EDA

March 16, 2021

[3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib
import category_encoders as ce
import warnings
warnings.filterwarnings("ignore")
```

[4]:
```
grades_df = pd.read_csv('school_grades_dataset.csv')


with pd.option_context('display.max_columns', None):
    print(grades_df.head())
```

```
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher
1     GP   F   17       U     GT3       T     1     1  at_home     other
2     GP   F   15       U     LE3       T     1     1  at_home     other
3     GP   F   15       U     GT3       T     4     2   health  services
4     GP   F   16       U     GT3       T     3     3    other     other

   reason guardian  traveltime  studytime  failures schoolsup famsup paid  \
0  course   mother           2          2         0       yes     no   no
1  course   father           1          2         0        no    yes   no
2   other   mother           1          2         0       yes     no   no
3    home   mother           1          3         0        no    yes   no
4    home   father           1          2         0        no    yes   no

  activities nursery higher internet romantic  famrel  freetime  goout  Dalc  \
0         no     yes    yes       no       no       4         3      4     1
1         no      no    yes      yes       no       5         3      3     1
2         no     yes    yes      yes       no       4         3      2     2
3        yes     yes    yes      yes      yes       3         2      2     1
4         no     yes    yes       no       no       4         3      2     1

   Walc  health  absences  G1  G2  G3
0     1       3         4   0  11  11
```

```
1     1       3         2   9  11  11
2     3       3         6  12  13  12
3     1       5         0  14  14  14
4     2       5         0  11  13  13
```

[5]:
```
grades_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      649 non-null    object
 1   sex         649 non-null    object
 2   age         649 non-null    int64
 3   address     649 non-null    object
 4   famsize     649 non-null    object
 5   Pstatus     649 non-null    object
 6   Medu        649 non-null    int64
 7   Fedu        649 non-null    int64
 8   Mjob        649 non-null    object
 9   Fjob        649 non-null    object
 10  reason      649 non-null    object
 11  guardian    649 non-null    object
 12  traveltime  649 non-null    int64
 13  studytime   649 non-null    int64
 14  failures    649 non-null    int64
 15  schoolsup   649 non-null    object
 16  famsup      649 non-null    object
 17  paid        649 non-null    object
 18  activities  649 non-null    object
 19  nursery     649 non-null    object
 20  higher      649 non-null    object
 21  internet    649 non-null    object
 22  romantic    649 non-null    object
 23  famrel      649 non-null    int64
 24  freetime    649 non-null    int64
 25  goout       649 non-null    int64
 26  Dalc        649 non-null    int64
 27  Walc        649 non-null    int64
 28  health      649 non-null    int64
 29  absences    649 non-null    int64
 30  G1          649 non-null    int64
 31  G2          649 non-null    int64
 32  G3          649 non-null    int64
dtypes: int64(16), object(17)
memory usage: 124.3+ KB
```

[6]:
```
attributes_df = pd.read_csv('attributes_school_grades.csv')

with pd.option_context('max_colwidth', 150):
    print(attributes_df)
```

```
          name     type  \
0       school   string
1          sex   string
2          age  integer
3      address   string
4      famsize   string
5      Pstatus   string
6         Medu  integer
7         Fedu  integer
8         Mjob   string
9         Fjob   string
10      reason   string
11    guardian   string
12  traveltime  integer
13   studytime  integer
14    failures  integer
15   schoolsup   string
16      famsup   string
17        paid   string
18  activities   string
19     nursery   string
20      higher   string
21    internet   string
22    romantic   string
23      famrel  integer
24    freetime  integer
25       goout  integer
26        Dalc  integer
27        Walc  integer
28      health  integer
29    absences  integer
30          G1  integer
31          G2  integer
32          G3  integer

                                                                 description
0
student's school (binary: GP" Gabriel Pereira or "MS" Mousinho da Silveira)"
1
student's sex (binary: F" female or "M" male)"
2
student's age (numeric: from 15 to 22)
```

3
student's home address type (binary: U" urban or "R" rural)"
4
family size (binary: LE3" less or equal to 3 or "GT3" greater than 3)"
5
parent's cohabitation status (binary: T" living together or "A" apart)"
6 mother's education (numeric: 0: none, 1: primary education (4th grade),
2: 5th to 9th grade, 3 _ secondary education or 4 _ higher education)
7 father's education (numeric: 0 - none, 1 - primary education (4th grade),
2 _ 5th to 9th grade, 3 _ secondary education or 4 _ higher education)
8 mother's job (nominal: teacher, health care
related, civil services (e.g. administrative or police), at_home or other)
9 father's job (nominal: teacher, health care
related, civil services (e.g. administrative or police), at_home or other)
10 reason to choose this school
(nominal: close to home, school reputation, course preference or other)
11
student's guardian (nominal: mother, father or other)
12 home to school travel time (numeric:
1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
13 weekly study time
(numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
14
number of past class failures (numeric: n if 1<=n<3, else 4)
15
extra educational support (binary: yes or no)
16
family educational support (binary: yes or no)
17 extra paid
classes within the course subject (Math or Portuguese) (binary: yes or no)
18
extra-curricular activities (binary: yes or no)
19
attended nursery school (binary: yes or no)
20
wants to take higher education (binary: yes or no)
21
Internet access at home (binary: yes or no)
22
with a romantic relationship (binary: yes or no)
23
quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
24
free time after school (numeric: from 1 - very low to 5 - very high)
25
going out with friends (numeric: from 1 - very low to 5 - very high)
26
workday alcohol consumption (numeric: from 1 - very low to 5 - very high)

```
27
weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
28
current health status (numeric: from 1 - very bad to 5 - very good)
29
number of school absences (numeric: from 0 to 93)
30
first period grade (numeric: from 0 to 20)
31
second period grade (numeric: from 0 to 20)
32
Predictor Class: final grade (numeric: from 0 to 20)
```

[7]:
```
grades_df.describe()
```

[7]:

	age	Medu	Fedu	traveltime	studytime	failures
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000
mean	16.744222	2.514638	2.306626	1.568567	1.930663	0.221880
std	1.218138	1.134552	1.099931	0.748660	0.829510	0.593235
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000
25%	16.000000	2.000000	1.000000	1.000000	1.000000	0.000000
50%	17.000000	2.000000	2.000000	1.000000	2.000000	0.000000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000

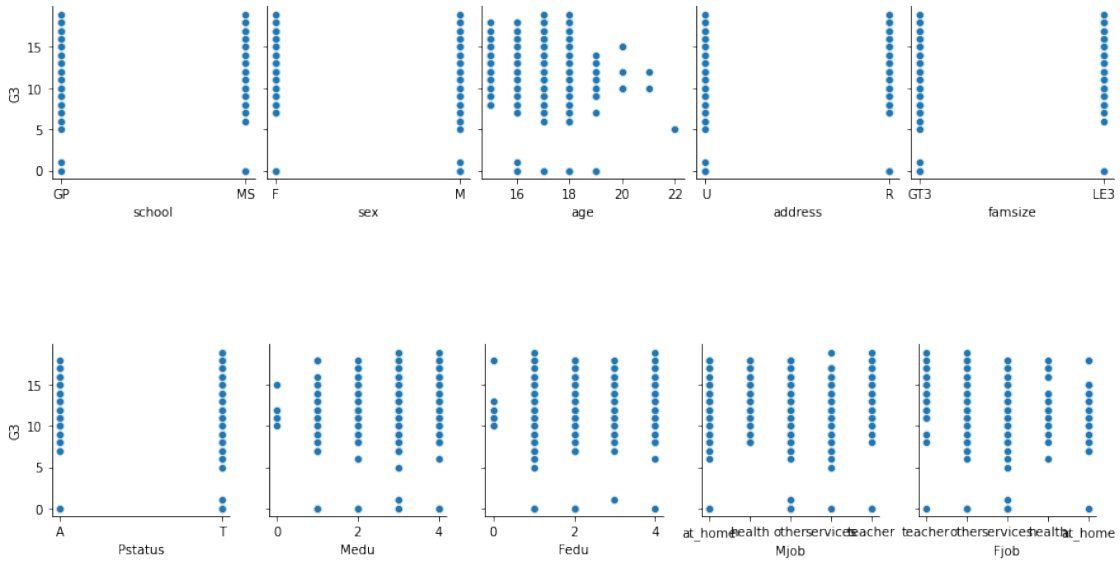
	famrel	freetime	goout	Dalc	Walc	health
count	649.000000	649.000000	649.000000	649.000000	649.000000	649.000000
mean	3.930663	3.180277	3.184900	1.502311	2.280431	3.536210
std	0.955717	1.051093	1.175766	0.924834	1.284380	1.446259
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	4.000000	3.000000	2.000000	1.000000	1.000000	2.000000
50%	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000
75%	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000
max	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

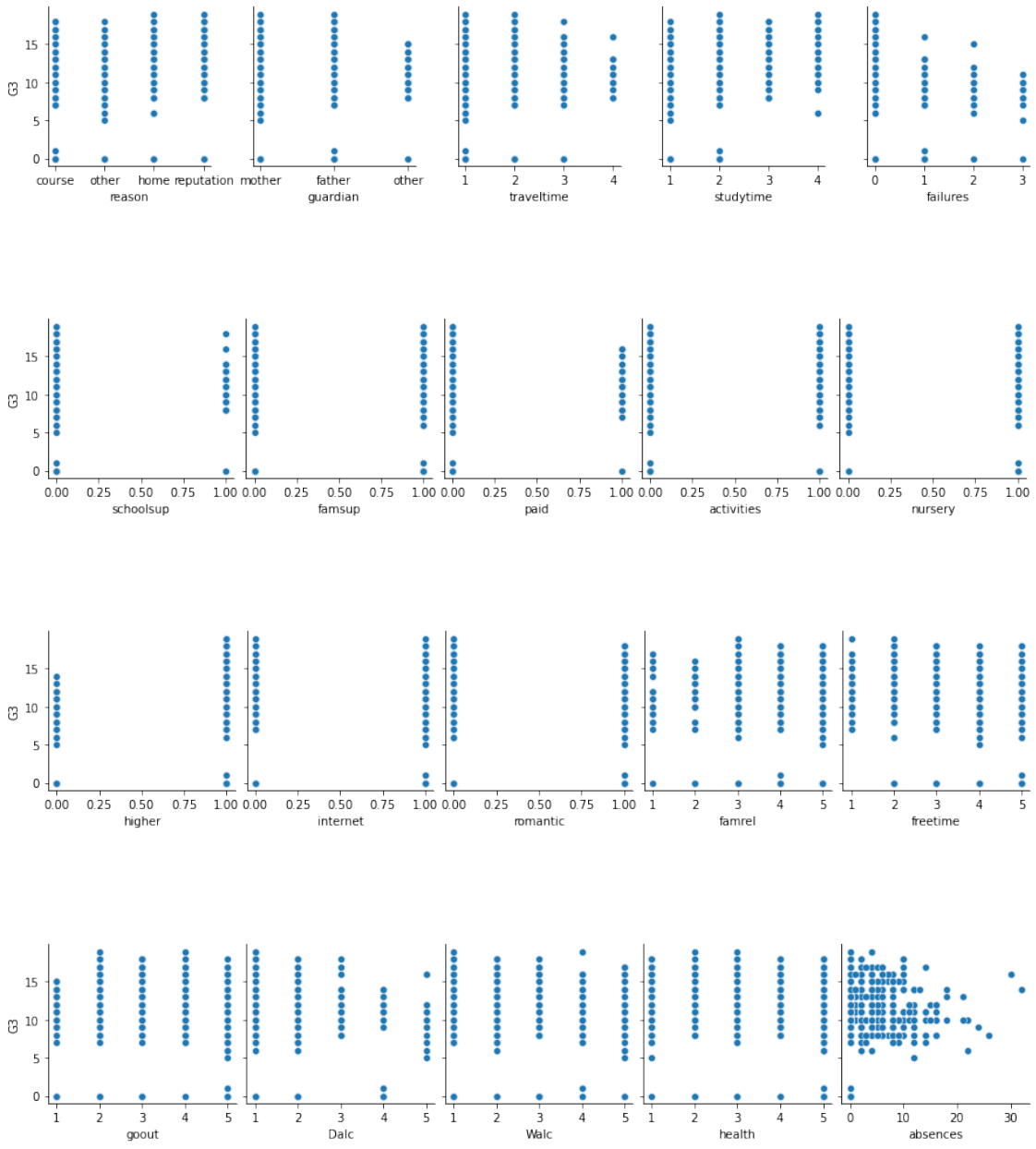
	absences	G1	G2	G3
count	649.000000	649.000000	649.000000	649.000000
mean	3.659476	11.399076	11.570108	11.906009
std	4.640759	2.745265	2.913639	3.230656
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	10.000000	10.000000	10.000000
50%	2.000000	11.000000	11.000000	12.000000
75%	6.000000	13.000000	13.000000	14.000000
max	32.000000	19.000000	19.000000	19.000000

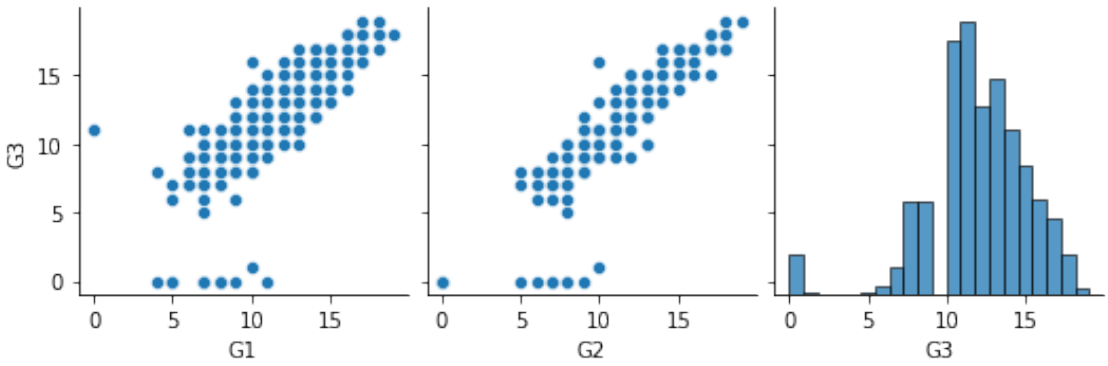
[8]:
```
# Mamy bardzo dużo zmiennych kategorycznych. W danych nie ma braków. Spróbujmy␣
↪zakodować zmienne kategorczne:
```

```
# Zmienne: schoolsup, famsup, paid, activities, nursery, higher, internet,␣
 ↪romantic jako zmienne o wartościach yes/no możemy
# zakodować za pomocą Ordinal Encoder'a
cols = ['schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher',␣
 ↪'internet', 'romantic']
grades_df_new = grades_df.drop(columns = cols)


for i in cols:
    encoder = ce.OrdinalEncoder(mapping = [{'col': i, 'mapping': {'yes': 1,␣
 ↪'no': 0}},])
    grades_df_new[i] = encoder.fit_transform(grades_df)[i]


bin_cols = cols + ['school', 'sex', 'address', 'famsize', 'Pstatus']
```

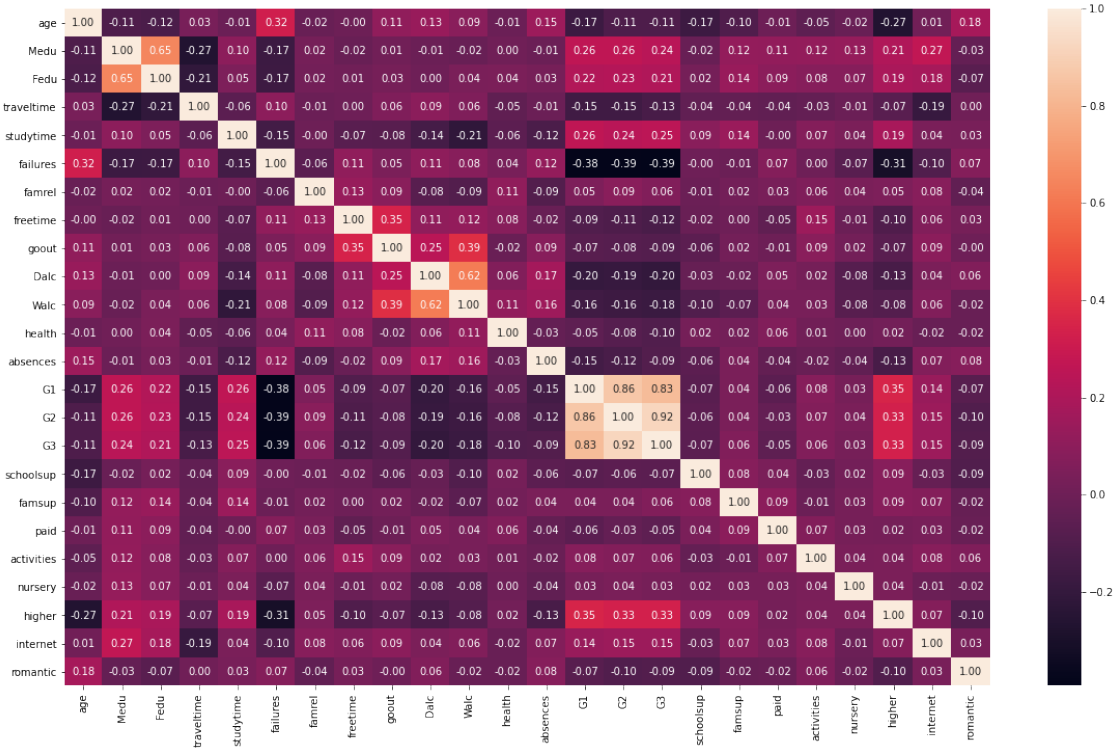
[9]:
```
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[0:5])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[5:10])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[10:15])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[15:20])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[20:25])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[25:30])
sns.pairplot(grades_df_new, y_vars="G3", x_vars=grades_df.columns.values[30:])

plt.show()
```



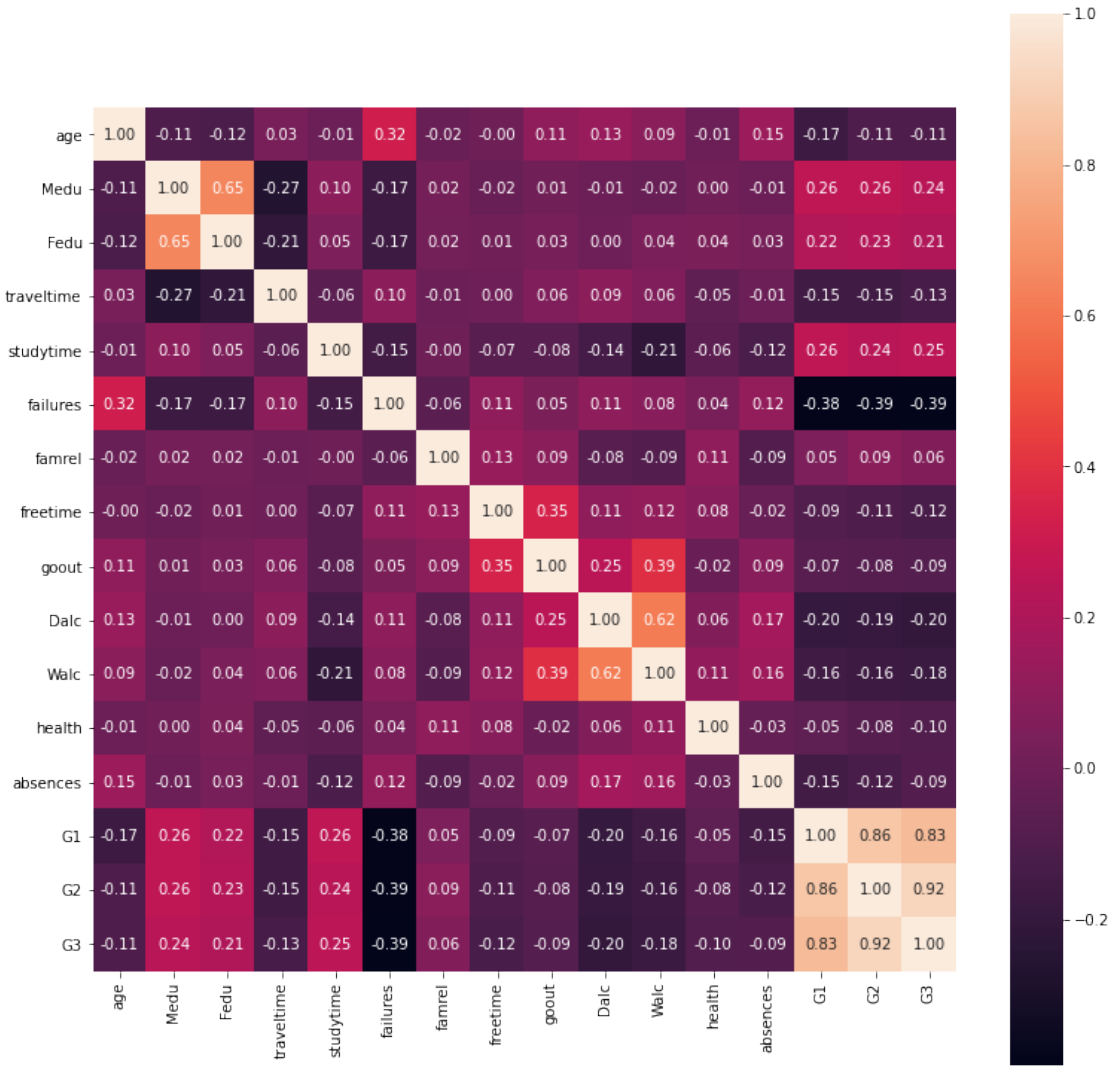

G3
reason
course
other
home
reputation
guardian
mother
father
other
traveltime
studytime
failures
schoolsup
famsup
paid
activities
nursery
higher
internet
romantic
famrel
freetime
goout
Dalc
Walc
health
absences



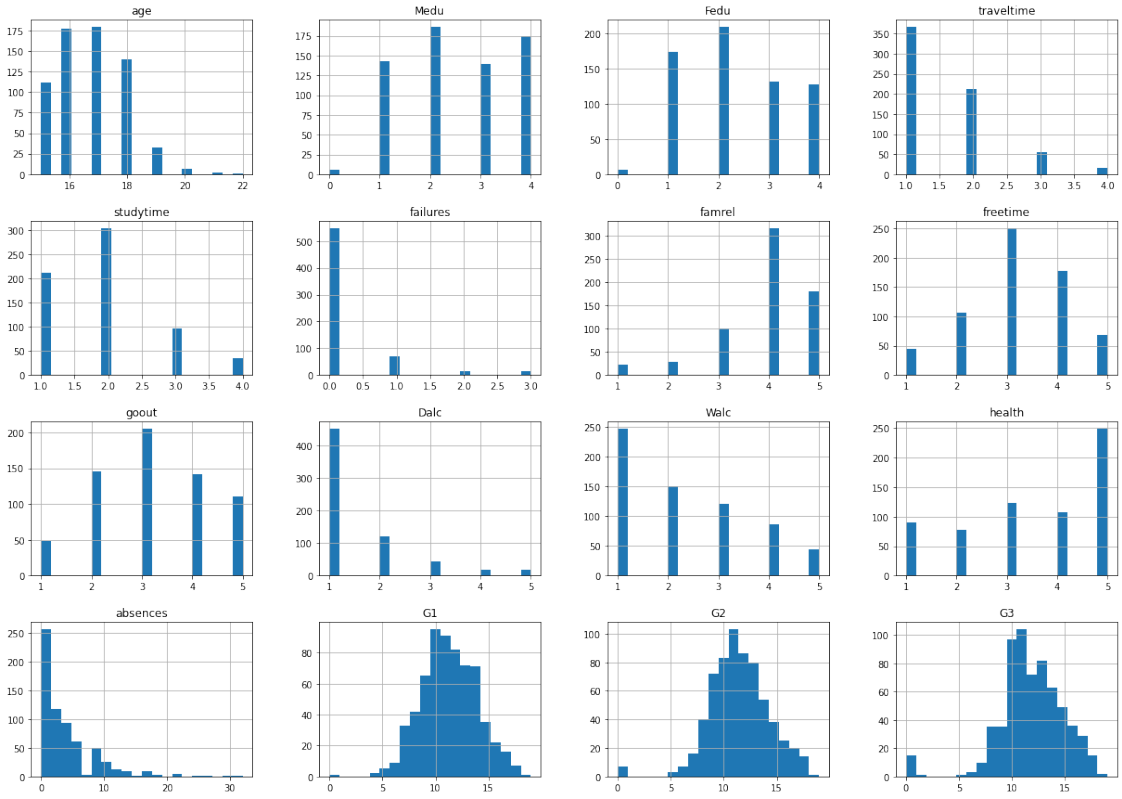
[10]:
```
corr = grades_df_new.corr()
f, ax = plt.subplots(figsize=(20, 12))
ax = sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values,
            annot = True,
            fmt='.2f')
```



[11]:
```
#heatmapa tylko dla zminnych numerycznych

fig, ax = plt.subplots(figsize=(13, 13))
sns.heatmap(grades_df.corr(), annot=True, square=True,  fmt='.2f')
plt.show()
```

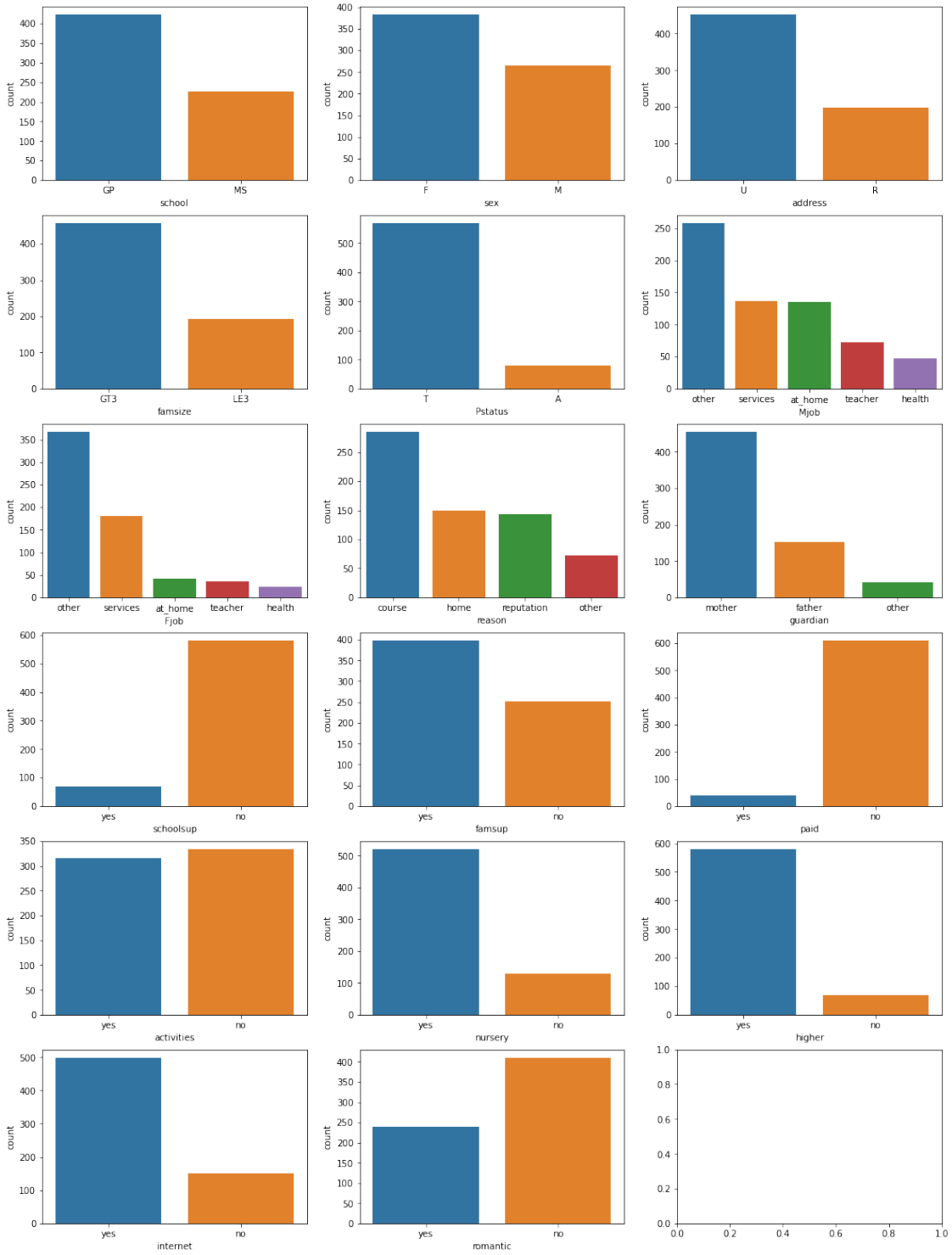


[12]:
```
cols = grades_df.describe().columns

grades_df[cols].hist(bins = 20, figsize=(21, 15))
plt.show()
```

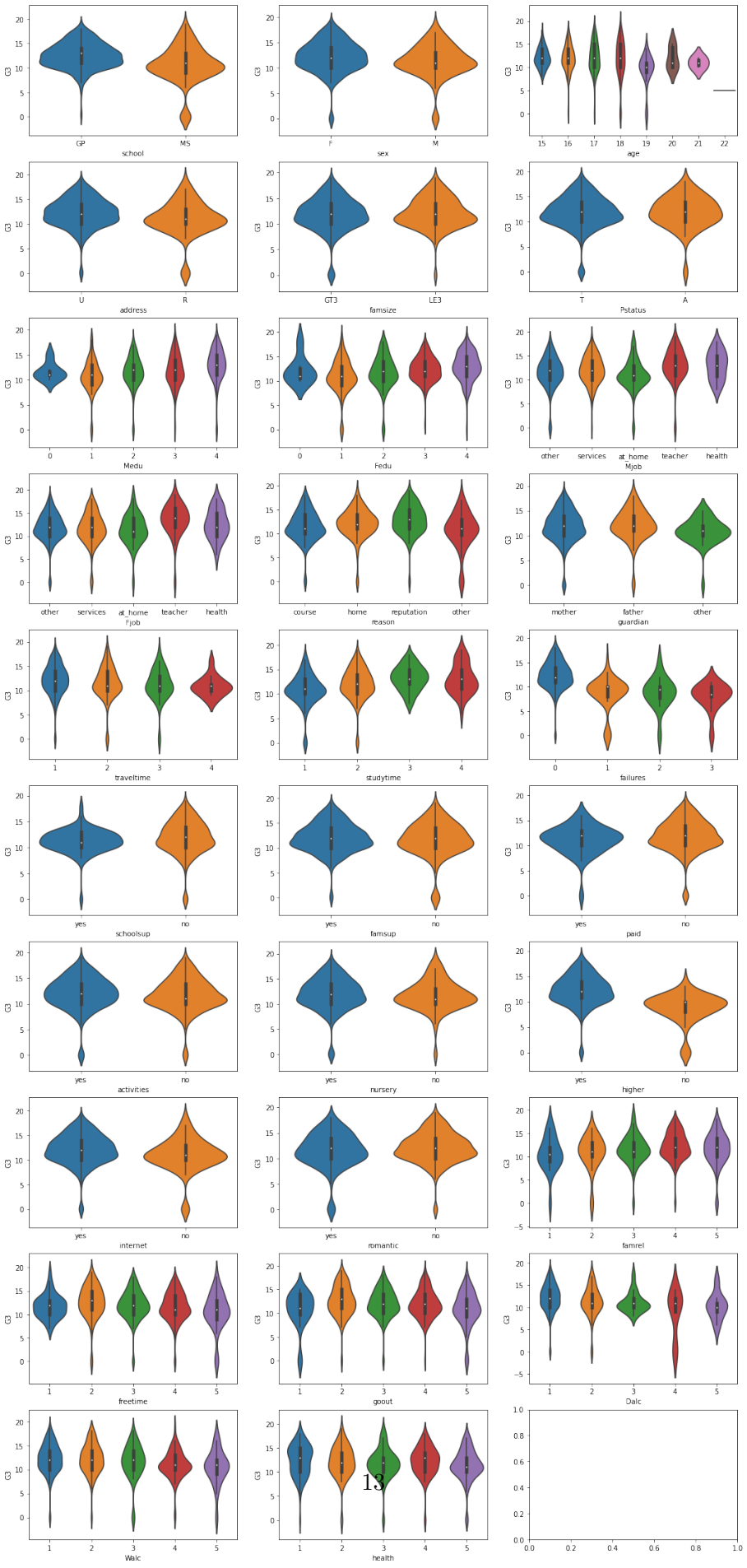


[13]:
```
discrete_vals = grades_df.drop(cols, axis=1)

len(discrete_vals.columns)

def y_or_n_vals(col):
    if len(col.value_counts()) == 2:
        if all(col.value_counts().index == ['yes', 'no']) | all(col.
 ↪value_counts().index == ['no', 'yes']):
            return True
    return False

fig, axs = plt.subplots(6, 3, figsize = (18, 25))
for i in range(len(discrete_vals.columns)):
    col = discrete_vals.columns[i]

    if y_or_n_vals(discrete_vals[col]):
        sns.countplot(data = discrete_vals, x=col, ax = axs[i // 3, i % 3],␣
 ↪order = ['yes', 'no'])
    else:
        order = discrete_vals[col].value_counts().sort_values(ascending=False).
 ↪index
```

```
        sns.countplot(data = discrete_vals, x=col, ax = axs[i // 3, i % 3],␣
 ↪order = order)
plt.show()
```

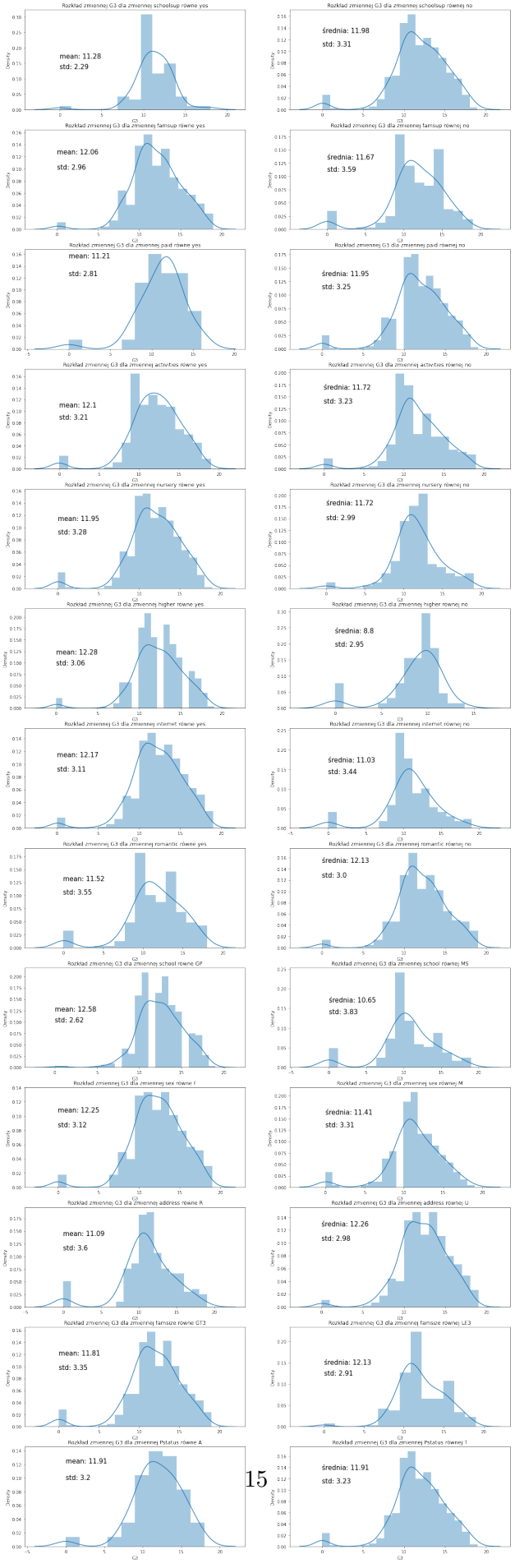


[15]:
```
df = grades_df.drop(['G1', 'G2', 'absences'], axis = 1)

def one_to_five_vals():
    if len(col.value_counts()) == 5:
        if all(col.value_counts().index == ['yes', 'no']) | all(col.
 ↪value_counts().index == ['no', 'yes']):
            return True
    return False

fig, axs = plt.subplots(10, 3, figsize = (18, 40))
for i in range(len(df.columns)-1):
    col = df.columns[i]

    if y_or_n_vals(df[col]):
        sns.violinplot(data = df, x=col, y = 'G3', ax = axs[i // 3, i % 3],␣
 ↪order = ['yes', 'no'])
    elif type(df[col].value_counts().index[0]) is np.int64:
        sns.violinplot(data = df, x=col, y = 'G3', ax = axs[i // 3, i % 3],␣
 ↪order = df[col].unique().sort())
    else:
        order = df[col].value_counts().sort_values(ascending=False).index
        sns.violinplot(data = df, x=col, y = 'G3', ax = axs[i // 3, i % 3],␣
 ↪order = order)
plt.show()
```


G3
school
GP
MS
sex
F
M
age
address
U
R
famsize
GT3
LE3
Pstatus
T
A
Medu
Fedu
Mjob
other
services
at_home
teacher
health
Fjob
reason
course
home
reputation
guardian
mother
father
traveltime
studytime
failures
schoolsup
yes
no
famsup
paid
activities
nursery
higher
internet
romantic
famrel
freetime
goout
Dalc
Walc
health

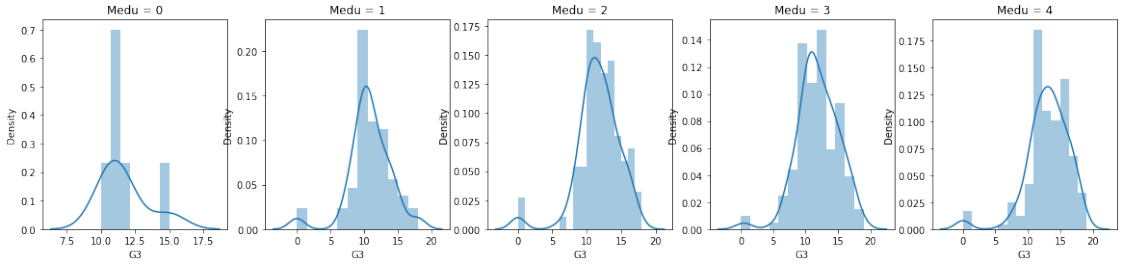
[ ]:
```
#Jeszcze raz te same wykresy, żeby wstawić do prezentacji

#fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(14, 4))
#sns.violinplot(data = grades_df, x='nursery', y = 'G3', ax = ax1, order =␣
↪['yes', 'no'])
#sns.violinplot(data = grades_df, x='reason', y = 'G3', ax = ax2, order =␣
↪grades_df['reason'].unique())
#sns.violinplot(data = grades_df, x='higher', y = 'G3', ax = ax3, order =␣
↪['yes', 'no'])
#plt.show()
```

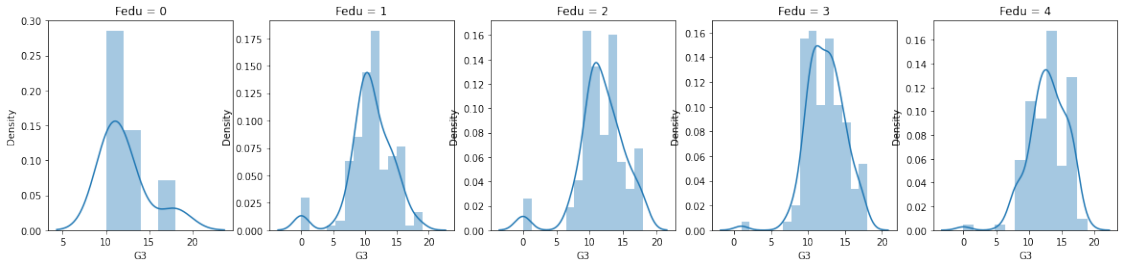
[16]:
```
fig, axs = plt.subplots(len(bin_cols), 2, figsize = (20, 65))
for i in range(len(bin_cols)):
    a, b = sorted(grades_df[bin_cols[i]].unique())
    if (a == 'no'): a,b = b,a

    X = grades_df.loc[grades_df[bin_cols[i]] == a]
    plot_dens=sns.distplot(X['G3'], ax = axs[i, 0])
    plot_dens.set_title('Rozkład zmiennej G3 dla zmiennej ' + bin_cols[i] + '␣
↪równe ' + a)
    txt = "mean: " + str(np.mean(X['G3']).round(2))
    top = (max(X['G3'].value_counts()))/len(X)
    plot_dens.text(0, 0.75 * top,s = txt, fontsize=15)
    txt = "std: " + str(np.std(X['G3']).round(2))
    plot_dens.text(0, 0.6 * top,s = txt, fontsize=15)

    X = grades_df.loc[grades_df[bin_cols[i]] == b]
    plot_dens=sns.distplot(X['G3'], ax = axs[i, 1])
    plot_dens.set_title('Rozkład zmiennej G3 dla zmiennej ' + bin_cols[i] + '␣
↪równej ' + b)
    txt = "średnia: " + str(np.mean(X['G3']).round(2))
    top = (max(X['G3'].value_counts()))/len(X)
    plot_dens.text(0, 0.85 * top,s= txt, fontsize=15)
    txt = "std: " + str(np.std(X['G3']).round(2))
    plot_dens.text(0, 0.7 * top,s = txt, fontsize=15)
plt.show()
```

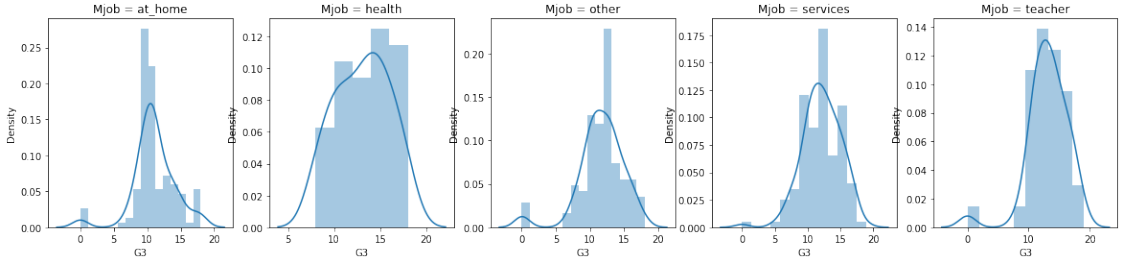

Rozkład zmiennej G3 dla zmiennej schoolsup równe yes
mean: 11.28
std: 2.29
Rozkład zmiennej G3 dla zmiennej schoolsup równej no
średnia: 11.98
std: 3.31
Rozkład zmiennej G3 dla zmiennej famsup równe yes
mean: 12.06
std: 2.96
Rozkład zmiennej G3 dla zmiennej famsup równej no
średnia: 11.67
std: 3.59
Rozkład zmiennej G3 dla zmiennej paid równe yes
mean: 11.21
std: 2.81
Rozkład zmiennej G3 dla zmiennej paid równej no
średnia: 11.95
std: 3.25
Rozkład zmiennej G3 dla zmiennej activities równe yes
mean: 12.1
std: 3.21
Rozkład zmiennej G3 dla zmiennej activities równej no
średnia: 11.72
std: 3.23
Rozkład zmiennej G3 dla zmiennej nursery równe yes
mean: 11.95
std: 3.28
Rozkład zmiennej G3 dla zmiennej nursery równej no
średnia: 11.72
std: 2.99
Rozkład zmiennej G3 dla zmiennej higher równe yes
mean: 12.28
std: 3.06
Rozkład zmiennej G3 dla zmiennej higher równej no
średnia: 8.8
std: 2.95
Rozkład zmiennej G3 dla zmiennej internet równe yes
mean: 12.17
std: 3.11
Rozkład zmiennej G3 dla zmiennej internet równej no
średnia: 11.03
std: 3.44
Rozkład zmiennej G3 dla zmiennej romantic równe yes
mean: 11.52
std: 3.55
Rozkład zmiennej G3 dla zmiennej romantic równej no
średnia: 12.13
std: 3.0
Rozkład zmiennej G3 dla zmiennej school równe GP
mean: 12.58
std: 2.62
Rozkład zmiennej G3 dla zmiennej school równej MS
średnia: 10.65
std: 3.83
Rozkład zmiennej G3 dla zmiennej sex równe F
mean: 12.25
std: 3.12
Rozkład zmiennej G3 dla zmiennej sex równej M
średnia: 11.41
std: 3.31
Rozkład zmiennej G3 dla zmiennej address równe R
mean: 11.09
std: 3.6
Rozkład zmiennej G3 dla zmiennej address równej U
średnia: 12.26
std: 2.98
Rozkład zmiennej G3 dla zmiennej famsize równe GT3
mean: 11.81
std: 3.35
Rozkład zmiennej G3 dla zmiennej famsize równej LE3
średnia: 12.13
std: 2.91
Rozkład zmiennej G3 dla zmiennej Pstatus równe A
mean: 11.91
std: 3.2
Rozkład zmiennej G3 dla zmiennej Pstatus równej T
średnia: 11.91
std: 3.23

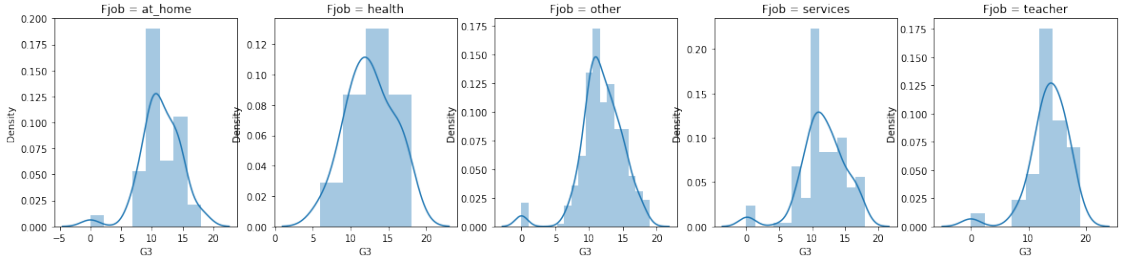
- Osoby nie posiadający korepetycji uczą się lepiej (zaskakujące).
- Osoby posiadające wsparcie rodziny uczą się lepiej. Wykresy osób posiadających / nie posiadających korepetycji przypominają rozkład normalny.
- Osoby biorące udział w zajęciach pozalekcyjnych lepiej się uczą.
- Osoby po przedszkolu lepiej się uczą.
- Zdecydowanie lepiej uczą się osoby chcące osiągnąć wykształcenie wyższe
- Zdecydowanie lepiej uczą się osoby posiadające dostęp do internetu
- Rozkłady zmiennej romantic wyglądają podobnie, natomiast znacznie wyższa średnia wychodzi gdy 'romantic' jest 'no' Lepsze oceny mają uczniowie szkoły GP.
- Dziewczynki uczą się lepiej od chłopców.
- Mieszkańcy miast uczą się lepiej niż mieszkańcy wsi.
- Histogram sugeruje, że lepiej uczą się dzieci z małych rodzin, matomiast wyższą średnią ocen uzyskały dzieci z dużych rodzin. - Rozkłady dla zmiennej "Pstatus" znacząco się różnią, natomiast średnia wygląda identycznie.
- szkoła GP lepsza od MS

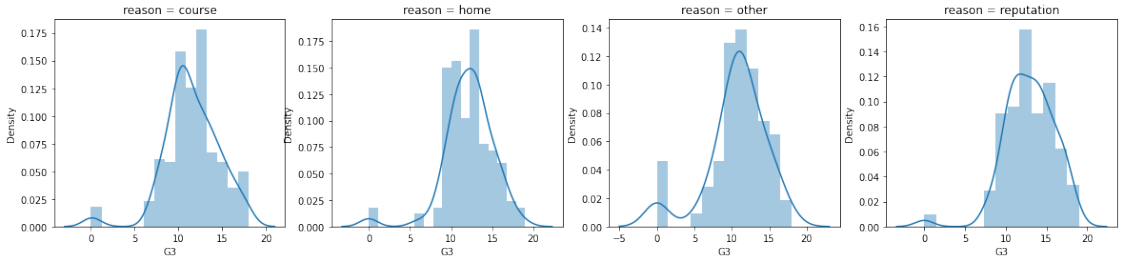
[17]:
```
cols_cat = ['Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'famrel',␣
 ↪'freetime', 'goout', 'Dalc', 'Walc', 'health', 'traveltime', 'studytime',␣
 ↪'failures']

for i in range(len(cols_cat)):
    vals = sorted(grades_df[cols_cat[i]].unique())

    fig, axs = plt.subplots(1, len(vals), figsize = (20, 4))
    for a in range(len(vals)):
        X = grades_df.loc[grades_df[cols_cat[i]] == vals[a]]

        plot_dens=sns.distplot(X['G3'], ax = axs[a])
        plot_dens.set_title(cols_cat[i] + ' = ' + str(vals[a]))


    plt.show()
```

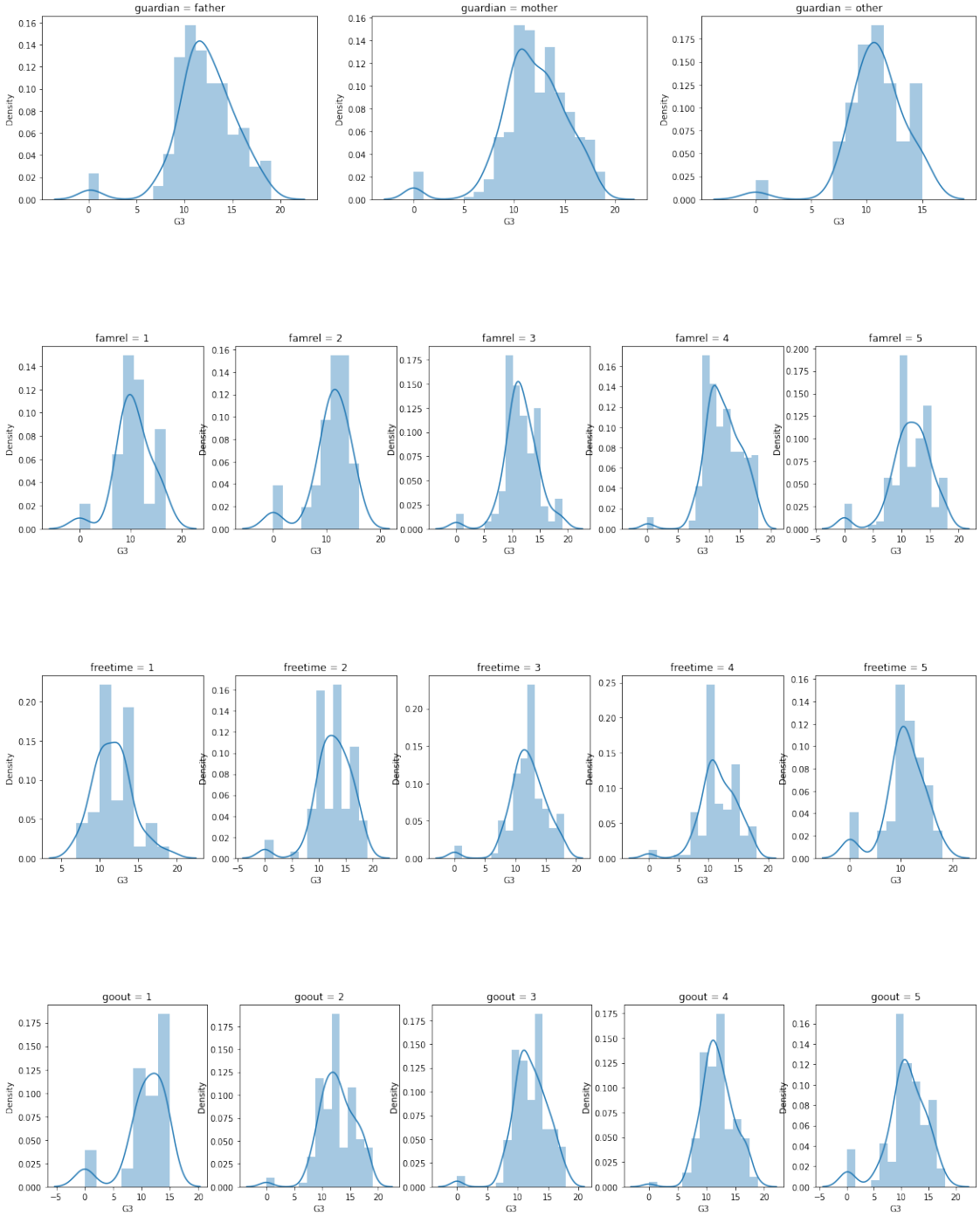


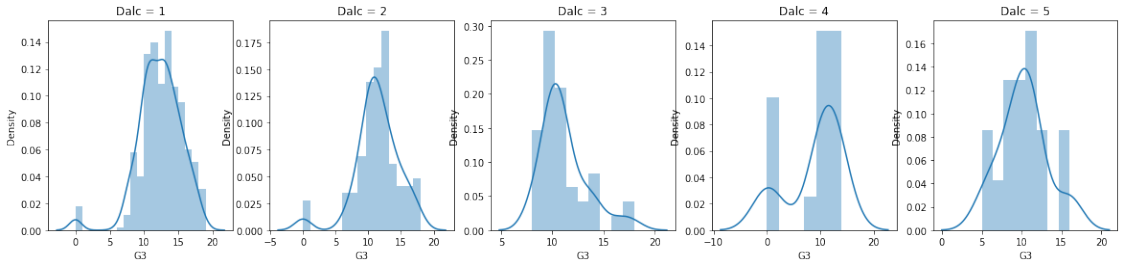

Fedu = 0
Fedu = 1
Fedu = 2
Fedu = 3
Fedu = 4
Density
G3

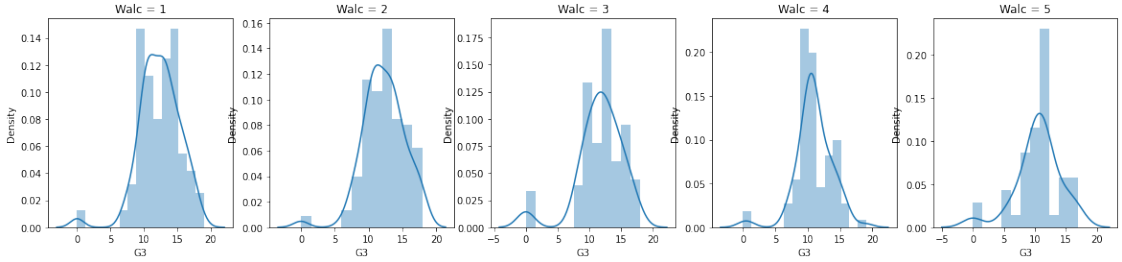

Mjob = at_home
Mjob = health
Mjob = other
Mjob = services
Mjob = teacher
Density
G3

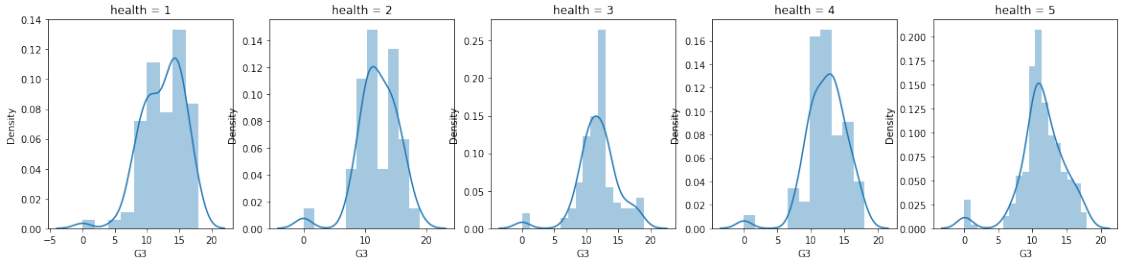

Fjob = at_home
Fjob = health
Fjob = other
Fjob = services
Fjob = teacher
Density
G3

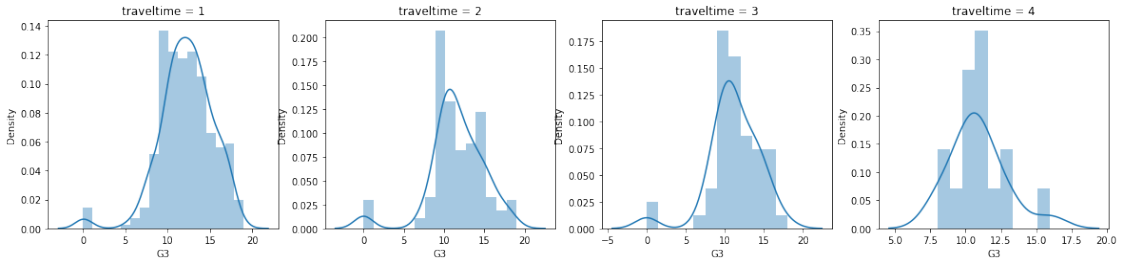

reason = course
reason = home
reason = other
reason = reputation
Density
G3

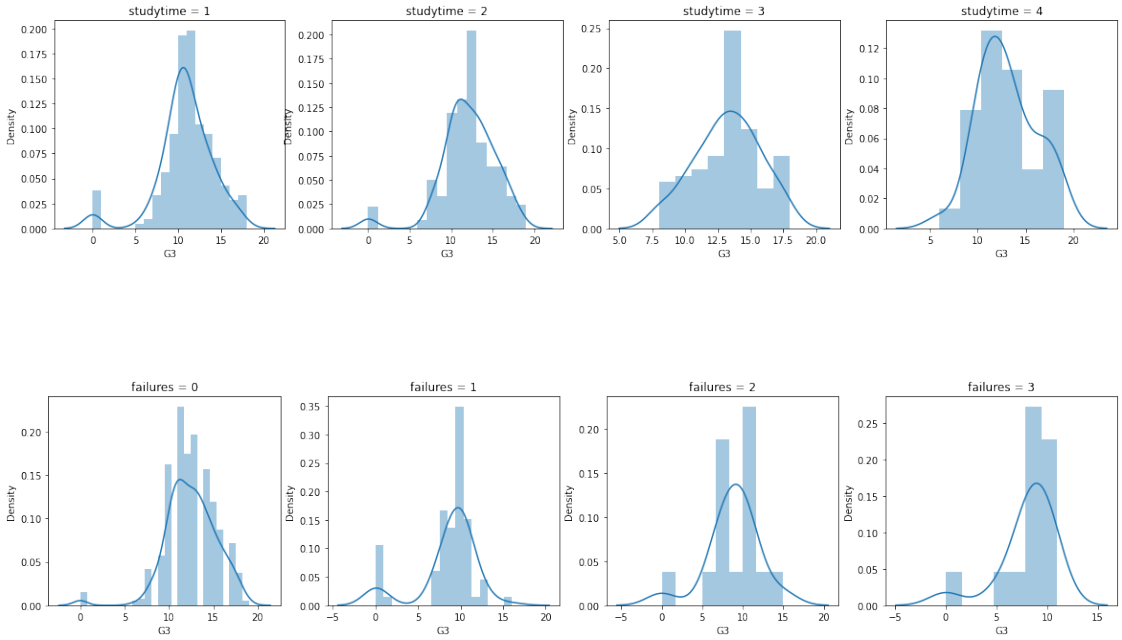

guardian = father
guardian = mother
guardian = other
famrel = 1
famrel = 2
famrel = 3
famrel = 4
famrel = 5
freetime = 1
freetime = 2
freetime = 3
freetime = 4
freetime = 5
goout = 1
goout = 2
goout = 3
goout = 4
goout = 5
Density
G3


Dalc = 1
Dalc = 2
Dalc = 3
Dalc = 4
Dalc = 5
Density
G3


Walc = 1
Walc = 2
Walc = 3
Walc = 4
Walc = 5
Density
G3


health = 1
health = 2
health = 3
health = 4
health = 5
Density
G3


traveltime = 1
traveltime = 2
traveltime = 3
traveltime = 4
Density
G3



[18]:
```
grades_df['Mjob'].value_counts()
```

[18]:
```
other       258
services    136
at_home     135
teacher      72
health       48
Name: Mjob, dtype: int64
```

[19]:
```
grades_df['Fjob'].value_counts()
```

[19]:
```
other       367
services    181
at_home      42
teacher      36
health       23
Name: Fjob, dtype: int64
```

- Najlepiej uczą się osoby, dla których współczynnik Medu i Fedu jest większy od 2, co było do przewidzenia, że dzieci wykształconych rodziców będą miały dobre oceny.
- Jeśli chodzi o wykształcenie rodziców, to najwyższe oceny mają dzieci lekarzy i nauczycieli, jednak te grupy są stosunkwo nieliczne w porównaniu do pozostałych
- Najlepsze oceny mają osoby, które wybrały szkołę ze względu na jej reputację, natomiast histogram dla powodu 'pozostałe' jest bardzo podobny do rozkładu naturalnego.
- Jeśli chodzi o wolny czas, to najlepiej uczą się osoby ze środka: Nadmiar / niedomiar wolnego czasu nie sprzyja dobrym ocenom.
- Pesymistycznym wnioskiem jest, że lepiej uczą się osoby rzadko wychodzące ze znajomymi.

- Spodziewanym wnioskiem jest, żę osoby pijące mało alkoholu, zarówno w dni powszednie, jak i weekendy mają lepsze oceny.
- Osoby o lepszym stanie zdrowia, mają lepsze oceny.
- Lepiej uczą się osoby, które mało czasu przeznaczają na podróż do szkoły.
- Lepiej uczą się osoby, które wiecej czasu przeznaczaja na naukę
- Lepiej uczą się osoby nie mające na koncie porażek, histogram przypomina wykres rozkładu normalnego
- najbardziej skorelowane są wyniki egzaminów - uczniowie utrzymują stały poziom

[21]:
```
grades_df['G3-result'] = np.where(grades_df['G3'] < 10, 'failed',
                                  np.where(grades_df['G3'] < 13, 'weak',
                                  np.where(grades_df['G3'] < 16, 'ok',
                                  np.where(grades_df['G3'] < 19, 'good',␣
 ↪'excellent'))))

grades_df.head()
```

[21]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	…	\
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	…	
1	GP	F	17	U	GT3	T	1	1	at_home	other	…	
2	GP	F	15	U	LE3	T	1	1	at_home	other	…	
3	GP	F	15	U	GT3	T	4	2	health	services	…	
4	GP	F	16	U	GT3	T	3	3	other	other	…	

	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3	G3-result
0	3	4	1	1	3	4	0	11	11	weak
1	3	3	1	1	3	2	9	11	11	weak
2	3	2	2	3	3	6	12	13	12	weak
3	2	2	1	1	5	0	14	14	14	ok
4	3	2	1	2	5	0	11	13	13	ok

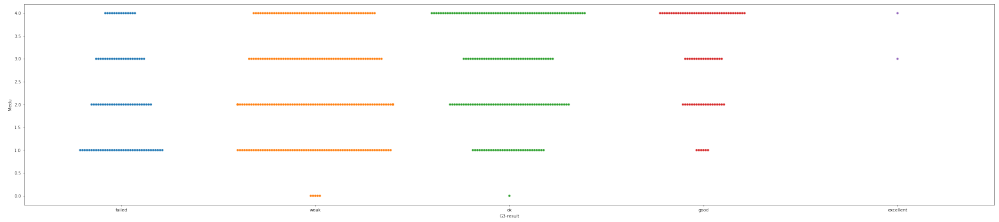
[5 rows x 34 columns]

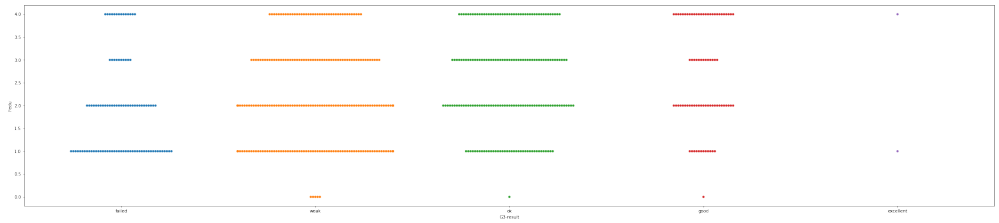
[22]:
```
grades_df['G3-result'].value_counts()
```

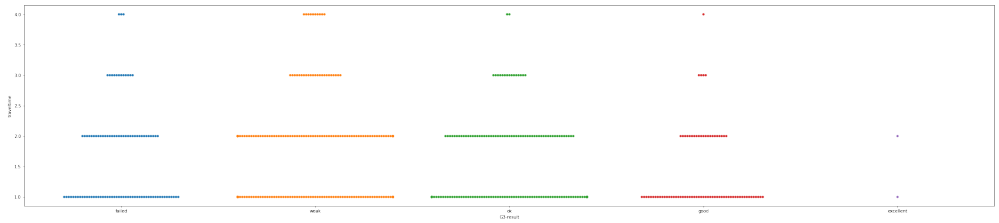
[22]:
```
weak         273
ok           194
failed       100
good          80
excellent      2
Name: G3-result, dtype: int64
```

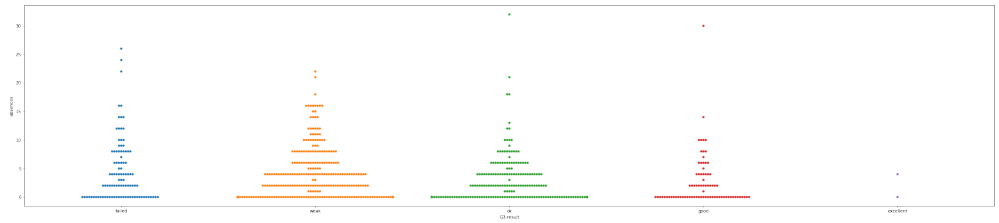
[23]:
```
interesting_cols = ['Medu', 'Fedu', 'traveltime', 'absences', 'Dalc', 'Walc',␣
 ↪'failures', 'G3-result']
df = grades_df[interesting_cols]
order = ['failed', 'weak', 'ok', 'good', 'excellent']
n = len(interesting_cols)
```

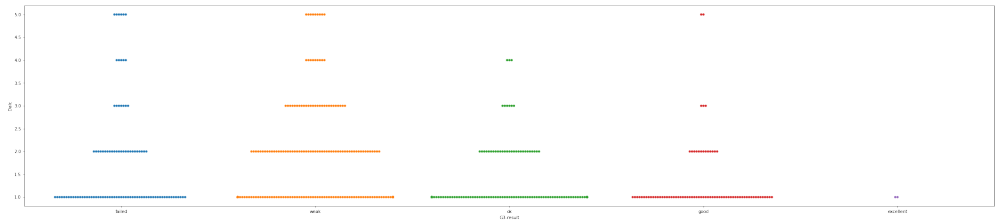
```
k = 0

fig, axs = plt.subplots(n, 1, figsize=(40, 80))
for i in range(n-1):
    col = interesting_cols[i]
    sns.swarmplot(data = grades_df, x = 'G3-result', y = col,  order = order,␣
 ↪ax = axs[i])
plt.show()
```

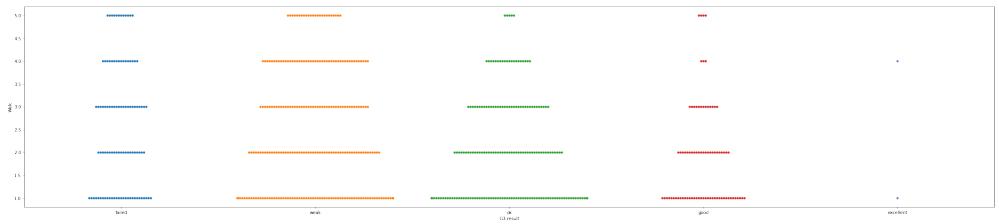

Meals
failed
weak
ok
good
excellent
Q1 result

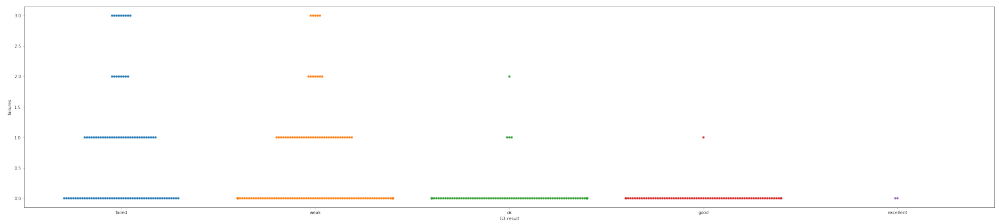

Hobby
failed
weak
ok
good
excellent
Q1 result

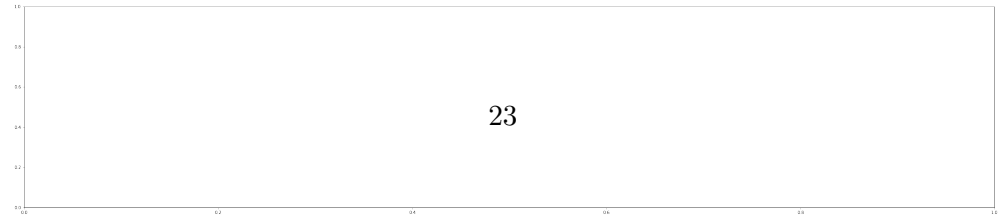

Smartphone
failed
weak
ok
good
excellent
Q1 result


glasses
failed
weak
ok
good
excellent
Q1 result

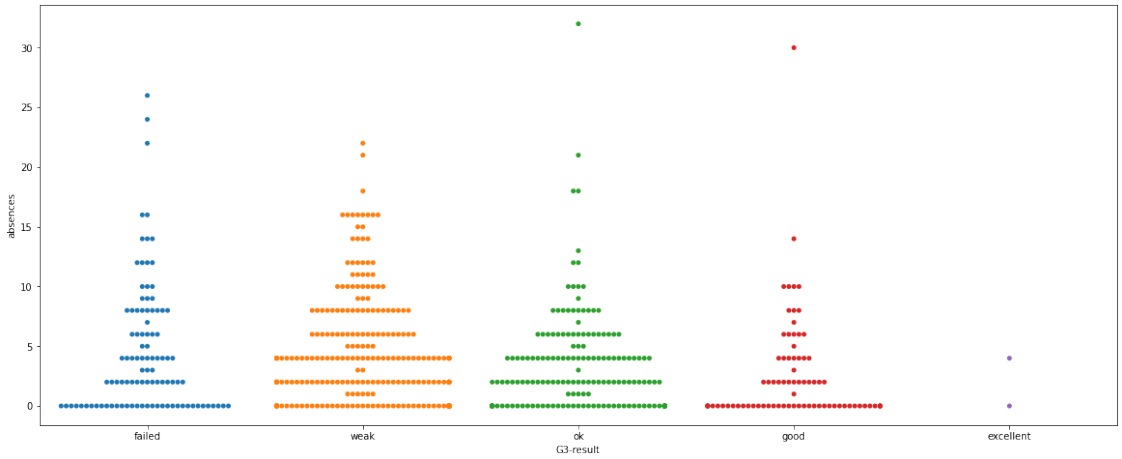

Tank
failed
weak
ok
good
excellent
Q1 result


Mic
failed
weak
ok
good
excellent
Q1 result

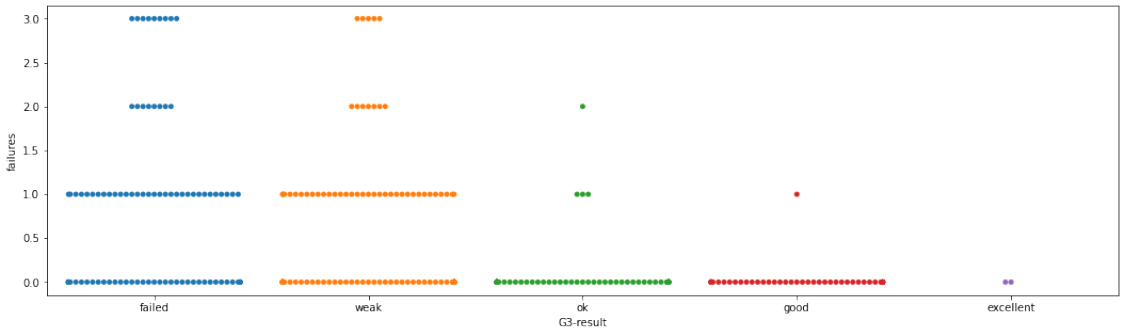

Buttons
failed
weak
ok
good
excellent
Q1 result



[24]:
```
fig, ax = plt.subplots(1, 1, figsize=(20, 8))
sns.swarmplot(data = grades_df, x = 'G3-result', y = 'absences',  order = order)
plt.show()
```



[25]:
```
fig, axs = plt.subplots(1, 1, figsize=(18, 5))
sns.swarmplot(data = grades_df, x = 'G3-result', y = 'failures',  order = order)
plt.show()
```



[ ]: