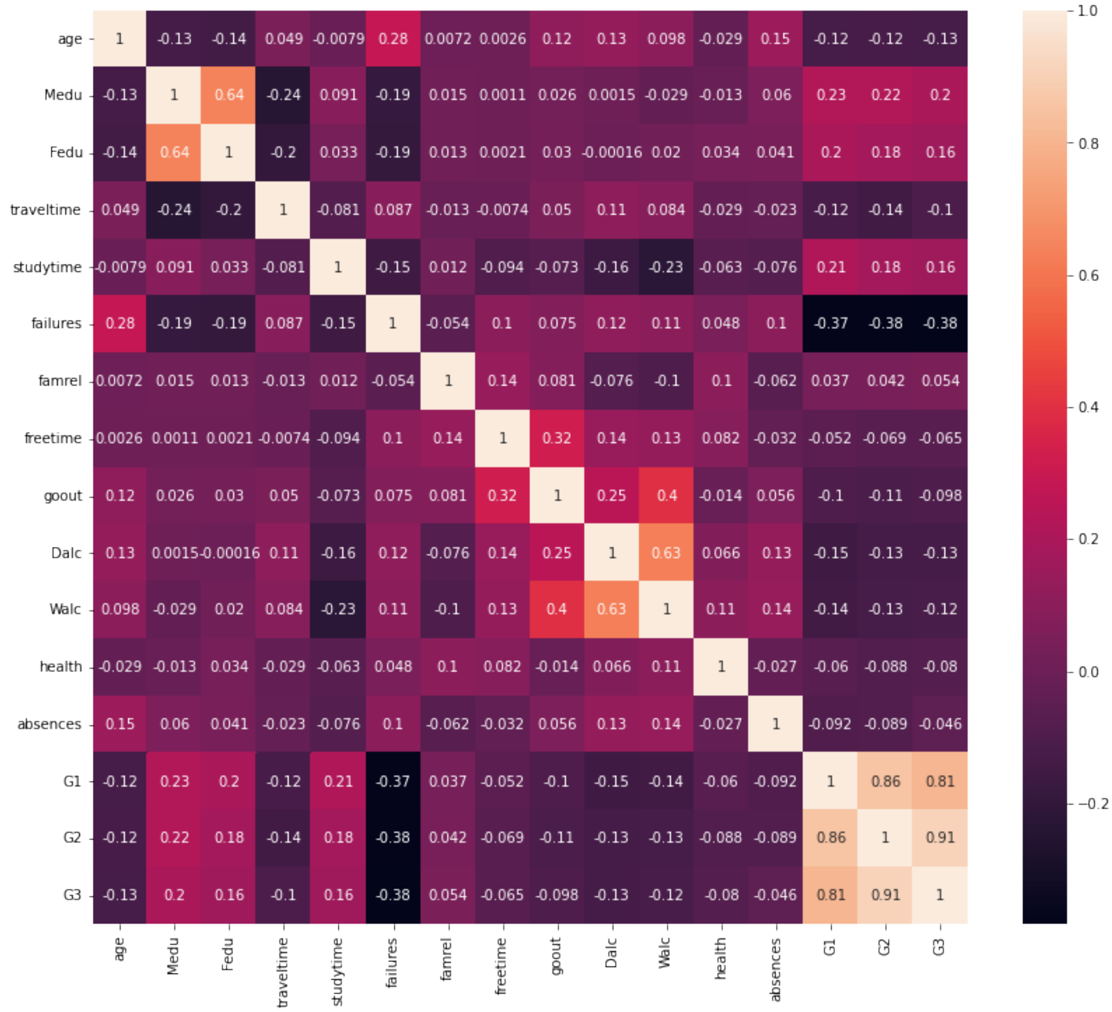
```
data.describe()
```

```
               age         Medu  ...           G2           G3
count  1044.000000  1044.000000  ...  1044.000000  1044.000000
mean     16.726054     2.603448  ...    11.246169    11.341954
std       1.239975     1.124907  ...     3.285071     3.864796
min      15.000000     0.000000  ...     0.000000     0.000000
25%      16.000000     2.000000  ...     9.000000    10.000000
50%      17.000000     3.000000  ...    11.000000    11.000000
75%      18.000000     4.000000  ...    13.000000    14.000000
max      22.000000     4.000000  ...    19.000000    20.000000

[8 rows x 16 columns]
```

Visualizing the data

Correlation among variables

```
plt.figure(figsize=(14, 12))
sns.heatmap(data.corr(), annot=True)
plt.show()
```



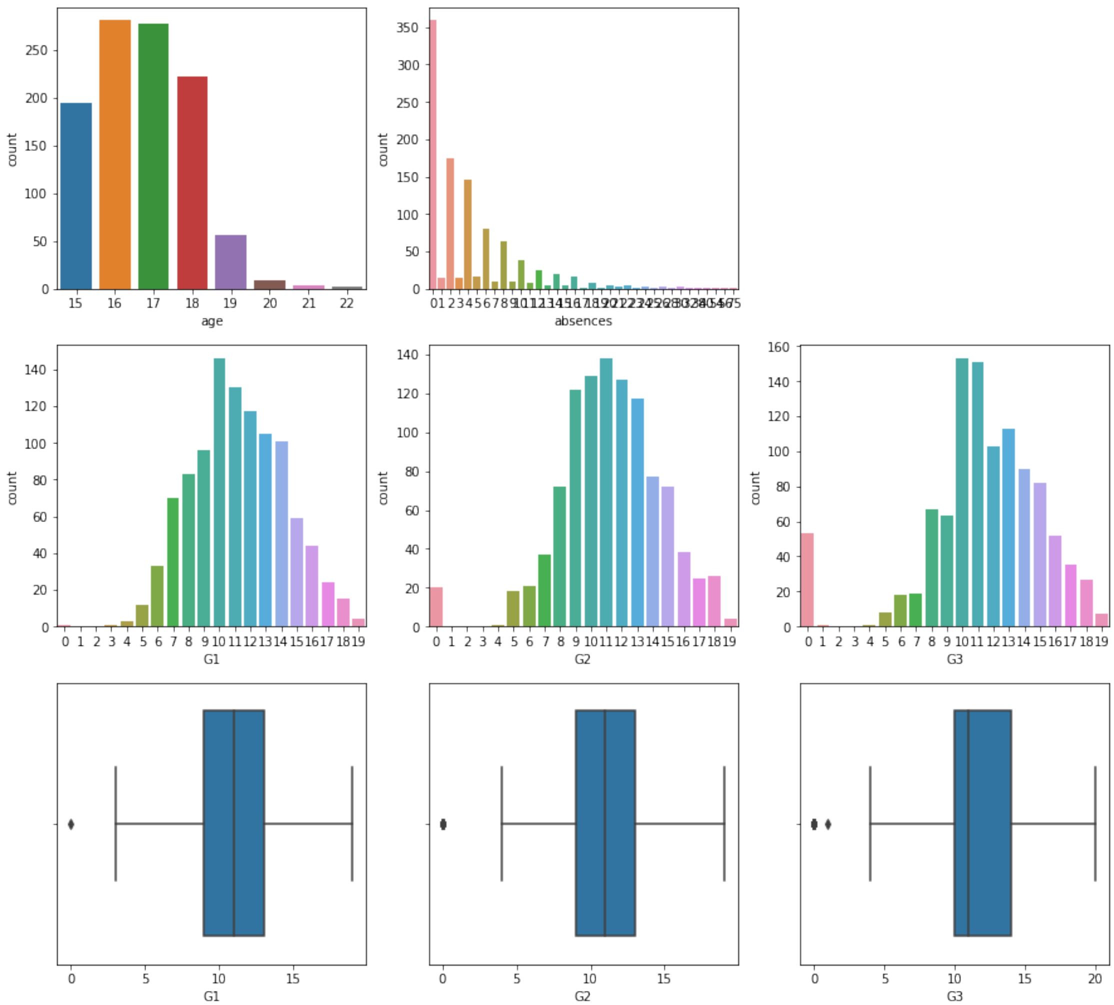
Dividing features into continous, ordinal, nominal and binary

```
cont = ["age", "absences", "G1", "G2", "G3"]
ordin = ["Medu", "Fedu", "traveltime", "studytime", "failures",
"famrel", "freetime", "goout", "Dalc", "Walc", "health"]
nom = ["Mjob", "Fjob", "reason", "guardian"]
binary = ["school", "sex", "address", "famsize", "Pstatus",
"schoolsup", "famsup", "paid", "activities", "nursery", "higher",
"internet", "romantic"]
```

Distribution of continous variables

```
import matplotlib.gridspec as gs
fig = plt.figure(figsize = (15,14))
g = gs.GridSpec(nrows = 3, ncols = 3, figure = fig)
i = 0

ax1 = plt.subplot(g[0,0])
ax1 = sns.countplot(data[cont[0]])

ax2 = plt.subplot(g[0,1])
```

```
ax2 = sns.countplot(data[cont[1]])

rg = list(range(0,20))
for feature in cont[2:]:
    ax = plt.subplot(g[1,i])
    ax = sns.countplot(data[feature], order = rg)
    i = i+1
i = 0
for feature in cont[2:]:
    ax = plt.subplot(g[2,i])
    ax = sns.boxplot(data = data, x = data[feature], order = rg)
    i = i+1
```

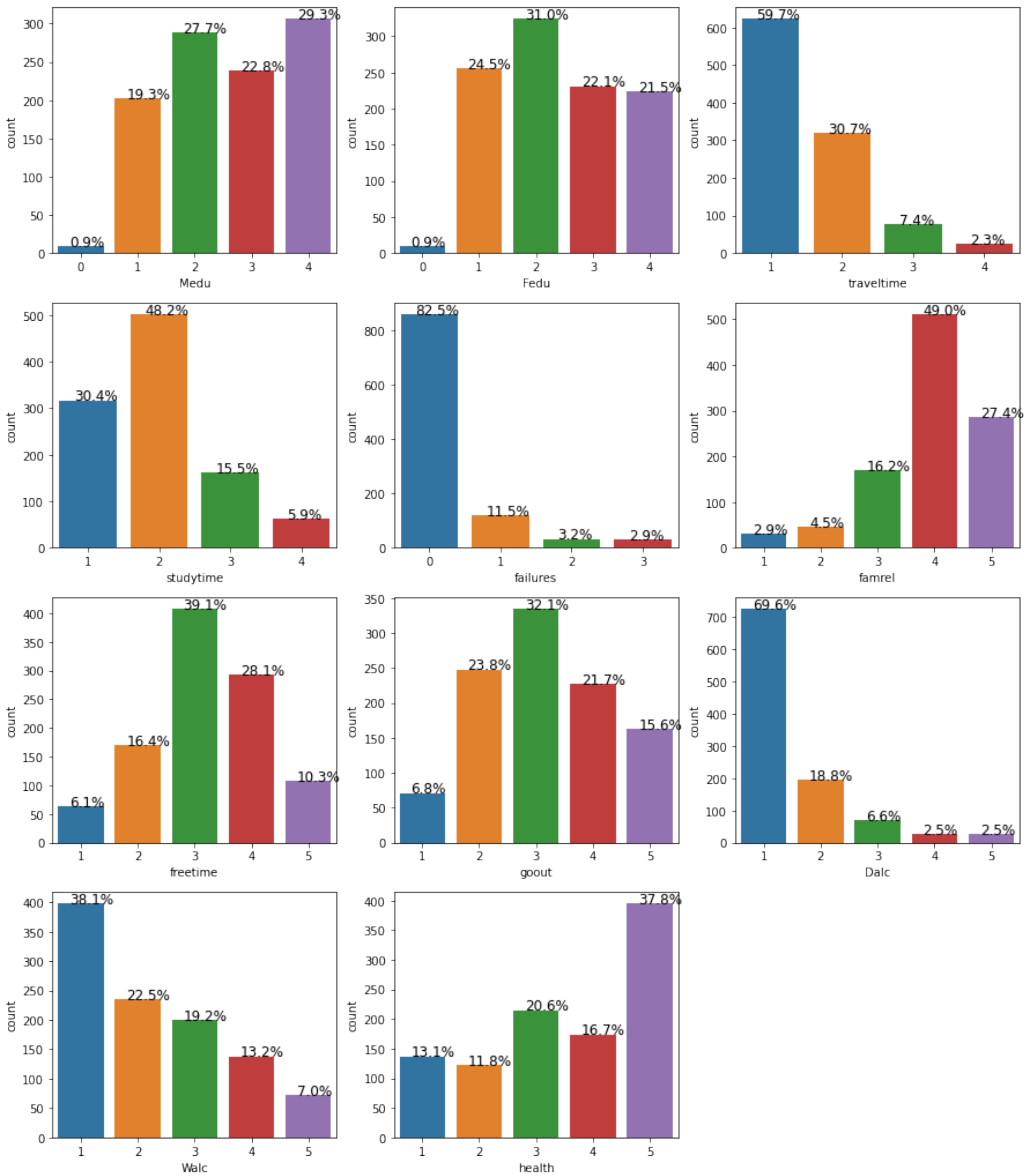



```
def with_hue(plot, feature, Number_of_categories, hue_categories):
    a = [p.get_height() for p in plot.patches]
    patch = [p for p in plot.patches]
    for i in range(Number_of_categories):
        total = feature.value_counts().values[i]
        for j in range(hue_categories):
            percentage = '{:.1f}%'.format(100 *
```

```
a[(j*Number_of_categories + i)]/total)
            x = patch[(j*Number_of_categories + i)].get_x() +
patch[(j*Number_of_categories + i)].get_width() / 2 - 0.15
            y = patch[(j*Number_of_categories + i)].get_y() +
patch[(j*Number_of_categories + i)].get_height()
            plot.annotate(percentage, (x, y), size = 12)
   # plt.show()

def without_hue(plot, feature):
    total = len(feature)
    for p in plot.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total)
        x = p.get_x() + p.get_width() / 2 - 0.2
        y = p.get_y() + p.get_height()
        plot.annotate(percentage, (x, y), size = 12)
    #plt.show()
```

Distribution of ordinal variables

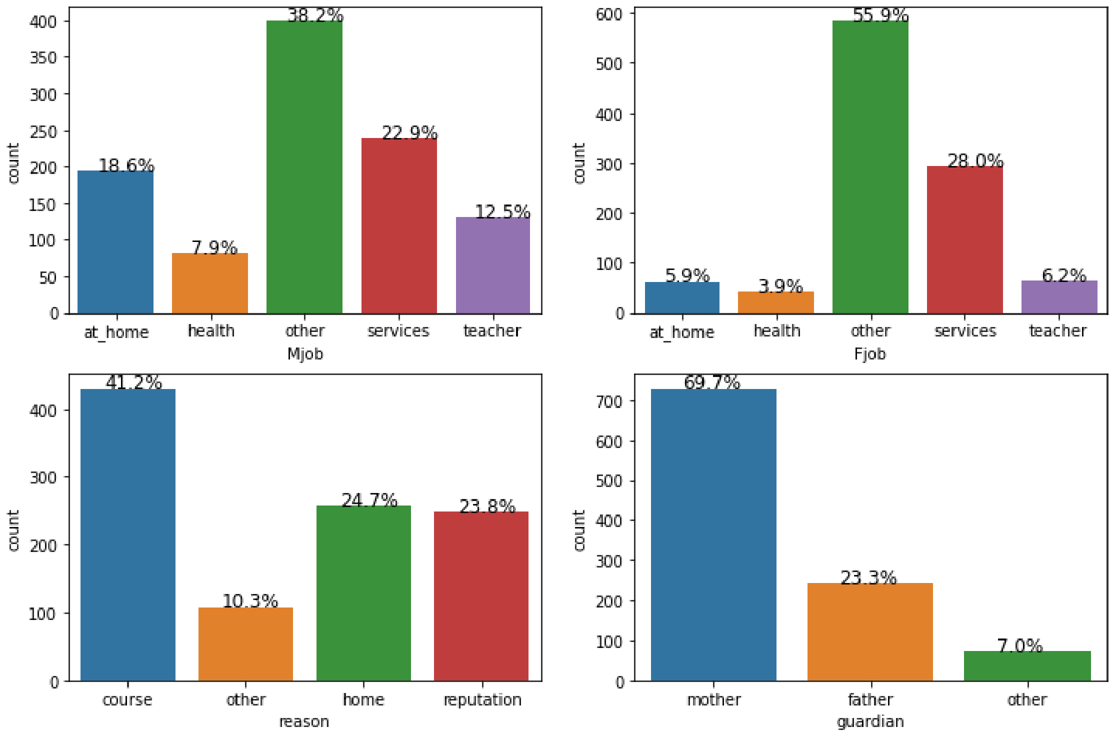
```
fig = plt.figure(figsize = (15,18))
g = gs.GridSpec(nrows = 4, ncols = 3, figure = fig)
i = 0
j = 0
for feature in ordin:
    if j == 3:
        i = i + 1
        j = 0
    ax1 = plt.subplot(g[i,j])
    ax1 = sns.countplot(data[feature])
    without_hue(ax1,data[feature])
    j = j + 1
```



Distribution of nominal variables

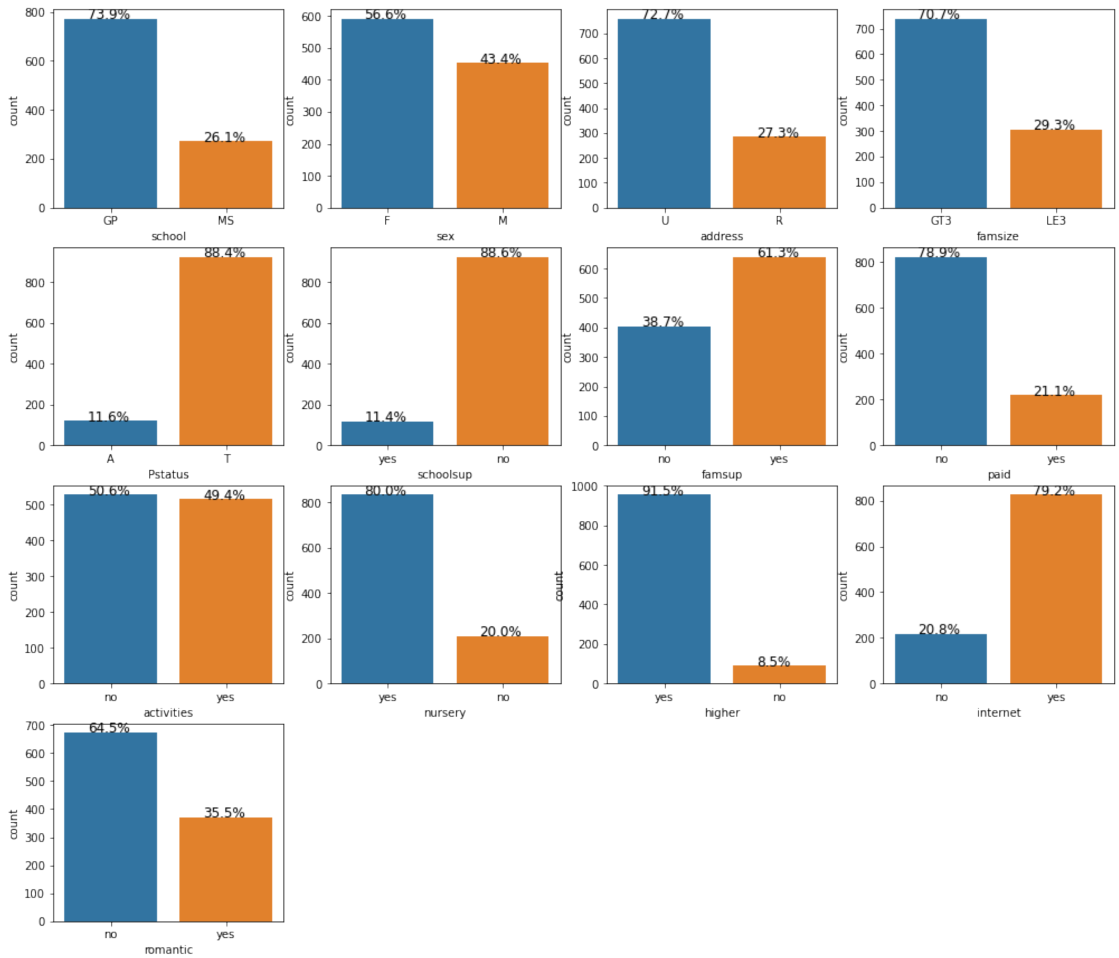
```
fig = plt.figure(figsize = (12,8))
g = gs.GridSpec(nrows = 2, ncols = 2, figure = fig)

ax1 = plt.subplot(g[0,0])
ax1 = sns.countplot(data[nom[0]])
without_hue(ax1,data[nom[0]])

ax2 = plt.subplot(g[0,1])
ax2 = sns.countplot(data[nom[1]], order = ["at_home", "health",
"other", "services", "teacher"])
```

```
without_hue(ax2,data[nom[1]])

ax3 = plt.subplot(g[1,0])
ax3 = sns.countplot(data[nom[2]])
without_hue(ax3,data[nom[2]])

ax4 = plt.subplot(g[1,1])
ax4 = sns.countplot(data[nom[3]])
without_hue(ax4,data[nom[3]])
```



Distribution of binary variables

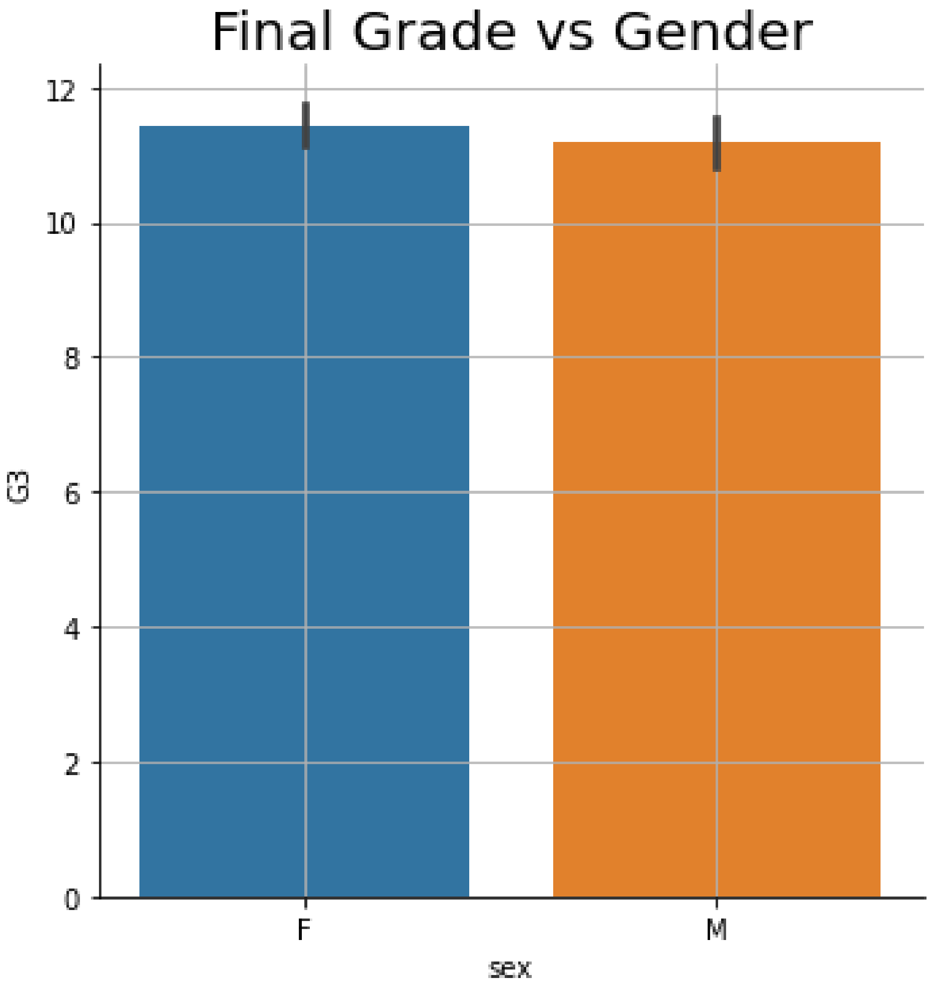
```
fig = plt.figure(figsize = (17,15))
g = gs.GridSpec(nrows = 4, ncols = 4, figure = fig)
i = 0
j = 0
for feature in binary:
    if j == 4:
        i = i + 1
        j = 0
    ax1 = plt.subplot(g[i,j])
    ax1 = sns.countplot(data[feature])
    without_hue(ax1,data[feature])
    j = j + 1
```



Finding students' performance on the basis of -

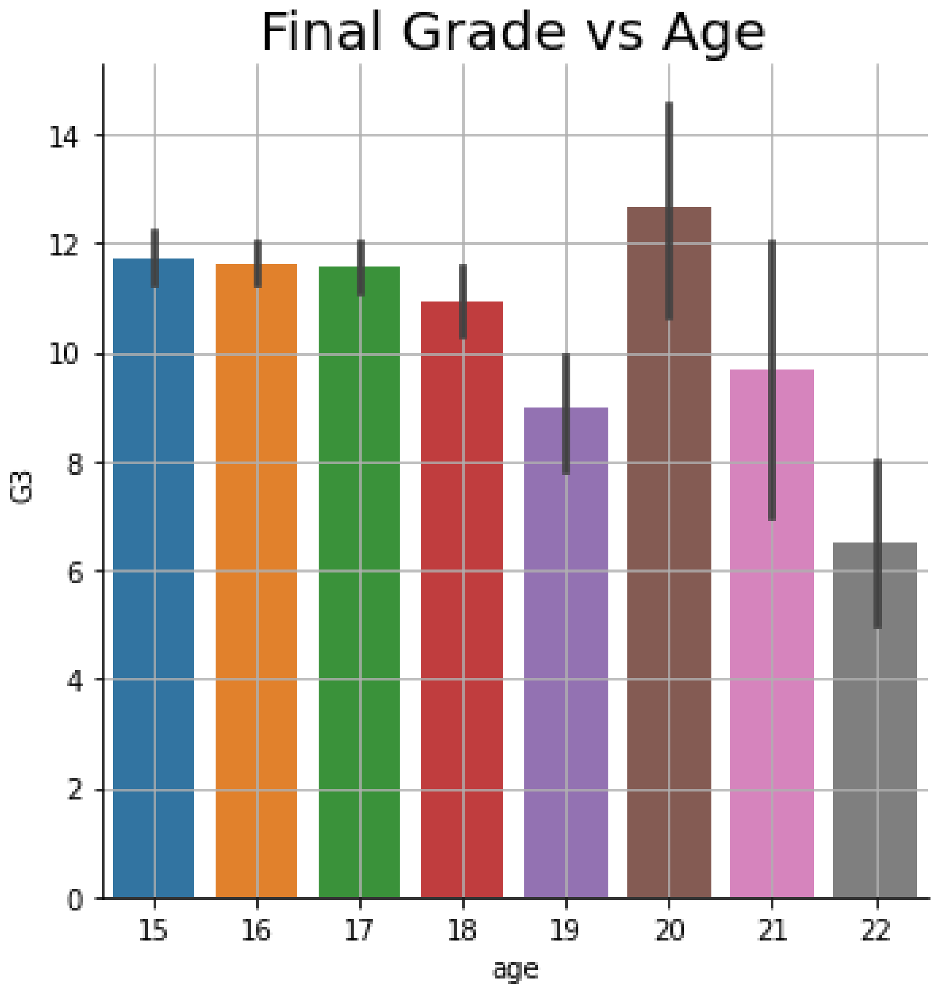
Gender

```
sns.catplot(y='G3',x='sex',data=data,kind='bar')
plt.title('Final Grade vs Gender',size=20)
plt.grid()
```



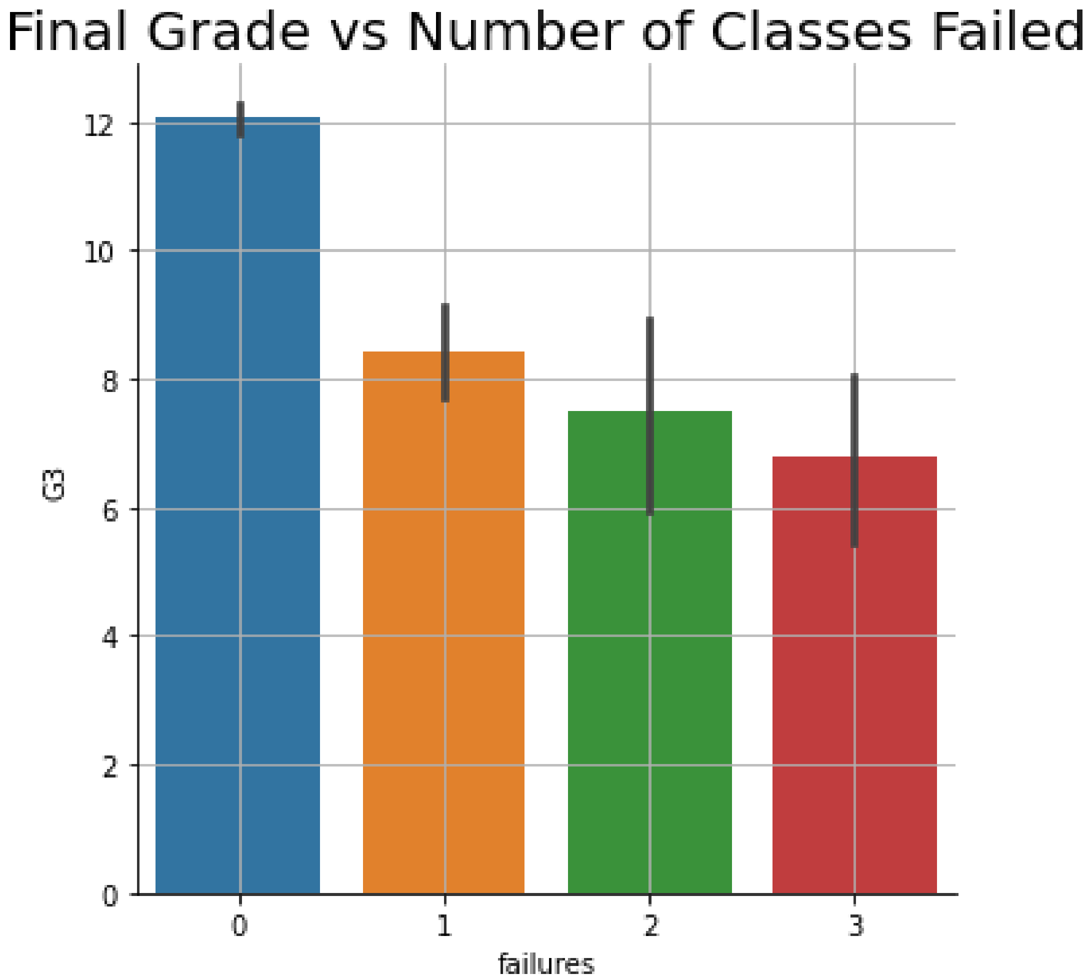
Age

```
sns.catplot(y='G3',x='age',data=data,kind='bar')
plt.title('Final Grade vs Age',size=20)
plt.grid()
```



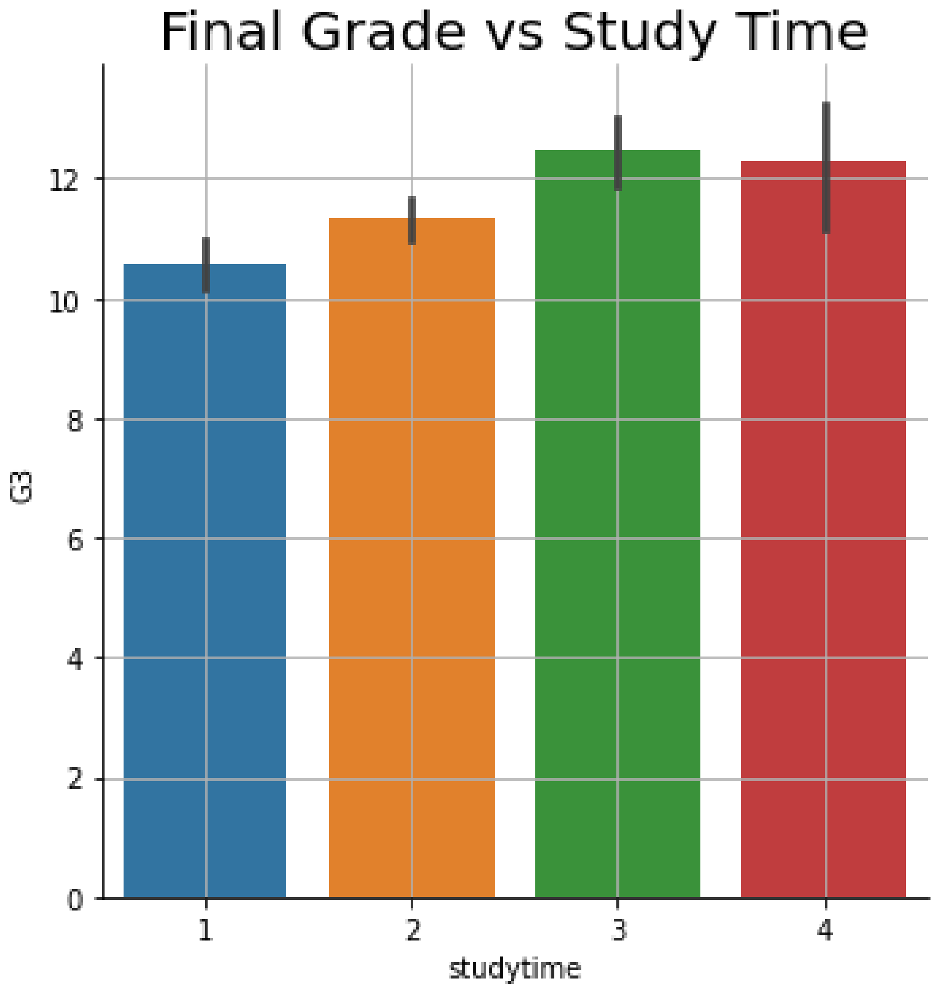
No of classes failed

```
sns.catplot(y='G3',x='failures',data=data,kind='bar')
plt.title('Final Grade vs Number of Classes Failed',size=20)
plt.grid()
```



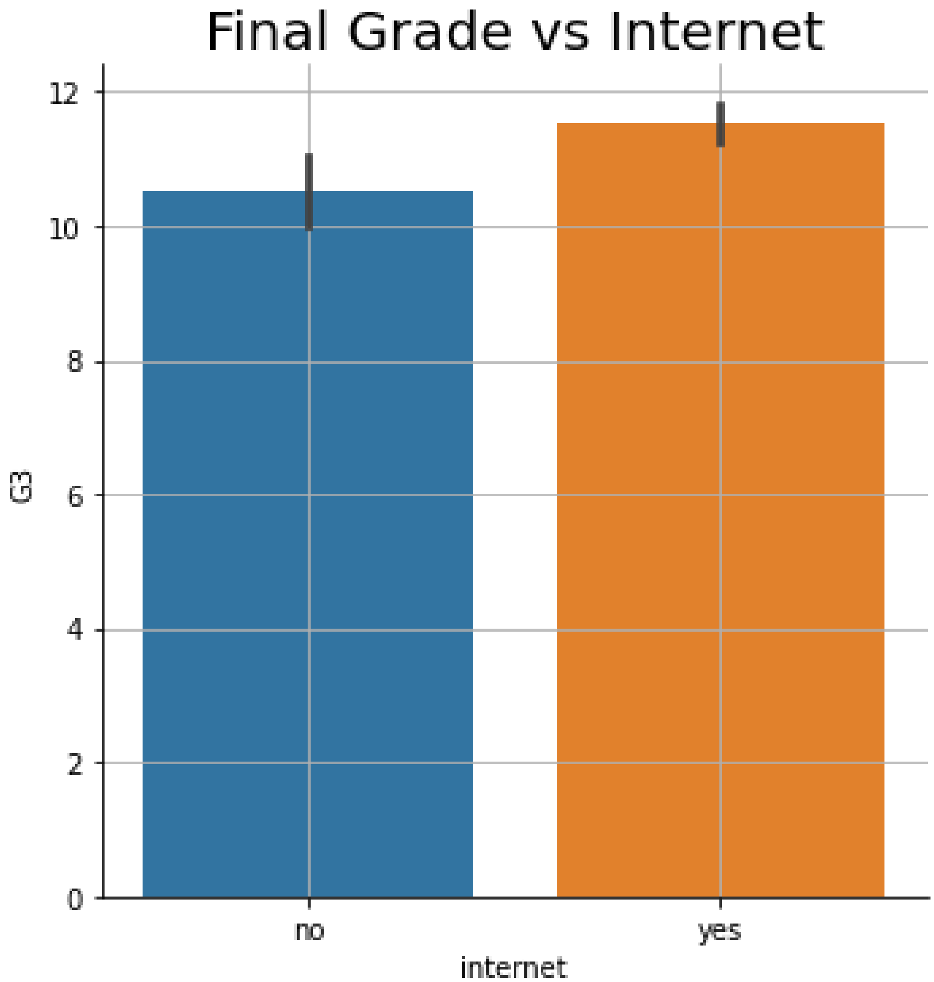
Study time

```
sns.catplot(y='G3',x='studytime',data=data,kind='bar')
plt.title('Final Grade vs Study Time',size=20)
plt.grid()
```



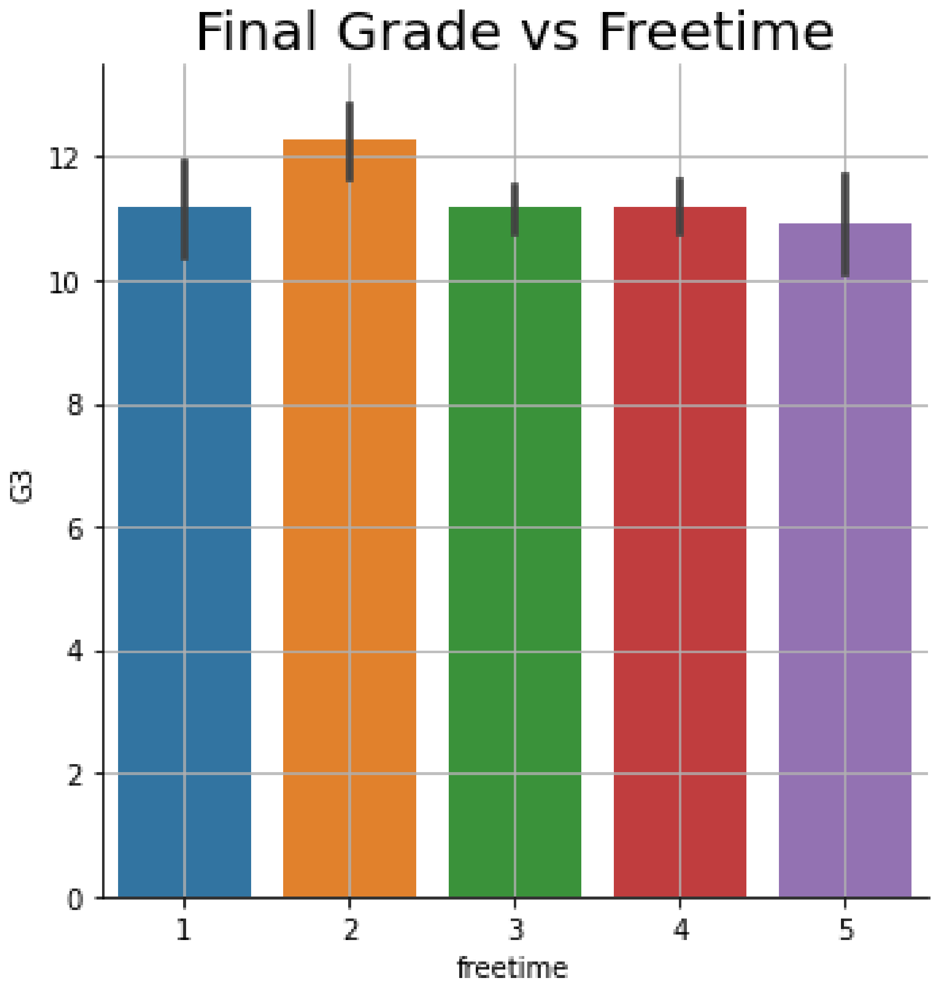
Internet connectivity

```
sns.catplot(y='G3',x='internet',data=data,kind='bar')
plt.title('Final Grade vs Internet',size=20)
plt.grid()
```



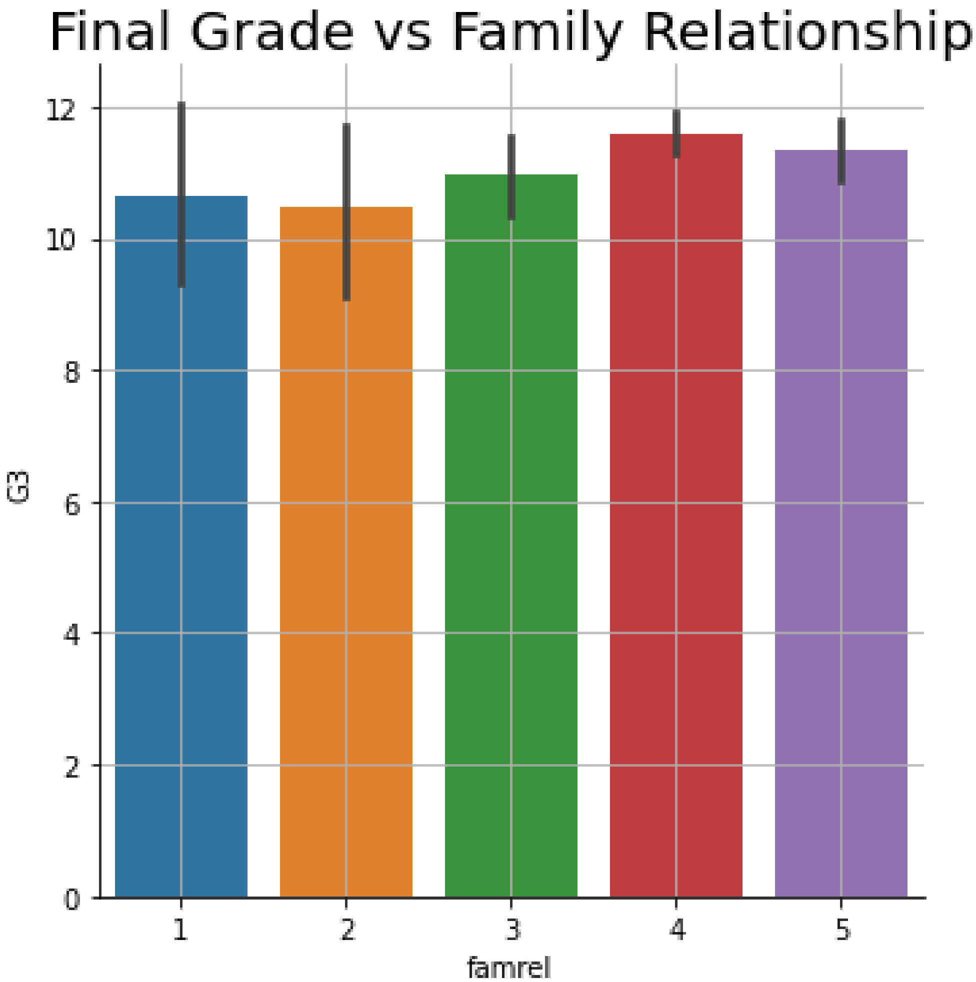
Leisure period

```
sns.catplot(y='G3',x='freetime',data=data,kind='bar')
plt.title('Final Grade vs Freetime',size=20)
plt.grid()
```



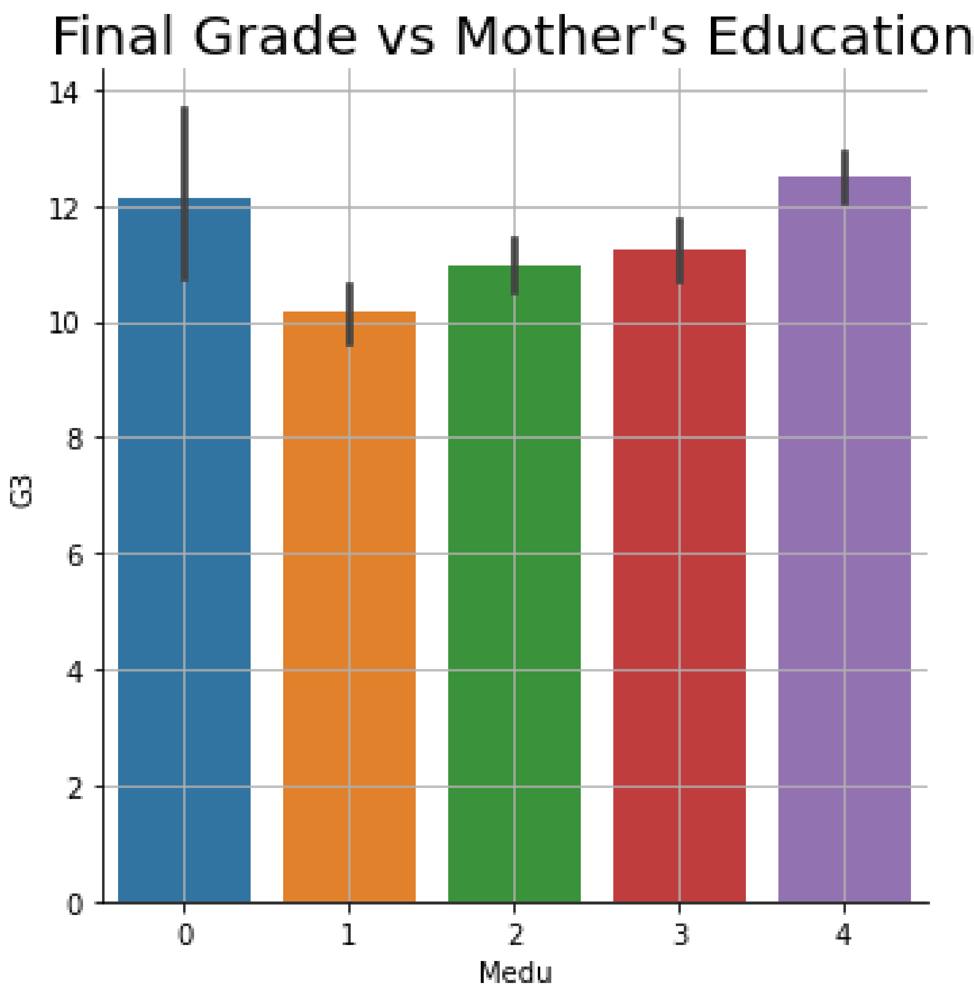
Quality of family relationship

```
sns.catplot(y='G3',x='famrel',data=data,kind='bar')
plt.title('Final Grade vs Family Relationship',size=20)
plt.grid()
```



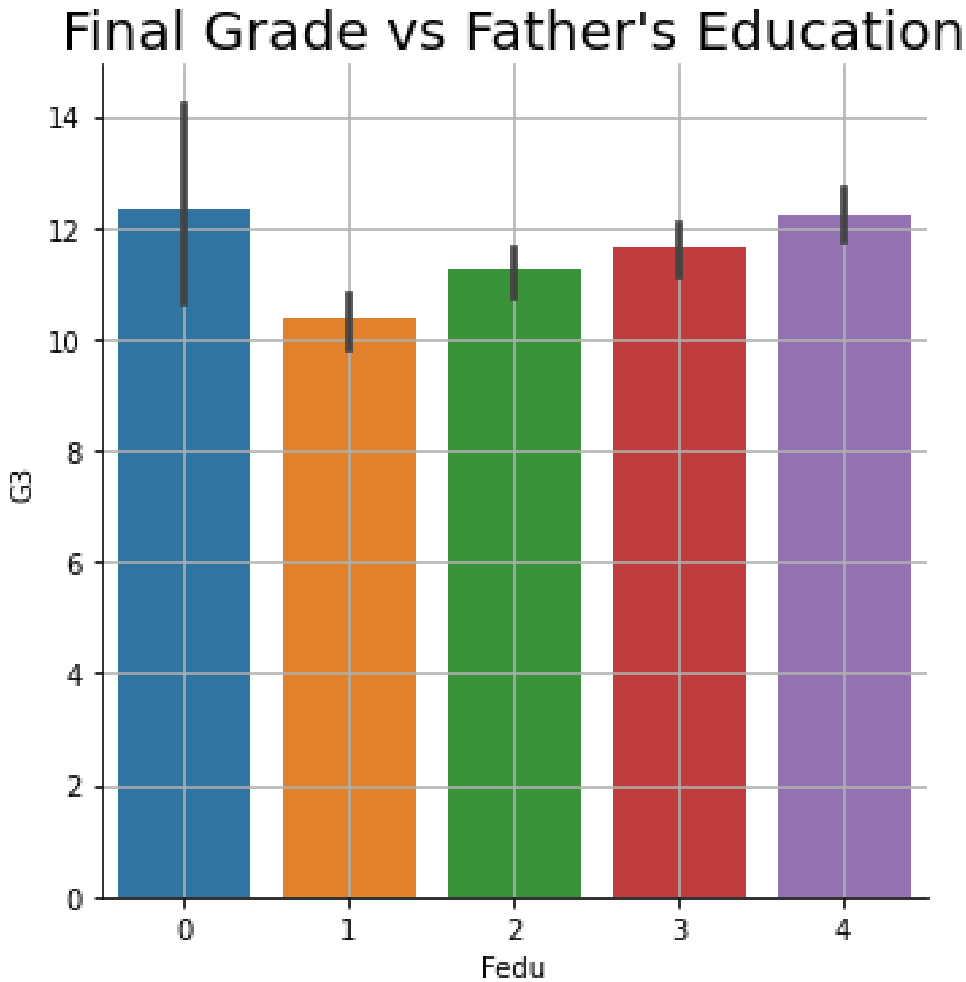
Mother's education

```
sns.catplot(y='G3',x='Medu',data=data,kind='bar')
plt.title('Final Grade vs Mother\'s Education',size=20)
plt.grid()
```



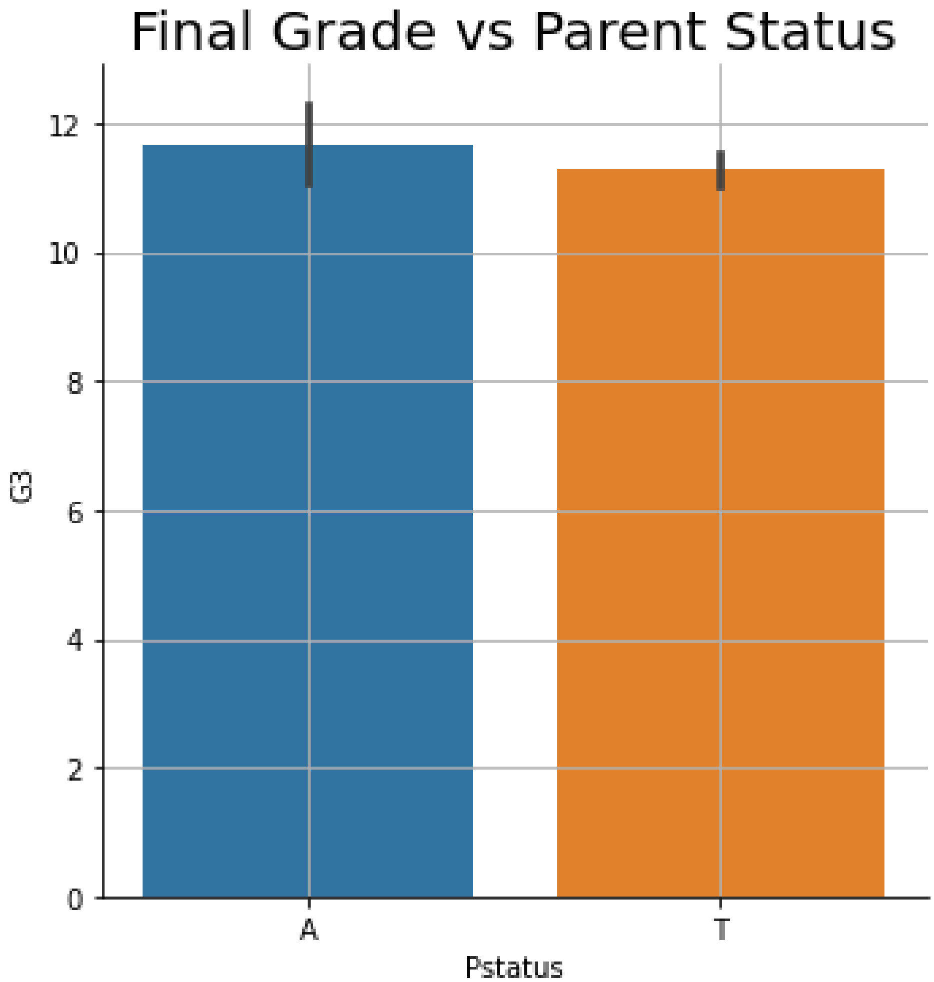
Father's education

```
sns.catplot(y='G3',x='Fedu',data=data,kind='bar')
plt.title('Final Grade vs Father\'s Education',size=20)
plt.grid()
```



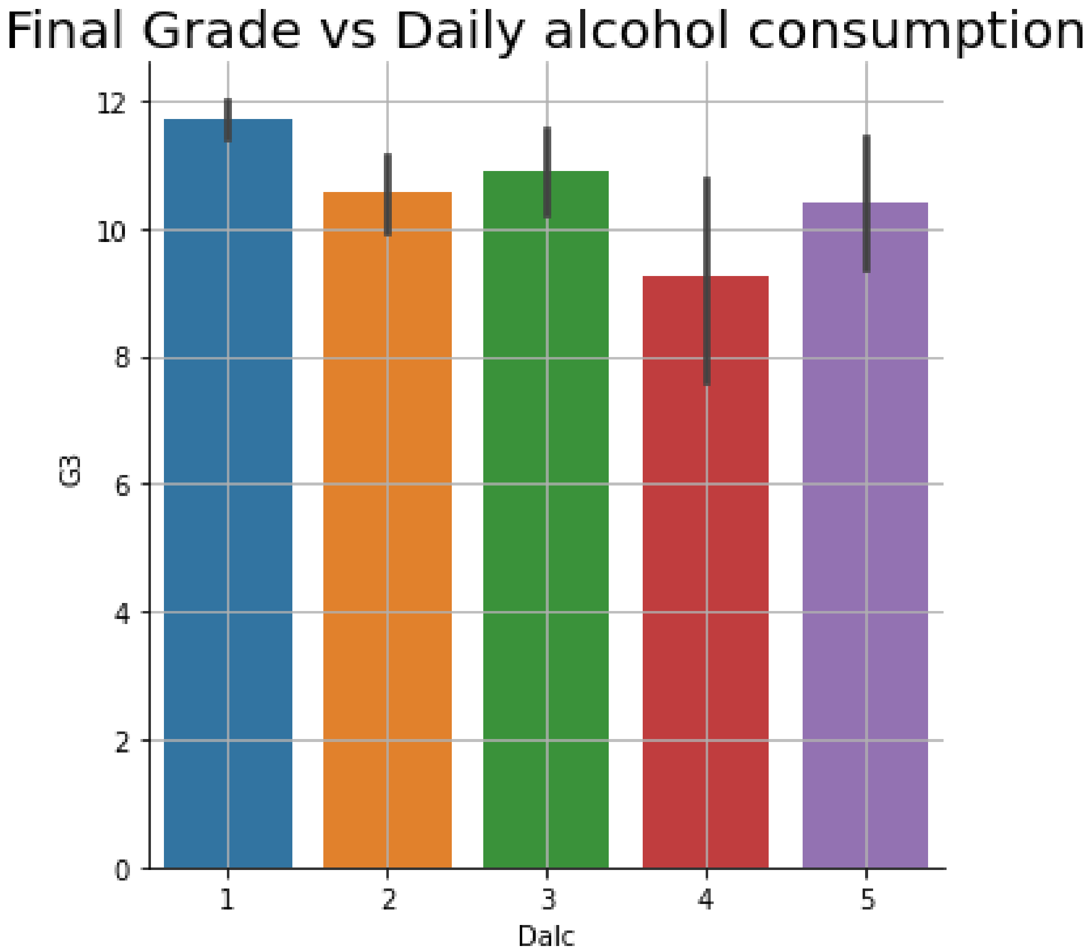
Parenting Status

```
sns.catplot(y='G3',x='Pstatus',data=data,kind='bar')
plt.title('Final Grade vs Parent Status',size=20)
plt.grid()
```



Daily alcohol consumption

```
sns.catplot(y='G3',x='Dalc',data=data,kind='bar')
plt.title('Final Grade vs Daily alcohol consumption',size=20)
plt.grid()
```



There are several factors which are also important for building and developing the model and generating the results obtained, which we have proceeded further.

Data Pre-preprocessing

Checking missing values

```
data.isnull().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          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
```

Encoding

```
data.dtypes
```

```
school        object
sex           object
age            int64
address       object
famsize       object
Pstatus       object
Medu           int64
Fedu           int64
Mjob          object
Fjob          object
reason        object
guardian      object
traveltime     int64
studytime      int64
failures       int64
schoolsup     object
famsup        object
paid          object
activities    object
nursery       object
higher        object
internet      object
romantic      object
```

```
famrel         int64
freetime       int64
goout          int64
Dalc           int64
Walc           int64
health         int64
absences       int64
G1             int64
G2             int64
G3             int64
dtype: object
```

```
nonnumeric_columns = [data.columns[index] for index, dtype in
enumerate(data.dtypes) if dtype == 'object']
nonnumeric_columns
```

```
['school',
 'sex',
 'address',
 'famsize',
 'Pstatus',
 'Mjob',
 'Fjob',
 'reason',
 'guardian',
 'schoolsup',
 'famsup',
 'paid',
 'activities',
 'nursery',
 'higher',
 'internet',
 'romantic']
```

```
for column in nonnumeric_columns:
    print(f"{column}: {data[column].unique()}")
```

```
school: ['GP' 'MS']
sex: ['F' 'M']
address: ['U' 'R']
famsize: ['GT3' 'LE3']
Pstatus: ['A' 'T']
Mjob: ['at_home' 'health' 'other' 'services' 'teacher']
Fjob: ['teacher' 'other' 'services' 'health' 'at_home']
reason: ['course' 'other' 'home' 'reputation']
guardian: ['mother' 'father' 'other']
schoolsup: ['yes' 'no']
famsup: ['no' 'yes']
paid: ['no' 'yes']
activities: ['no' 'yes']
nursery: ['yes' 'no']
```

```
higher: ['yes' 'no']
internet: ['no' 'yes']
romantic: ['no' 'yes']
```

```
data['Mjob'] = data['Mjob'].apply(lambda x: "m_" + x)
data['Fjob'] = data['Fjob'].apply(lambda x: "f_" + x)
data['reason'] = data['reason'].apply(lambda x: "r_" + x)
data['guardian'] = data['guardian'].apply(lambda x: "g_" + x)

data
```

```
    school sex  age address famsize Pstatus  ...  Walc  health
absences  G1  G2  G3
0       GP   F   18       U     GT3       A  ...     1       3
6   5   6   6
1       GP   F   17       U     GT3       T  ...     1       3
4   5   5   6
2       GP   F   15       U     LE3       T  ...     3       3
10   7   8  10
3       GP   F   15       U     GT3       T  ...     1       5
2  15  14  15
4       GP   F   16       U     GT3       T  ...     2       5
4   6  10  10
..     ...  ..  ...     ...     ...     ...  ...   ...     ...      ..
.  ..  ..  ..
644     MS   F   19       R     GT3       T  ...     2       5
4  10  11  10
645     MS   F   18       U     LE3       T  ...     1       1
4  15  15  16
646     MS   F   18       U     GT3       T  ...     1       5
6  11  12   9
647     MS   M   17       U     LE3       T  ...     4       2
6  10  10  10
648     MS   M   18       R     LE3       T  ...     4       5
4  10  11  11

[1044 rows x 33 columns]
```

```
dummies = pd.concat([pd.get_dummies(data['Mjob']),
                     pd.get_dummies(data['Fjob']),
                     pd.get_dummies(data['reason']),
                     pd.get_dummies(data['guardian'])],
                     axis=1)
dummies
```

```
     m_at_home  m_health  m_other  ...  g_father  g_mother  g_other
0            1         0        0  ...         0         1        0
1            1         0        0  ...         1         0        0
2            1         0        0  ...         0         1        0
3            0         1        0  ...         0         1        0
4            0         0        1  ...         1         0        0
```

```
..          ...       ...       ...  ...        ...       ...      ...
644           0         0         0  ...          0         1        0
645           0         0         0  ...          0         1        0
646           0         0         1  ...          0         1        0
647           0         0         0  ...          0         1        0
648           0         0         0  ...          0         1        0

[1044 rows x 17 columns]
```

```
data = pd.concat([data, dummies], axis=1)

data.drop(['Mjob', 'Fjob', 'reason', 'guardian'], axis=1,
inplace=True)
data
```

```
    school sex  age address  ... r_reputation g_father  g_mother
g_other
0       GP   F   18       U  ...            0        0         1
0
1       GP   F   17       U  ...            0        1         0
0
2       GP   F   15       U  ...            0        0         1
0
3       GP   F   15       U  ...            0        0         1
0
4       GP   F   16       U  ...            0        1         0
0
..     ...  ..  ...     ...  ...          ...      ...       ...
...
644     MS   F   19       R  ...            0        0         1
0
645     MS   F   18       U  ...            0        0         1
0
646     MS   F   18       U  ...            0        0         1
0
647     MS   M   17       U  ...            0        0         1
0
648     MS   M   18       R  ...            0        0         1
0

[1044 rows x 46 columns]
```

```
nonnumeric_columns = [data.columns[index] for index, dtype in
enumerate(data.dtypes) if dtype == 'object']

for column in nonnumeric_columns:
    print(f"{column}: {data[column].unique()}")
```

```
school: ['GP' 'MS']
sex: ['F' 'M']
address: ['U' 'R']
```

```
famsize: ['GT3' 'LE3']
Pstatus: ['A' 'T']
schoolsup: ['yes' 'no']
famsup: ['no' 'yes']
paid: ['no' 'yes']
activities: ['no' 'yes']
nursery: ['yes' 'no']
higher: ['yes' 'no']
internet: ['no' 'yes']
romantic: ['no' 'yes']
```

```
encoder = LabelEncoder()

for column in nonnumeric_columns:
    data[column] = encoder.fit_transform(data[column])

for dtype in data.dtypes:
    print(dtype)
```

```
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
int64
uint8
uint8
uint8
```

```
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
uint8
```

```
y = data['G3']
X = data.drop('G3', axis=1)
```

Scaling

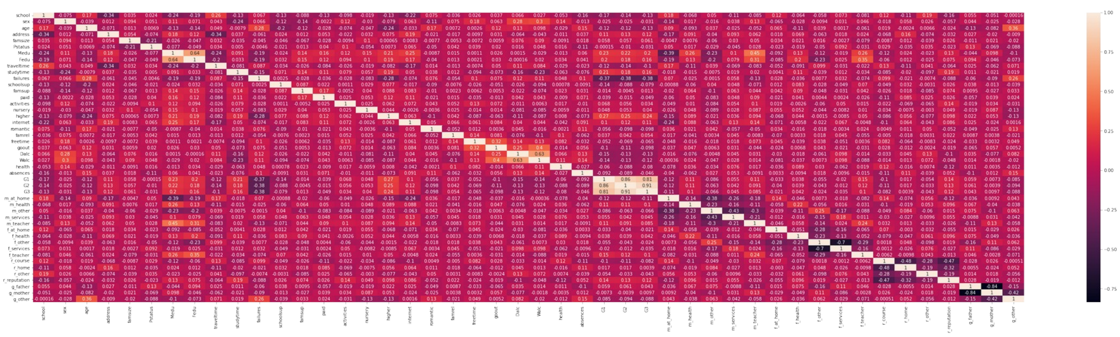
```
X
```

```
     school  sex  age  address  ...  r_reputation  g_father  g_mother
g_other
0         0    0   18        1  ...             0         0         1
0
1         0    0   17        1  ...             0         1         0
0
2         0    0   15        1  ...             0         0         1
0
3         0    0   15        1  ...             0         0         1
0
4         0    0   16        1  ...             0         1         0
0
..      ...  ...  ...      ...  ...           ...       ...       ...
...
644       1    0   19        0  ...             0         0         1
0
645       1    0   18        1  ...             0         0         1
0
646       1    0   18        1  ...             0         0         1
0
647       1    1   17        1  ...             0         0         1
0
648       1    1   18        0  ...             0         0         1
0

[1044 rows x 45 columns]
```

```
scaler = StandardScaler()

X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

X
```

```
        school       sex       age  ...  g_father  g_mother  g_other
0    -0.593575 -0.875498  1.027889  ... -0.550791  0.658837 -0.27419
1    -0.593575 -0.875498  0.221035  ...  1.815571 -1.517827 -0.27419
2    -0.593575 -0.875498 -1.392674  ... -0.550791  0.658837 -0.27419
3    -0.593575 -0.875498 -1.392674  ... -0.550791  0.658837 -0.27419
4    -0.593575 -0.875498 -0.585820  ...  1.815571 -1.517827 -0.27419
...        ...       ...       ...  ...       ...       ...      ...
1039  1.684706 -0.875498  1.834744  ... -0.550791  0.658837 -0.27419
1040  1.684706 -0.875498  1.027889  ... -0.550791  0.658837 -0.27419
1041  1.684706 -0.875498  1.027889  ... -0.550791  0.658837 -0.27419
1042  1.684706  1.142207  0.221035  ... -0.550791  0.658837 -0.27419
1043  1.684706  1.142207  1.027889  ... -0.550791  0.658837 -0.27419

[1044 rows x 45 columns]
```

```
plt.figure(figsize=(50, 12))
sns.heatmap(data.corr(), annot=True)
plt.show()
```



Building the model

Training

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
train_size=0.6, random_state=42)
```

Linear Regression

```
model = LinearRegression()
model.fit(X_train, y_train)
prediction = model.predict(X_test)
```

Results

```
print(f"Model R2: {model.score(X_test, y_test)}")
```

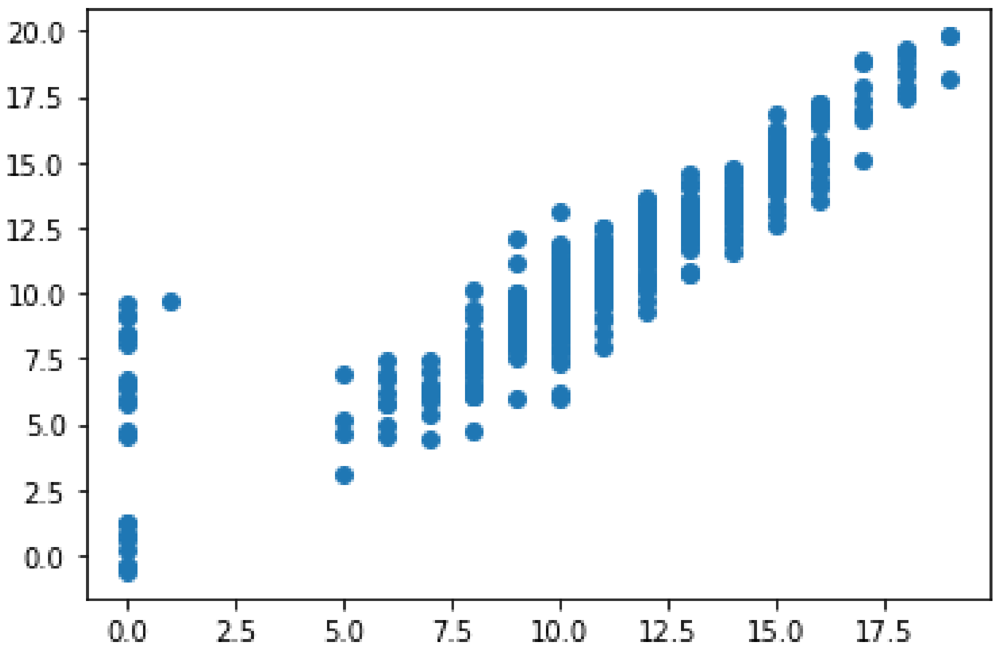
```
Model R2: 0.7929947624168348
```

```
print("R2  :", r2_score(y_test, prediction))
print("MAE :", mean_absolute_error(y_test, prediction))
print("MSE :", mean_squared_error(y_test, prediction))
```

```
R2  : 0.7929947624168348
MAE : 1.0532870868212523
MSE : 3.133300626912634
```

```
plt.scatter(y_test, prediction)
```

```
<matplotlib.collections.PathCollection at 0x7fdeee6fd310>
```




```
R.fit(X_train,y_train)
prediction = R.predict(X_test)
```

```
R.score(X_test,y_test)
```

```
0.7928724645401053
```

Random Forrest Regression

```
RF.fit(X_train, y_train)
prediction = RF.predict(X_test)
```

```
print("R2  :", r2_score(y_test, prediction))
print("MAE :", mean_absolute_error(y_test, prediction))
print("MSE :", mean_squared_error(y_test, prediction))
```

```
R2  : 0.8283704357538773
MAE : 0.9590430622009569
MSE : 2.597842583732058
```

```
plt.scatter(y_test, prediction)
```

```
<matplotlib.collections.PathCollection at 0x7fdeed11a650>
```