Projekt - Eksploracja Danych

Dane uczniów szkoły

Patryk Zając 297110

Do wykonania zadania użyłem środowiska Google Colab z językiem Python. Wszystkie wykorzystane biblioteki i pakiety są wymienione przy ich zastosowaniu poniżej.

1 Ocena jakości danych

Zacznijmy od wczytania danych z pliku csv:

```
import pandas as pd
dane = pd.read_csv('student-matA.csv', sep=';')
dane
```

Zanim zaczniemy analizować dane, sprawdzimy je pod kątem poprawności, czy nie występują w nich błędy, braki, duplikaty itp. Przyjrzyjmy się jakie wartości przyjmują poszczególne zmienne. Wykorzystamy funkcję `unique`, która zwraca listę unikalnych wartości danej zmiennej:

```
for kolumna in dane.columns:
    print(kolumna, ' : ', dane[kolumna].unique())
```

```
school  :  ['GP' 'MS']
sex  :  ['F' 'M']
age  :  [18 17 15 16 19 22 20 21]
address  :  ['U' 'R']
famsize  :  ['GT3' 'LE3']
Pstatus  :  ['A' 'T']
Medu  :  [4 1 3 2 0]
Fedu  :  [4 1 2 3 0]
Mjob  :  ['at_home' 'health' 'other' 'services' 'teacher']
Fjob  :  ['teacher' 'other' 'services' 'health' 'at_home']
reason  :  ['course' 'other' 'home' 'reputation']
guardian  :  ['mother' 'father' 'other']
traveltime  :  [2 1 3 4]
studytime  :  [2 3 1 4]
failures  :  [0 3 2 1]
schoolsup  :  ['yes' 'no']
famsup  :  ['no' 'yes']
paid  :  ['no' 'yes']
activities  :  ['no' 'yes']
nursery  :  ['yes' 'no']
higher  :  ['yes' 'no']
internet  :  ['no' 'yes']
romantic  :  ['no' 'yes']
famrel  :  [4 5 3 1 2]
freetime  :  [3 2 4 1 5]
goout  :  [4 3 2 1 5]
Dalc  :  [1 2 5 3 4]
Walc  :  [1 3 2 4 5]
health  :  [3 5 1 2 4]
absences  :  [ 6  4 10  2  0 16 14  7  8 25 12 54 18 26 20 56 24 28  5 13 15 22  3 21
  1 75 30 19  9 11 38 40 23 17]
G1  :  [ 5  7 15  6 12 16 14 10 13  8 11  9 17 19 18  4  3]
G2  :  [ 6  5  8 14 10 15 12 18 16 13  9 11  7 19 17  4  0]
G3  :  [ 6 10 15 11 19  9 12 14 16  5  8 17 18 13 20  7  0  4]
```

Nie widzimy żadnych braków danych, po za tym wartości przyjmowane przez zmienne pokrywają się z opisem danych z pliku student.txt. Dla tego zbioru nie musimy zajmować się duplikatami, ponieważ po przeanalizowaniu opisu danych wynika że mogą wystąpić dwa identyczne wpisy i będzie to jak najbardziej normalne.

2 Wybór predyktorów

Do wyboru predyktorów użyjemy testów **ANOVA** (dla zmiennych ciągłych) i testu **chi2** (dla zmiennych kategorycznych).

2.1 Zmienne ciągłe

Zanim przejdziemy do wykonania testu ANOVA musimy najpierw sprawdzić homoskedastyczność zmiennej celu wśród danej grupy, aby dobrać dobry rodzaj testu. Gdy rozkład zmiennej celu będzie równy wśród wszystkich grup to użyjemy testu **ANOVA**, w przeciwnym wypadku użyjemy testu **Welch ANOVA**. Musimy także wcześniej zainstalować **pakiet pingouin**.

```
pip install pingouin
```

2.1.1 Wiek

Najpierw sprawdzamy homoskedastyczność, określamy hipotezy:

- H0 - homoskedastyczność
- H1 - heteroskedastyczność

```
import random
import numpy as np
import pingouin as pg
from scipy.stats import ttest_ind, ttest_rel, f_oneway
from scipy import stats
hom = pg.homoscedasticity(data=dane, dv='age', group='G3')
print(hom)
```

```
                W      pval  equal_var
levene  0.671841  0.830885       True
```

Mamy p > 0.05 więc nie mamy podstaw by odrzucić H0, więc jest zachowana homoskedastyczność. W takim razie użyjemy testu ANOVA, określamy hipotezy:

- H0 - nie ma różnicy pomiędzy średnimi wartościami wieku wśród oceny G3
- H1 - jest różnica pomiędzy średnimi wartościami wieku wśród oceny G3

```
aov = pg.anova(data=dane, dv="age", between="G3")
print(aov)
```

	Source	ddof1	ddof2	F	p-unc	np2
0	G3	17	377	1.397433	0.133909	0.059279

Mamy p > 0.05 więc nie mamy podstaw aby odrzucić H0. Nie ma zatem różnicy pomiędzy średnimi wartościami wieku wśród oceny G3. Oznacza to, że wiek nie ma wpływu na ocenę końcową G3, więc nie wybieramy go jako predyktora do naszego modelu.

2.1.2 Nieobecności

Najpierw sprawdzamy homoskedastyczność, określamy hipotezy:

- H0 - homoskedastyczność
- H1 - heteroskedastyczność

```
hom = pg.homoscedasticity(data=dane, dv='G3', group='absences')
print(hom)
```

	W	pval	equal_var
levene	2.736703	0.000267	False

Mamy p < 0.05 więc odrzucamy H0, zachodzi więc heteroskedastyczność. Użyjemy w takim razie testu Welch ANOVA, określamy hipotezy:

- H0 - nie ma różnicy pomiędzy średnimi wartościami wieku wśród oceny G3
- H1 - jest różnica pomiędzy średnimi wartościami wieku wśród oceny G3

```
aov = pg.welch_anova(data=dane, dv="absences", between="G3")
print(aov)
```

	Source	ddof1	ddof2	F	p-unc	np2
0	G3	17	88.522357	17.793441	1.331298e-21	0.14663

Mamy p < 0.05 więc odrzucamy H0 na rzecz H1. Istnieje więc różnica pomiędzy średnimi wartościami nieobecności wśród oceny G3. Oznacza to, że nieobecności mają wpływ na ocenę końcową G3. Zatem nieobecności będą naszym predyktorem.

2.2 Zmienne kategoryczne

Przed uruchomieniem testów tworzymy odpowiednie tabele krzyżowe, które potem używamy w teście chi2, który sprawdza czy predyktor i zmienna celu są zależne od siebie czy nie. Jeśli zmienne będą zależne to daną zmienną wybierzemy jako predyktor do naszego modelu. Określamy hipotezy:

- H0 – nie ma zależności pomiędzy badaną zmienną a oceną G3
- H1 – jest zależność pomiędzy badaną zmienną a oceną G3

Tak będzie wyglądać kod wykonywanych działań (przykład dla zmiennej `school`, przy innych zmiennych zmieniamy tylko nazwę kolumny przy tworzeniu tabeli krzyżowej):

```
from scipy.stats import chi2_contingency
from scipy.stats import chi2
c_t = pd.crosstab(dane['school'], dane['G3'] ,margins = False)
stat, p, dof, expected = chi2_contingency(c_t)
print('dof=%d' % dof)
print('p_value', p)
prob = 0.95
critical = chi2.ppf(prob, dof)
print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical,
stat))
if p < 0.05:
    print('Zależne (odrzucamy hipotezę zerową)')
else:
    print('Niezależne (nie mamy podstaw aby odrzucić hipotezę
zerową)')
```

2.2.1 Szkoła

```
dof=17
p_value 0.8197135999934838
probability=0.950, critical=27.587, stat=11.670
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Rodzaj szkoły nie ma wpływu na ocenę końcową G3, więc szkoła do której chodzi student nie będzie predyktorem w naszym modelu.

2.2.2 Płeć

```
dof=17
p_value 0.26939988682273464
probability=0.950, critical=27.587, stat=20.095
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Płeć nie ma wpływu na ocenę końcową G3, więc płeć studenta nie będzie predyktorem w naszym modelu.

2.2.3 Miejsce zamieszkania

```
dof=17
p_value 0.06788170469842143
probability=0.950, critical=27.587, stat=26.376
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Miejsce zamieszkania nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.4 Rozmiar rodziny

```
dof=17
p_value 0.42283498376803474
probability=0.950, critical=27.587, stat=17.472
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Rozmiar rodziny nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.5 Status związku rodziców

```
dof=17
p_value 0.4897177548214079
probability=0.950, critical=27.587, stat=16.485
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Status związku rodziców nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.6 Edukacja matki

```
dof=68
p_value 0.25938557528758116
probability=0.950, critical=88.250, stat=75.096
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Edukacja matki nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.7 Edukacja ojca

```
dof=68
p_value 0.43930355096435997
probability=0.950, critical=88.250, stat=69.121
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Edukacja ojca nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.8 Praca matki

```
dof=68
p_value 0.028216255760662633
probability=0.950, critical=88.250, stat=91.942
Zależne (odrzucamy hipotezę zerową)
```

Praca matki ma wpływ na ocenę końcową G3, więc będzie ona predyktorem w naszym modelu.

2.2.9 Praca ojca

```
dof=68
p_value 0.5033281510657115
probability=0.950, critical=88.250, stat=67.238
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Praca ojca nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.10 Powód wybrania danej szkoły

```
dof=51
p_value 0.4560900441274288
probability=0.950, critical=68.669, stat=51.448
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Powód wybrania danej szkoły nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.11 Opiekun studenta

```
dof=34
p_value 0.8744565611942643
probability=0.950, critical=48.602, stat=24.835
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Rodzaj opiekuna studenta nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.12 Czas powrotu ze szkoły

```
dof=51
p_value 0.9774993200085966
probability=0.950, critical=68.669, stat=32.826
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Czas powrotu ze szkoły nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.13 Tygodniowy czas poświęcony na naukę

```
dof=51
p_value 0.19543507371441438
probability=0.950, critical=68.669, stat=59.432
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Tygodniowy czas poświęcony na naukę nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.14 Liczba niezdanych klas

```
dof=51
p_value 3.756699727487313e-09
probability=0.950, critical=68.669, stat=132.338
Zależne (odrzucamy hipotezę zerową)
```

Liczba niezdanych klas ma wpływ na ocenę końcową G3, więc będzie ona predyktorem w naszym modelu.

2.2.15 Dodatkowe wsparcie edukacyjne

```
dof=17
p_value 0.012967250942184287
probability=0.950, critical=27.587, stat=32.517
Zależne (odrzucamy hipotezę zerową)
```

Dodatkowe wsparcie edukacyjne ma wpływ na ocenę końcową G3, więc będzie ono predyktorem w naszym modelu.

2.2.16 Wsparcie edukacyjne ze strony rodziców

```
dof=17
p_value 0.688779619694134
probability=0.950, critical=27.587, stat=13.692
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Wsparcie edukacyjne ze strony rodziców nie ma wpływu na ocenę końcową G3, więc nie będzie ono predyktorem w naszym modelu.

2.2.17 Dodatkowo płatne zajęcia z przedmiotu

```
dof=17
p_value 0.028033354736395906
probability=0.950, critical=27.587, stat=29.772
Zależne (odrzucamy hipotezę zerową)
```

Dodatkowo płatne zajęcia z przedmiotu mają wpływ na ocenę końcową G3, więc będą one predyktorem w naszym modelu.

2.2.18 Zajęcia dodatkowe

```
dof=17
p_value 0.5612107675941844
probability=0.950, critical=27.587, stat=15.476
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Zajęcia dodatkowe nie mają wpływu na ocenę końcową G3, więc nie będą one predyktorem w naszym modelu.

2.2.19 Uczęszczanie do przedszkola

```
dof=17
p_value 0.5182861015520465
probability=0.950, critical=27.587, stat=16.078
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Uczęszczanie do przedszkola nie ma wpływu na ocenę końcową G3, więc nie będzie ono predyktorem w naszym modelu.

2.2.20 Chęć podjęcia studiów wyższych

```
dof=17
p_value 0.1094482874216139
probability=0.950, critical=27.587, stat=24.381
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Chęć podjęcia studiów wyższych nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.21 Dostęp do internetu w domu

```
dof=17
p_value 0.3920390653260133
probability=0.950, critical=27.587, stat=17.949
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Dostęp do internetu w domu nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.22 Bycie w romantycznej relacji

```
dof=17
p_value 0.02512766406485856
probability=0.950, critical=27.587, stat=30.172
Zależne (odrzucamy hipotezę zerową)
```

Bycie w romantycznej relacji ma wpływ na ocenę końcową G3, więc będzie ono predyktorem w naszym modelu.

2.2.23 Jakość relacji rodzinnych

```
dof=68
p_value 0.9688434167511698
probability=0.950, critical=88.250, stat=47.970
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Jakość relacji rodzinnych nie ma wpływu na ocenę końcową G3, więc nie będzie ona predyktorem w naszym modelu.

2.2.24 Czas wolny po szkole

```
dof=68
p_value 0.6372694409871895
probability=0.950, critical=88.250, stat=63.344
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Czas wolny po szkole nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.2.25 Wyjścia ze znajomymi

```
dof=68
p_value 0.2529418599128008
probability=0.950, critical=88.250, stat=75.346
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Wyjścia ze znajomymi nie mają wpływu na ocenę końcową G3, więc nie będą one predyktorem w naszym modelu.

2.2.26 Dzienne spożycie alkoholu

```
dof=68
p_value 0.21553610593340955
probability=0.950, critical=88.250, stat=76.882
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Dzienne spożycie alkoholu nie ma wpływu na ocenę końcową G3, więc nie będzie ono predyktorem w naszym modelu.

2.2.27 Weekendowe spożycie alkoholu

```
dof=68
p_value 0.1623863059208239
probability=0.950, critical=88.250, stat=79.407
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Weekendowe spożycie alkoholu nie ma wpływu na ocenę końcową G3, więc nie będzie ono predyktorem w naszym modelu.

2.2.28 Aktualny stan zdrowia

```
dof=68
p_value 0.436117517360381 23
probability=0.950, critical=88.250, stat=69.217
Niezależne (nie mamy podstaw aby odrzucić hipotezę zerową)
```

Aktualny stan zdrowia nie ma wpływu na ocenę końcową G3, więc nie będzie on predyktorem w naszym modelu.

2.3 Podsumowanie

Wybraliśmy za pomocą testów nasze predyktory, a są nimi:

- praca matki
- liczba niezdanych klas
- dodatkowe wsparcie edukacyjne
- dodatkowo płatne zajęcia z przedmiotu
- bycie w romantycznej relacji
- nieobecności

3 Podział danych

Najpierw sprawdźmy czy w zbiorze danych nie występują pojedyncze wystąpienia oceny G3 (uniemożliwi to równomierny podział na dane uczące i testowe):

```
dane['G3'].value_counts()
```

```
10    56
11    47
0     38
15    33
8     32
13    31
12    31
9     28
14    27
16    16
6     15
18    12
7      9
5      7
17     6
19     5
20     1
4      1
Name: G3, dtype: int64
```

Jak widzimy G3 = 4 i G3 = 20 występują tylko raz. Aby móc równomiernie podzielić zbiór danych, usuniemy te rekordy, a następnie oddzielimy zmienną celu od reszty zmiennych:

```
dane.drop(dane[(dane['G3'] == 20) | (dane['G3'] == 4)].index, inplace
= True)
X = dane.iloc[:,:-1] # wszystkie wiersze, wszystkie kolumny poza
ostatnią
y = dane['G3']
```

Następnie przechodzimy do podziału danych na uczące i testowe:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=297110, stratify = y)
```

Nasze dane zostały podzielone, 0.7 danych do zbioru uczącego i 0.3 danych do zbioru testowego (z zachowaniem mniej więcej równych proporcji jeśli chodzi o rozłożenie ocen G3 w zbiorach `y_train` i `y_test` (argument `stratify`)). Teraz zostawiamy w zbiorach tylko nasze wybrane wcześniej na podstawie testów predyktory:

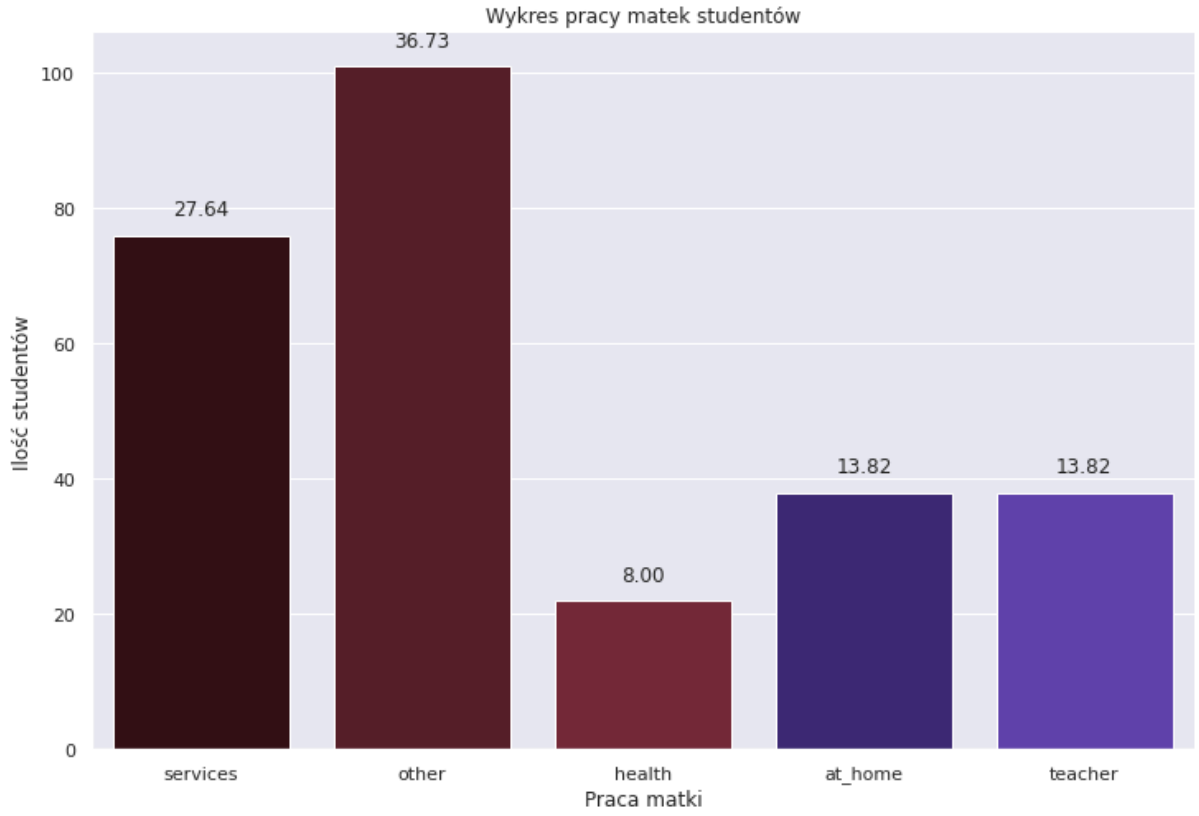
```
X_train =
X_train[['Mjob','failures','schoolsup','paid','romantic','absences']]
X_test =
X_test[['Mjob','failures','schoolsup','paid','romantic','absences']]
```

4 Eksploracyjna analiza danych

4.1 Analiza pojedynczych zmiennych

Przeanalizujemy wykresy dla pojedynczych zmiennych. Użyjemy do tego wykresów słupkowych i histogramów.

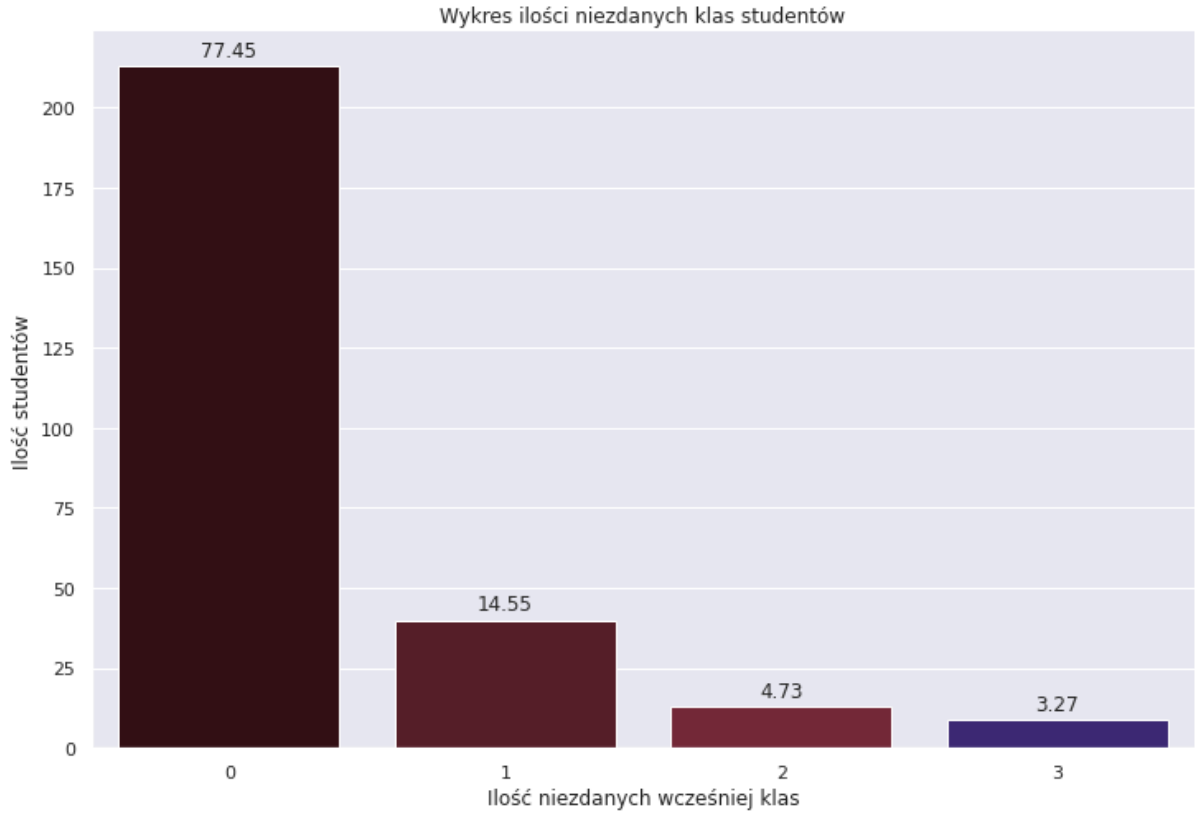
4.1.1 Praca matki



Powyższy wykres pokazuje że:

- mamy pięć unikalnych wartości: services, other, health, at_home, teacher
- najwięcej matek pracuje w innym sektorze niż pozostałe 4 wymienione - 36.73%
- najmniej matek pracuje w sektorze zdrowia - 8%

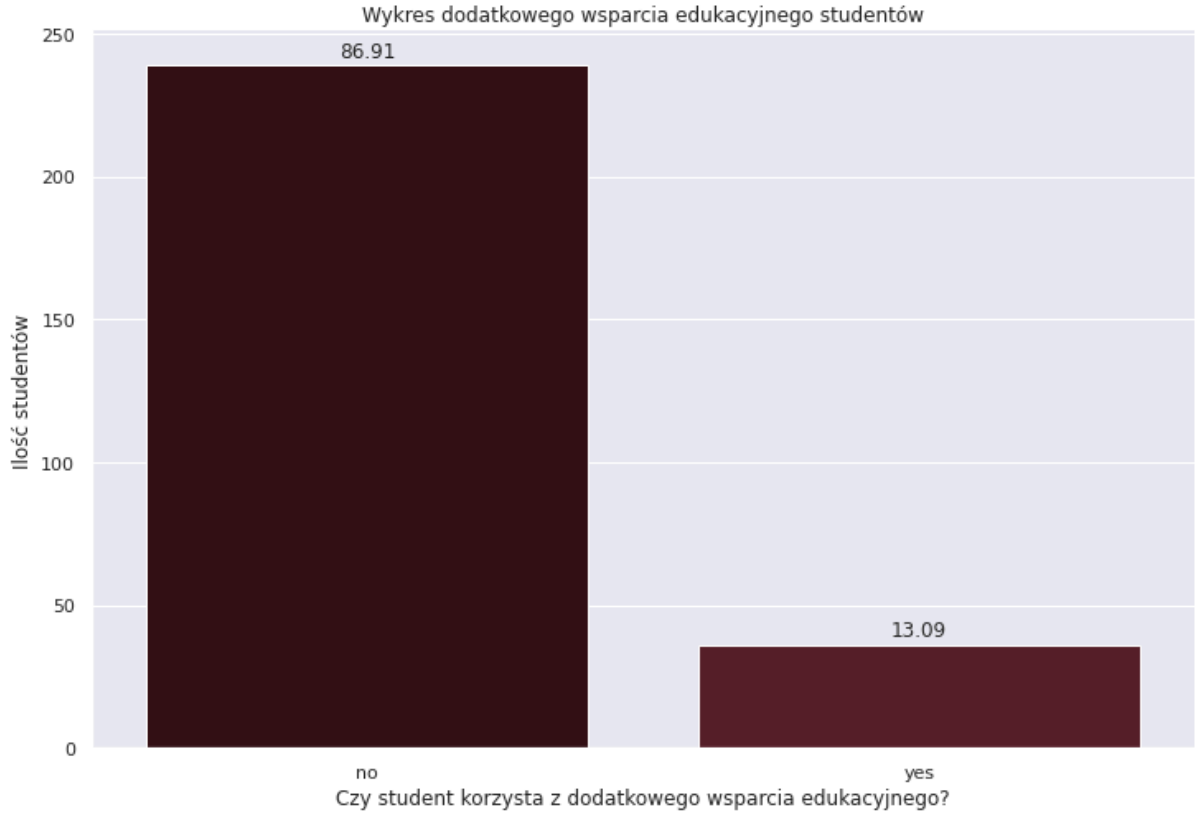
4.1.2 Ilość niezdanych klas



Powyższy wykres pokazuje że:

- mamy 4 unikalne wartości, określające ilość niezdanych klas: 0, 1, 2 i 3
- najwięcej mamy studentów, którzy nie musieli powtarzać żadnej klasy - 77.45%
- mamy zaledwie kilku studentów którzy powtarzali aż 3 klasy, stanowią oni 3.25%
- Istnieje ogromna nierównowaga w kategoriach ilości niezdanych klas

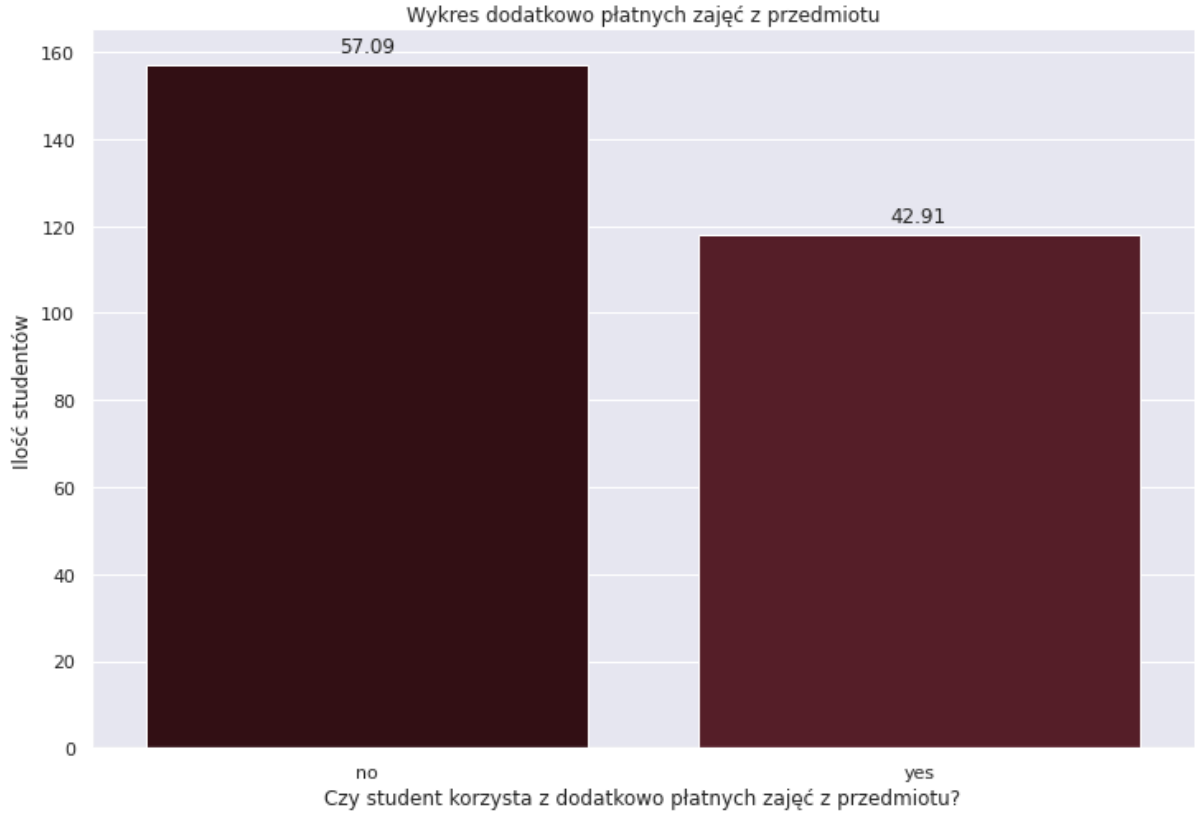
4.1.3 Dodatkowe wsparcie edukacyjne



Powyższy wykres pokazuje że:

- mamy 2 unikalne wartości - tak i nie
- najwięcej mamy studentów, którzy nie korzystają z dodatkowego wsparcia edukacyjnego - 86.91%
- studenci którzy korzystają z dodatkowego wsparcia edukacyjnego są w zdecydowanej mniejszości, stanowią oni 13.09%

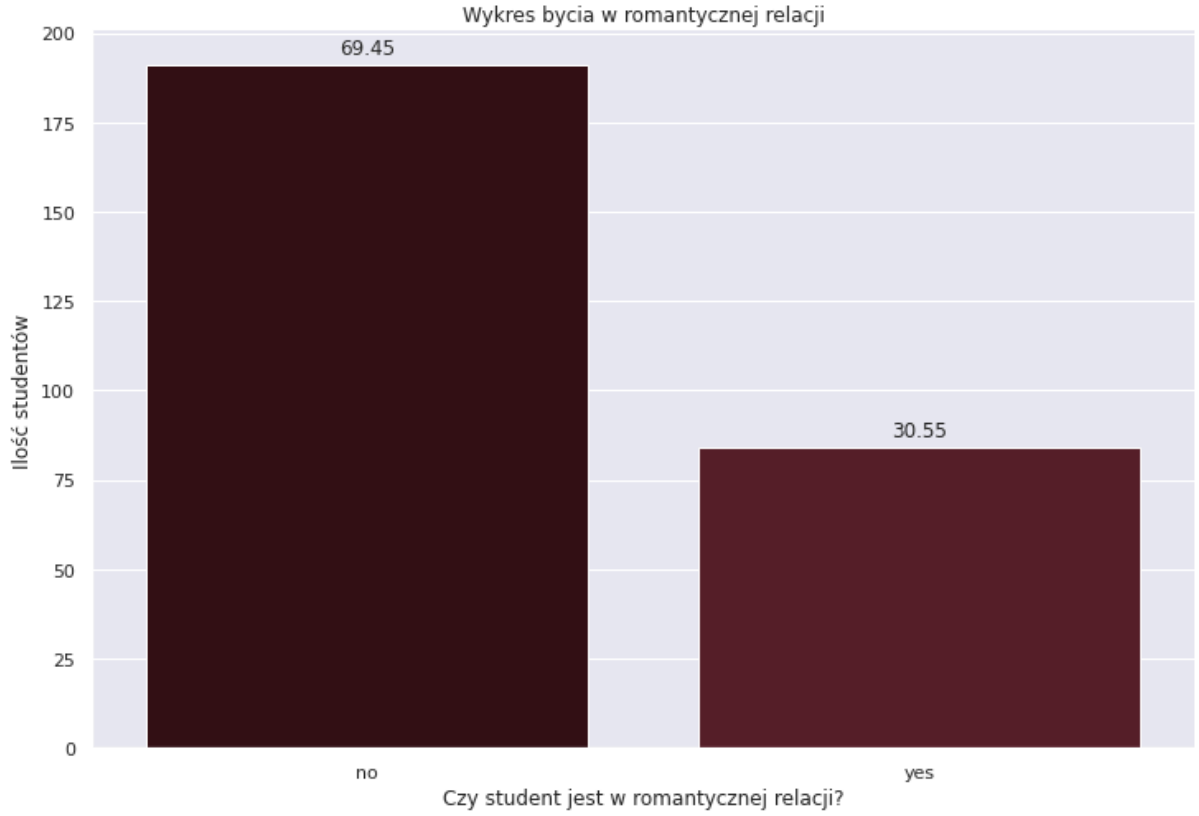
4.1.4 Dodatkowo płatne zajęcia z przedmiotu



Powyższy wykres pokazuje że:

- mamy 2 unikalne wartości - tak i nie
- więcej jest studentów którzy nie korzystają z dodatkowo płatnych zajęć z przedmiotu - 57.09%
- ten zestaw danych jest mniej więcej zbalansowany jeśli chodzi o ilość osób które korzystają lub nie korzystają z dodatkowo płatnych zajęć - 42.91% do 57.09%

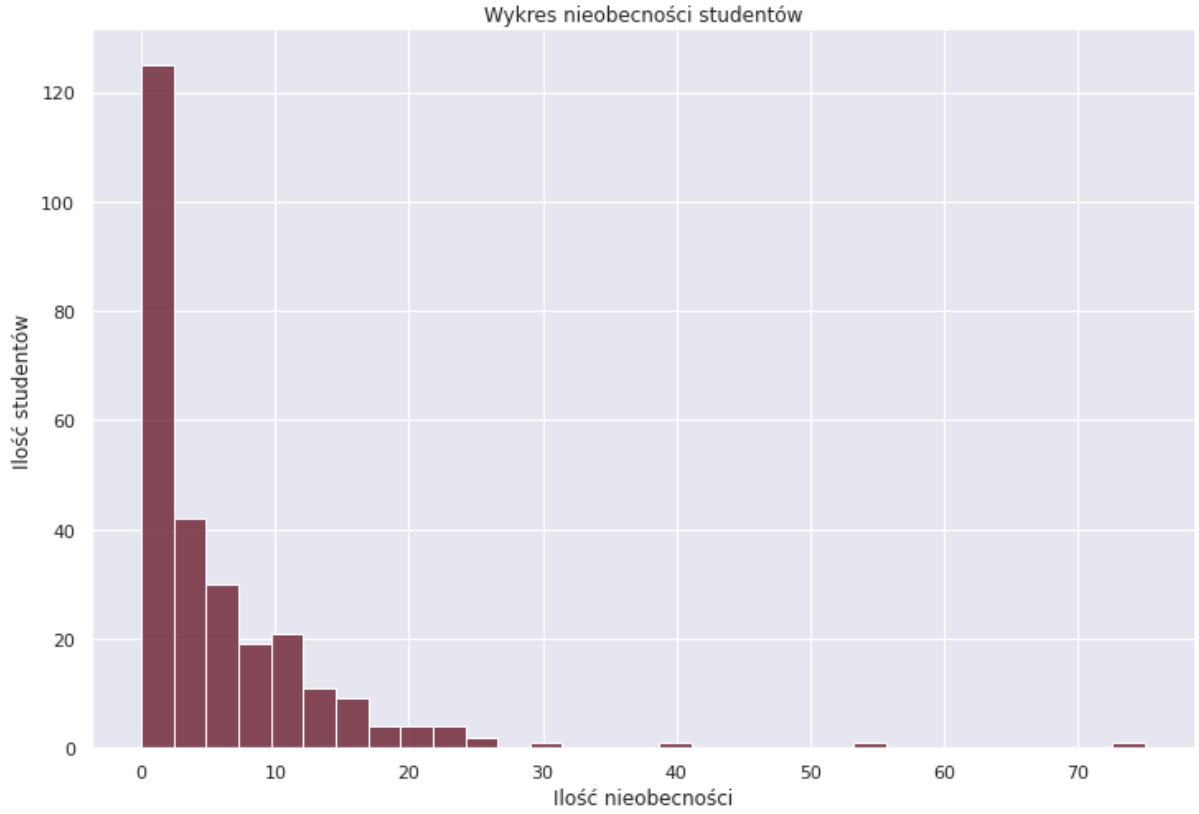
4.1.5 Bycie w romantycznej relacji



Powyższy wykres pokazuje że:

- mamy 2 unikalne wartości - tak i nie
- więcej jest studentów którzy nie posiadają drugiej połówki - 69.45%
- studenci którzy są w romantycznej relacji stanowią 30.55%

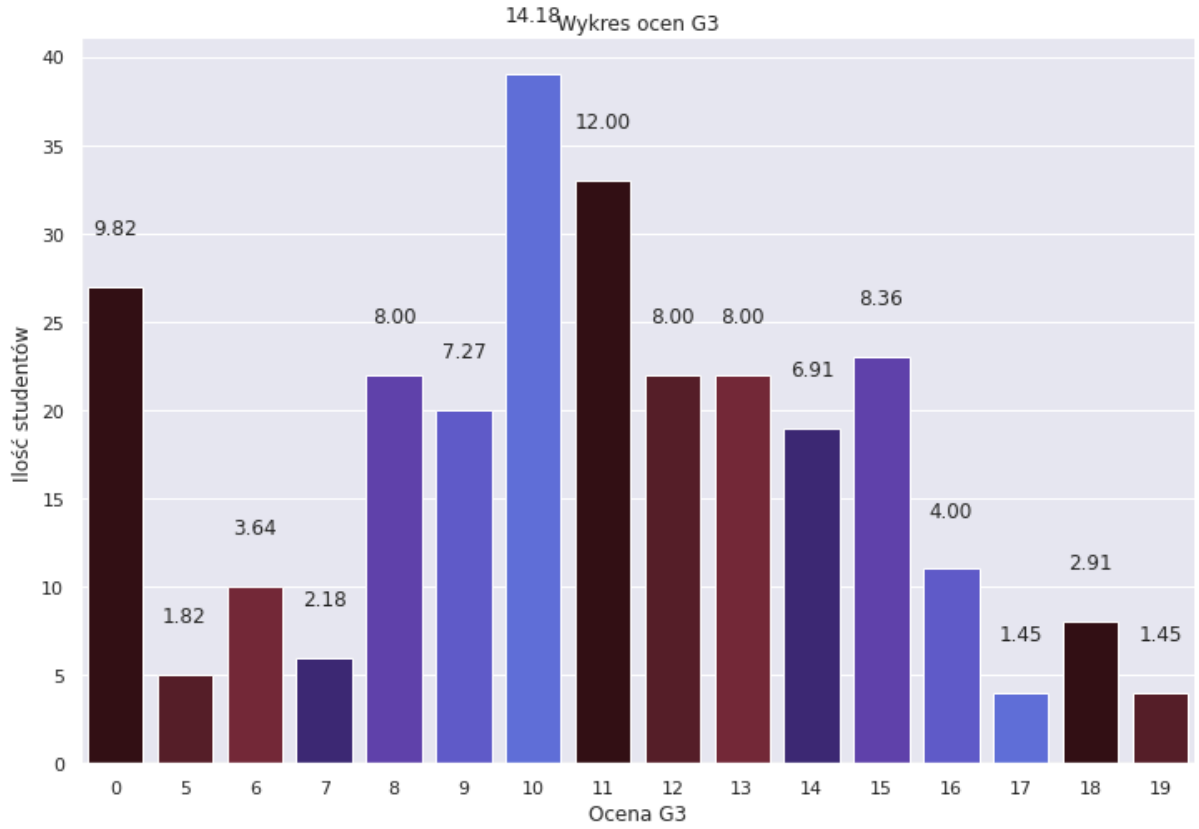
4.1.6 Nieobecności



Powyższy wykres pokazuje że:

- zakres nieobecności jest od 0 do 75
- zdecydowana większość studentów posiada nieobecności w zakresie 0 do 10
- nieobecności powyżej 20 są rzadkością
- mamy też kilka obserwacji mocno wybiegających poza średnią, są to m.in 40, 50 czy 75
- wykres jest prawostronnie skośny, ale jest to jak najbardziej normalne dla tego zbioru, w szkole raczej większość uczniów chodzi regularnie na zajęcia

4.1.7 Ocena G3



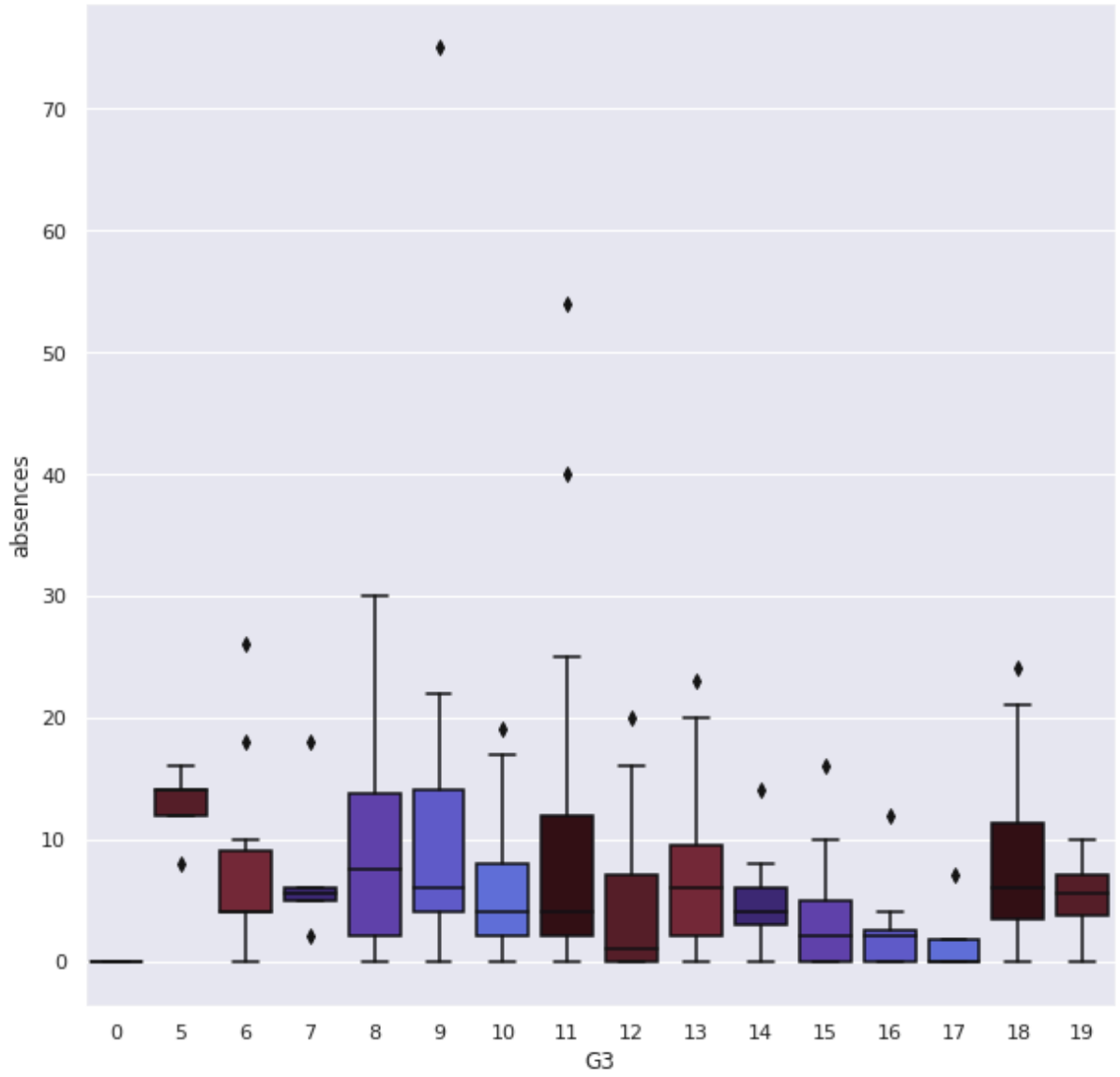
Powyższy wykres pokazuje że:

- mamy 16 unikalnych wartości - 0 i od 5 do 19
- najwięcej jest studentów którzy posiadają ocenę 10 - 14.18%
- najmniej jest studentów którzy posiadają ocenę 17 - 1.45%
- można by lepiej zbalansować zbiór danych jeśli chodzi o tę kategorię, dodając więcej obserwacji z oceną poniżej 8 i z oceną powyżej 16

4.2 Analiza predyktorów ze zmienną celu

W celu tej analizy wykorzystamy wykresy pudełkowe i zestawione wykresy słupkowe.

4.2.1 Nieobecności

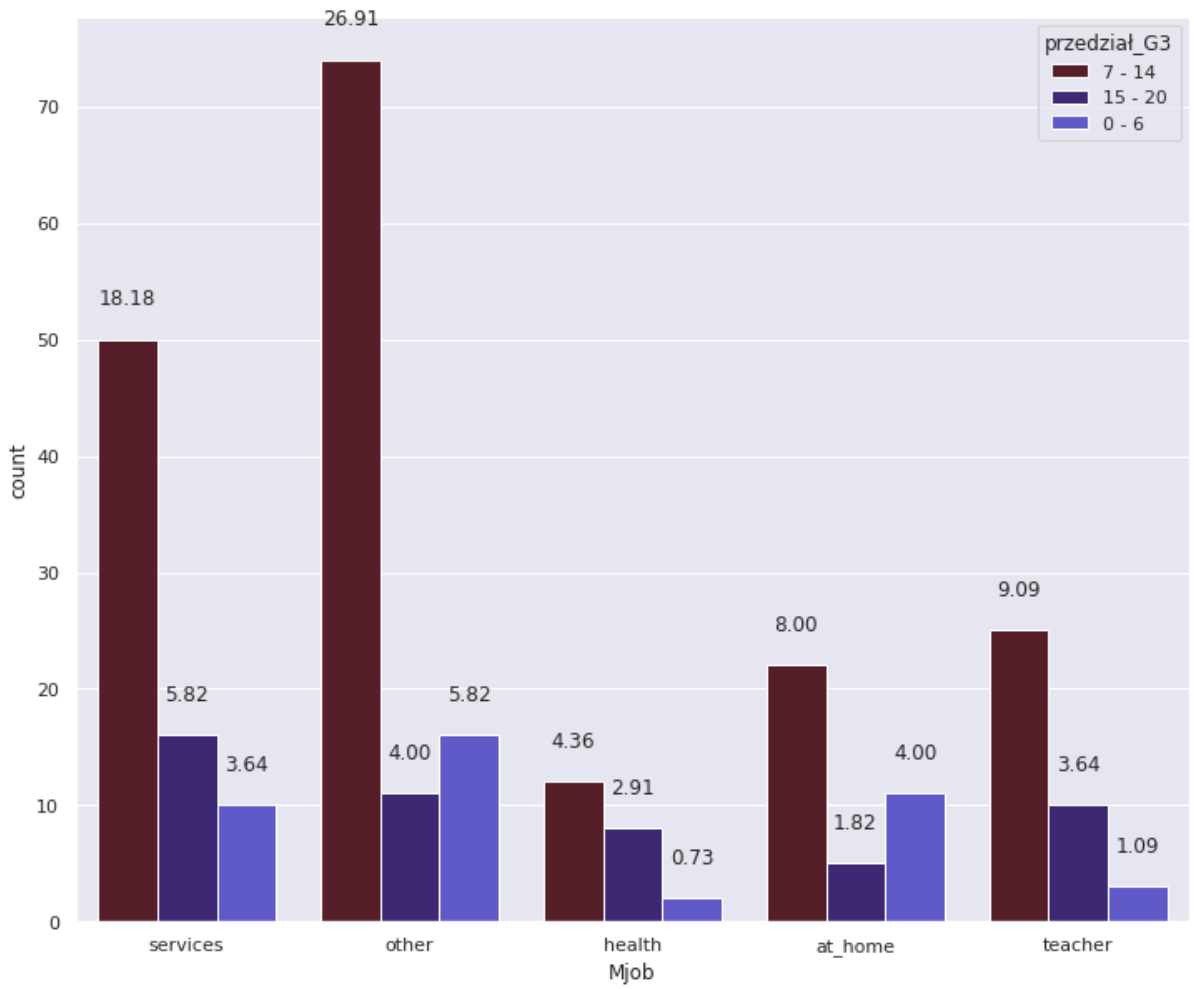


Z powyższego wykresu wynika:

- wartości odstające dla nieobecności występują u osób z każdą oceną poza 0, 8, 9 i 19
- największa odstająca wartość znajduje się w grupie osób z oceną 9
- najwyższą medianę nieobecności ma grupa osób z oceną 8, najniższą natomiast (nie licząc oceny 0 gdzie nieobecności są równe 0) grupa osób z oceną 12
- najszerszy rozstęp kwartylny nieobecności jest w grupie osób z oceną 8, najwęższy z 7

4.2.2 Praca matki

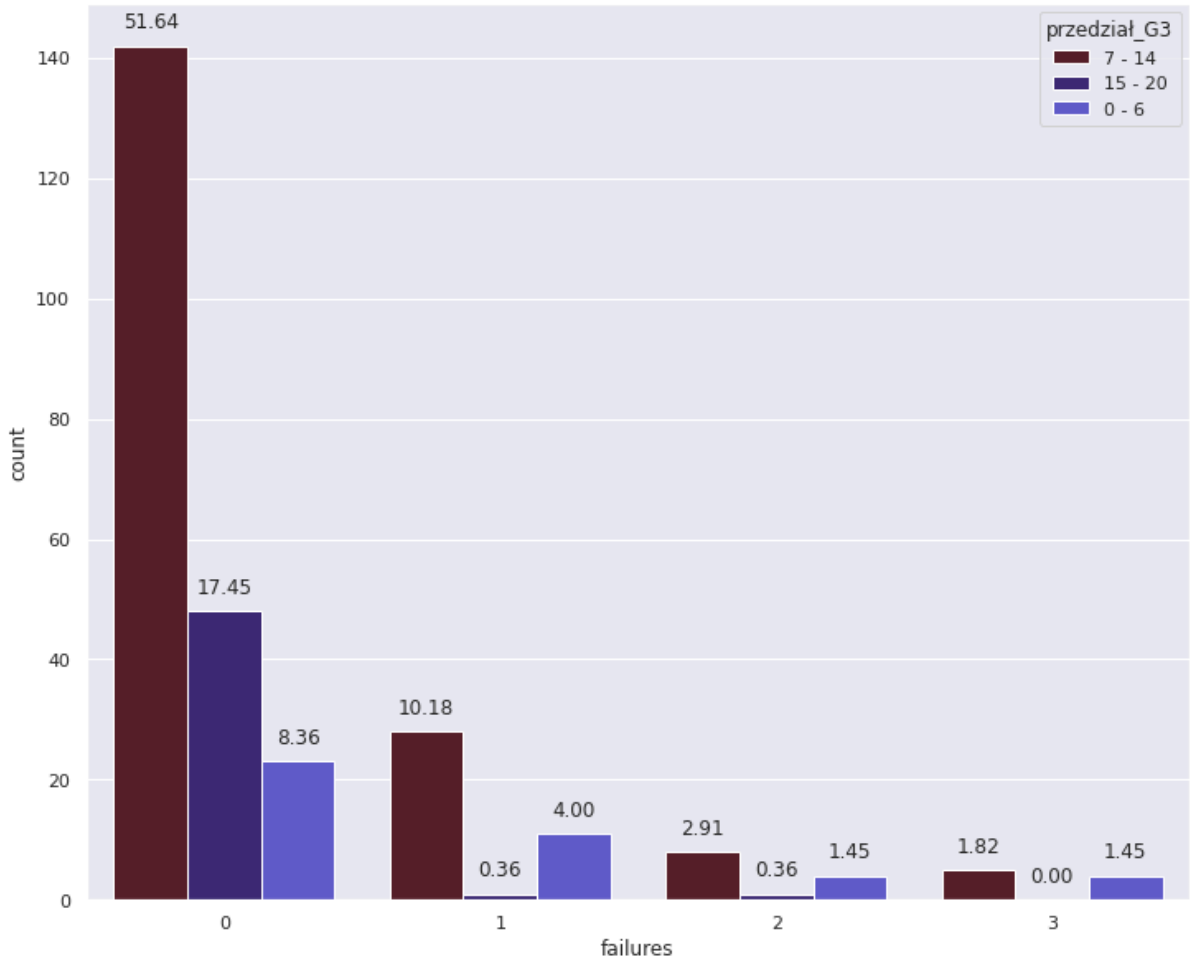
W celu lepszej przejrzystości wykresów, oceny G3 zostały pogrupowane w 3 grupy



Z powyższego wykresu wynika:

- najwięcej przeciętnie uczących się uczniów posiada matkę pracującą w innym sektorze niż wymienione, stanowią oni 26.91% badanych
- najwięcej słabo uczących się uczniów również posiada matkę pracującą w innym sektorze niż wymienione, stanowią oni 5.82% badanych
- najwięcej dobrze uczących się uczniów posiada matkę pracującą sektorze usługowym, stanowią oni 5.82% badanych

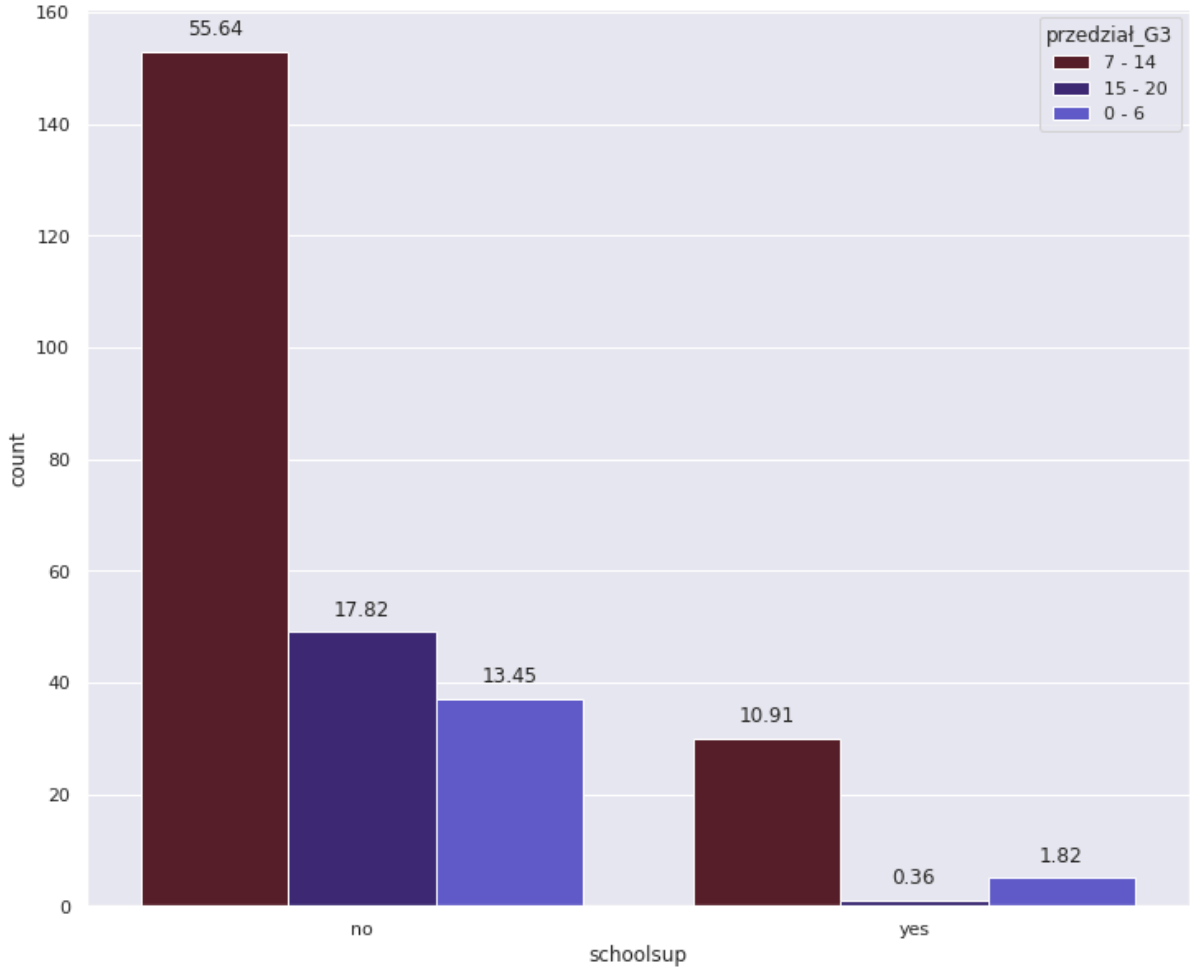
4.2.3 Liczba niezdanych klas



Z powyższego wykresu wynika:

- dla każdej grupy ocen, studenci najczęściej nie powtarzali żadnej klasy
- widzimy że ze wzrostem ilości niezdanych klas, maleje ilość studentów każdej grupy
- dobrze uczący się uczniowie praktycznie nie powtarzali żadnej klasy (co jest całkiem logiczne), ale są małe wyjątki od tej reguły, wystąpiły bowiem osoby które powtórzyły jedną czy dwie klasy, mając obecnie świetne oceny
- osoby które w przeszłości powtarzały jakąkolwiek klasę, obecnie najczęściej mają przeciętne oceny

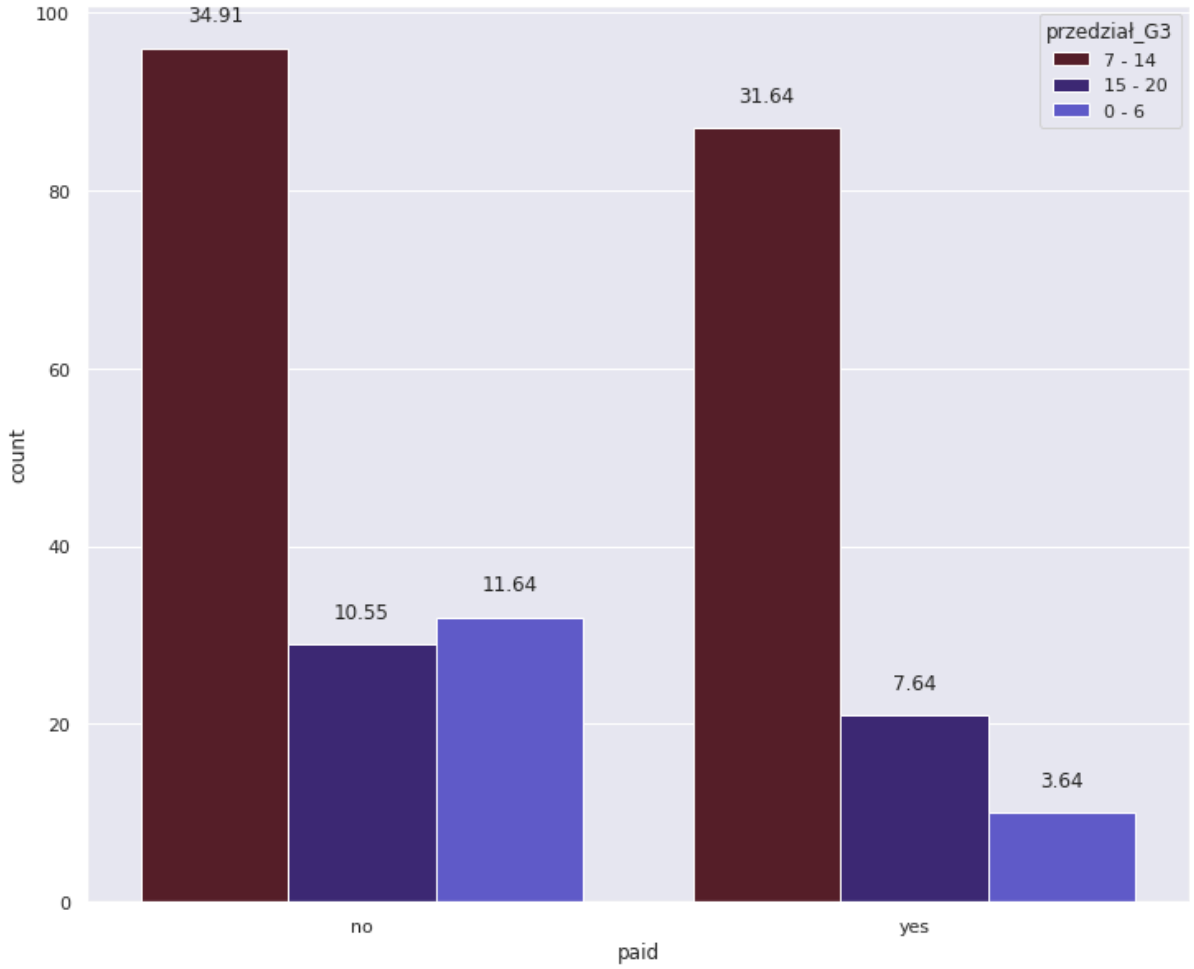
4.2.4 Dodatkowe wsparcie edukacyjne



Z powyższego wykresu wynika:

- studenci z każdej grupy ocen najczęściej nie podejmują dodatkowego wsparcia edukacyjnego
- jeżeli już ktoś podejmuje dodatkowe wsparcie edukacyjne to są to studenci uzyskujący przeciętne oceny - 10.91% badanych
- najrzadziej dodatkowe wsparcie edukacyjne podejmują osoby o świetnych ocenach - 0.36%

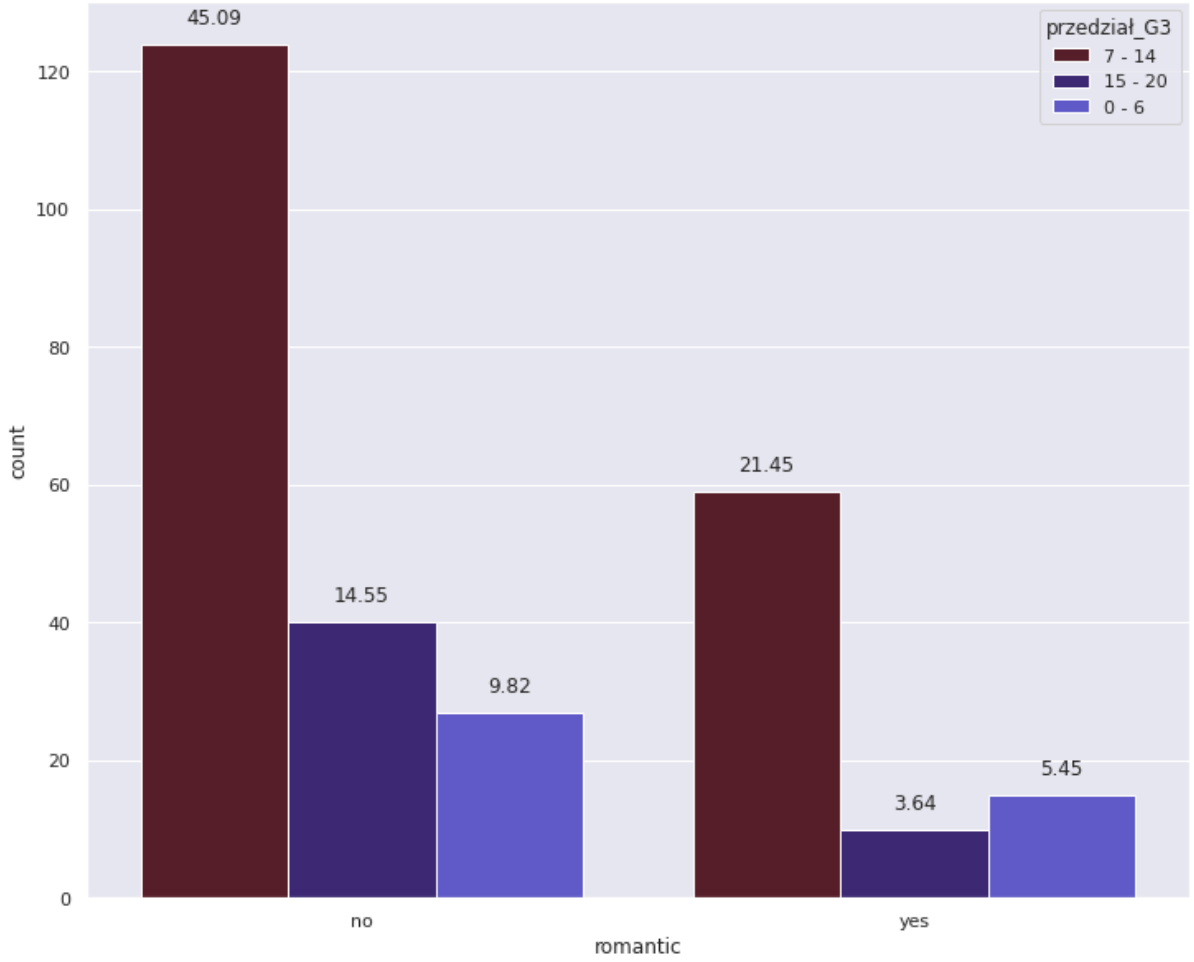
4.2.5 Dodatkowo płatne zajęcia z przedmiotu



Z powyższego wykresu wynika:

- studenci z każdej grupy ocen najczęściej nie podejmują dodatkowo płatnych zajęć z przedmiotu
- jeżeli już ktoś podejmuje to są to studenci uzyskujący przeciętne oceny - 31.64% badanych
- najrzadziej dodatkowe płatne zajęcia z przedmiotu podejmują osoby o słabych ocenach - 3.64%

4.2.6 Bycie w romantycznej relacji



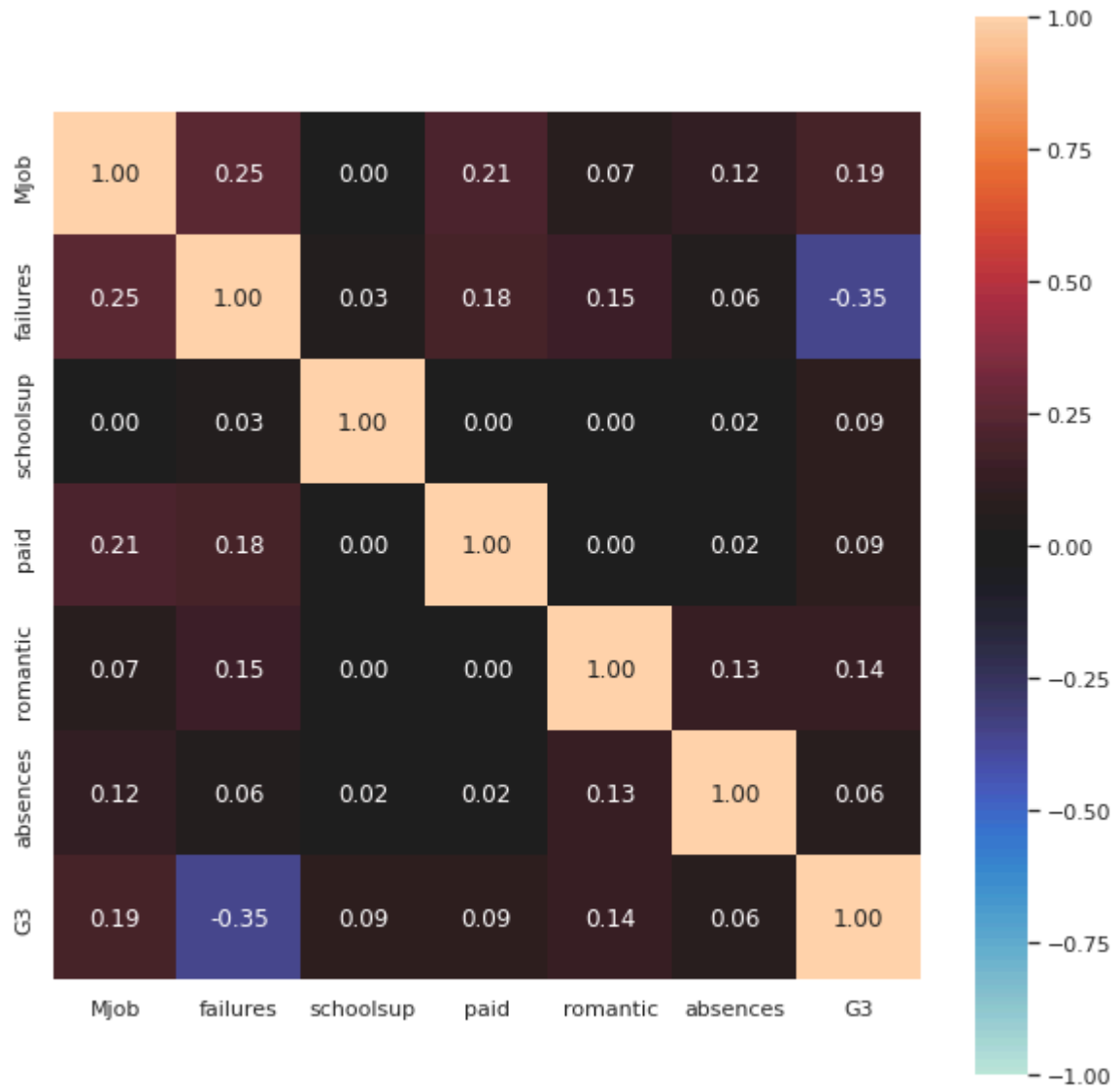
Z powyższego wykresu wynika:

- studenci z każdej grupy ocen najczęściej nie są w związku
- jeżeli już ktoś ma drugą połówkę to są to studenci uzyskujący przeciętne oceny - 21.45% badanych
- najrzadziej w romantycznej relacji są osoby o świetnych ocenach - 3.64%

4.3 Korelacje

Do zbadania które zmienne mogą mieć największy wpływ na ocenę G3 oraz na zbadania korelacji wykorzystamy mapę ciepła (używam pakietu dython, który pozwala na utworzenie mapy ciepła jednocześnie dla zmiennych ciągłych i kategorycznych)

```
from dython.nominal import associations
associations(dane_treningowe.iloc[:,:-1], figsize=(10,10))
```



Jak widzimy największy wpływ na ocenę G3 może mieć zmienna failures (wraz ze wzrostem oceny, spada ilość powtarzanych klas - korelacja ujemna równa 0.35), zaraz po niej mamy Mjob - dodatnia korelacja - 0.19 i romantic - dodatnia korelacja - 0.14. Jeżeli chodzi o korelacje pomiędzy pozostałymi zmiennymi, to największą wartość mamy dla failures i Mjob, są one dodatnio skorelowane - 0.25, więc może istnieć jakaś zależność pomiędzy liczbą niezdanych klas a pracą matki, ale nie jest to też wysoka wartość. Widzimy też dodatnią korelację równą 0.21 dla pracy matki i braniem dodatkowo płatnych zajęcia z przedmiotu (nie jest to także duża wartość, ale mogą być od siebie zależne, jeśli matka pracuje w lepszym sektorze i zarabia więcej to może to powodować że student jest w stanie na takie zajęcia chodzić).

5 Model klasyfikacyjny

Do wykonania modelu klasyfikacji użyjemy lasów losowych. Najpierw zajmiemy się przygotowaniem zmiennych do ich dalszego używania w modelu klasyfikacji i szacowania. Zmienną failures w analizie danych traktowaliśmy jako zmienną kategoryczną, więc zamienimy jej wartości na

typ string, żeby potem móc ją zastosować w metodzie **pd.get_dummies**. Zamieniamy dane kategoryczne na dane wskaźnikowe (powstanie nam więcej zmiennych, każda odpowiadająca na to czy dana wartość zmiennej kategorycznej występuje czy nie). Aby uniknąć silnej korelacji pomiędzy jedną zmienną wskaźnikową a resztą, usuniemy ją ze zbioru predyktorów (nie wpłynie to na wyniki, gdyż wszystkie informacje pozostają zachowane w reszcie zmiennych):

```
X_train['failures'] = X_train['failures'].apply(str)
X_test['failures'] = X_test['failures'].apply(str)
X_train = pd.get_dummies(data=X_train, drop_first=True)
X_test = pd.get_dummies(data=X_test, drop_first=True)
```

Następnie standaryzujemy zmienną absences:

```
from scipy import stats
X_train['absences'] = stats.zscore(np.log(X_train['absences'] + 1))
X_test['absences'] = stats.zscore(np.log(X_test['absences'] + 1))
```

Następnie przechodzimy do budowy modelu. Zbudujemy model oparty na lasach losowych:

```
from sklearn.ensemble import RandomForestRegressor
forest_reg1 = RandomForestRegressor(random_state=297110)
forest_reg1.fit(X_train, y_train)
y_train_pred = forest_reg1.predict(X_train)
y_test_pred = forest_reg1.predict(X_test)
```

Zaokrąglamy wyniki:

```
y_train_pred = np.round(y_train_pred)
y_test_pred = np.round(y_test_pred)
```

Definiujemy funkcję która będzie odpowiedzialna za ocenę jakości modelu:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
from math import sqrt
from sklearn.metrics import confusion_matrix, accuracy_score

def ocen_model(y_true, y_pred, digits = 3):
    with_deviation = 0
    for i, j in zip(y_true, y_pred):
        if i == j:
          with_deviation += 1
        if i == j+1:
          with_deviation += 1
        if i == j-1:
          with_deviation += 1
```

```
    dev_accuracy = with_deviation / y_true.size
    print("Trafność: ", round(100*accuracy_score(y_true, y_pred),
digits),'%')
    print("Trafność z dopuszczalnym odstępstwem o 1: ",
round(100*dev_accuracy, digits),'%')
    print('Średni błąd bezwzględny: ',
round(mean_absolute_error(y_true, y_pred), digits))
```

Sprawdzamy wyniki dla zbioru testowego:

```
print('Zbiór testowy')
ocen_model(y_test, y_test_pred)
```

```
Zbiór testowy
Trafność:  11.017 %
Trafność z dopuszczalnym odstępstwem o 1:  29.661 %
Średni błąd bezwzględny:  3.305
```

Możemy spróbować poprawić te wyniki za pomocą konfiguracji hiper parametrów:

```
from sklearn.model_selection import GridSearchCV
hyperparameters = {'max_depth' : range(1,10), 'min_samples_split' :
[10, 20, 50]}
best_forest_reg = GridSearchCV(RandomForestRegressor(random_state =
297110, max_features = 1.0), hyperparameters, n_jobs = -1)
best_forest_reg.fit(X_train, y_train)
best_forest_reg.best_params_
```

```
{'max_depth': 9, 'min_samples_split': 20}
```

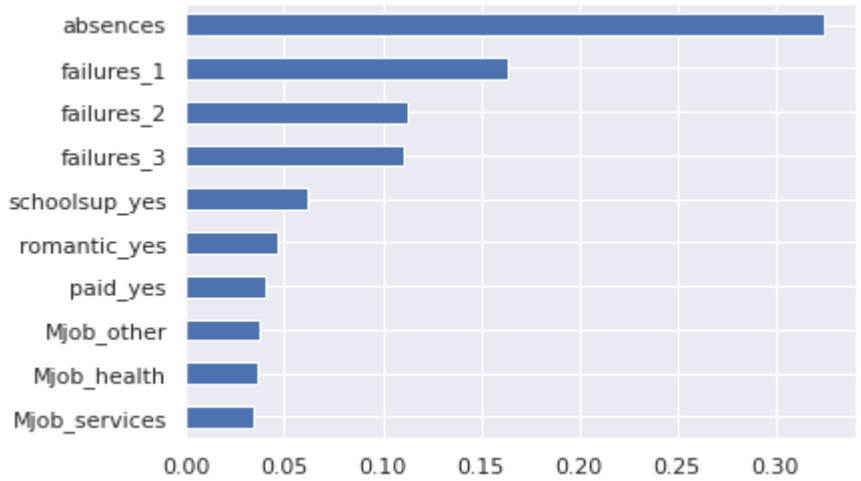
```
y_train_pred = best_forest_reg.predict(X_train)
y_test_pred = best_forest_reg.predict(X_test)
y_train_pred = np.round(y_train_pred)
y_test_pred = np.round(y_test_pred)
print('Zbiór testowy')
ocen_model(y_test, y_test_pred)
```

```
Zbiór testowy
Trafność:  11.864 %
Trafność z dopuszczalnym odstępstwem o 1:  29.661 %
Średni błąd bezwzględny:  3.017
```

Jak widzimy udało nam się poprawić trafność o około 0.8% i zredukować średni błąd bezwzględny o około 0.3. Sprawdźmy teraz które predyktory nasz model uznał za najważniejsze:

```
def waznosc_predyktorow(drzewo):
```

```
    waznosci = pd.Series(drzewo.feature_importances_,
index=X_train.columns)
    waznosci.sort_values(inplace=True)
    waznosci.iloc[-10:].plot(kind='barh', figsize=(6,4))
waznosc_predyktorow(best_forest_reg.best_estimator_)
```



Model za najważniejszą uznał zmienną **absences**, następnie trochę mniej ważne od absences są zmienne obrazujące ilość powtarzanych klas.

6 Model szacowania

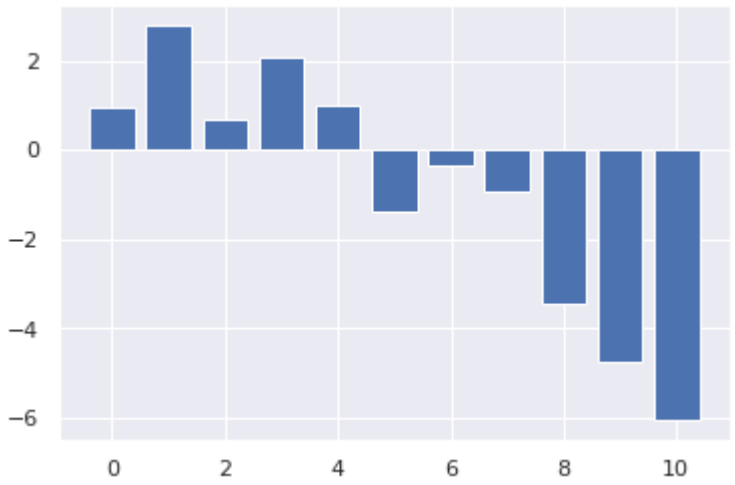
Przy modelu szacowania użyjemy klasy **LinearRegression** z modułu *sklearn.linear_model*:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

Zobaczmy jakie wartości przyjmują współczynniki modelu i jak układają się na wykresie, stąd możemy wywnioskować które zmienne są najbardziej istotne w naszym modelu regresji liniowej:

```
pd.DataFrame(model.coef_, index = model.feature_names_in_, columns =
['wspolczynnik modelu'])
from matplotlib import pyplot
pyplot.bar([x for x in range(len(model.coef_))], model.coef_)
pyplot.show()
```

	wspolczynnik modelu
absences	0.960849
Mjob_health	2.798218
Mjob_other	0.657961
Mjob_services	2.048000
Mjob_teacher	1.009256
schoolsup_yes	-1.393990
paid_yes	-0.368247
romantic_yes	-0.926814
failures_1	-3.460999
failures_2	-4.737826
failures_3	-6.067690



Widzimy że najważniejsze w naszym modelu są zmienne obrazujące ilość powtarzanych klas, zaraz po nich najważniejsza jest zmienna Mjob_health. Zobaczmy jak wygląda równanie naszego modelu:

```
print('G3 = ', end='')
for b,n in zip(model.coef_, model.feature_names_in_):
  print(round(b,2), '*', n, '+ ',end='')
  print(round(model.intercept_,2))
```

```
G3 = 0.96 * absences + 10.78
2.8 * Mjob_health + 10.78
0.66 * Mjob_other + 10.78
2.05 * Mjob_services + 10.78
1.01 * Mjob_teacher + 10.78
-1.39 * schoolsup_yes + 10.78
-0.37 * paid_yes + 10.78
-0.93 * romantic_yes + 10.78
-3.46 * failures_1 + 10.78
-4.74 * failures_2 + 10.78
-6.07 * failures_3 + 10.78
```

Wyznaczamy teraz współczynnik $\mathbf{R}^2$:

```
model.score(X_train,y_train)
```

```
0.23069816544883548
```

Widzimy, że jego wartość jest dość wysoka, jednak może być zbyt optymistyczna i dlatego wyznaczymy też **skorygowany współczynnik determinacji**, który określa się wzorem:

$$R^2 = 1 - (1 - R^2)\frac{n}{n-p}$$

```
def adjusted_R2(model, X, y):
  r2 = model.score(X,y)
  n, p = X.shape
  return 1 - (1-r2)*n/(n-p)
adjusted_R2(model, X_train, y_train)
```

```
0.19864392234253692
```

Wartość skorygowanego współczynnika jest nieco niższa.

Sprawdzimy teraz jakość predykcji naszego modelu:

```
y_train_pred_szac = model.predict(X_train)
y_test_pred_szac = model.predict(X_test)
y_train_pred_szac = np.round(y_train_pred_szac)
y_test_pred_szac = np.round(y_test_pred_szac)
print('Zbiór testowy')
ocen_model(y_test, y_test_pred_szac)
```

```
Zbiór testowy
Trafność:  13.559 %
Trafność z dopuszczalnym odstępstwem o 1:  33.898 %
Średni błąd bezwzględny:  3.153
```

Jak wiemy, nasz model za najważniejsze uznał zmienne failures. Możemy za pomocą klasy **SequentialFeatureSelector** z modułu *sklearn.feature_selection* dobrać mniejszą ilość predyktorów żeby poprawić wyniki naszego modelu. Wykonałem parę testów i znalazłem zbiór który daje lepsze wyniki. Zawiera on dalej najważniejsze zmienne uznane przez model, czyli failures, a po za tym również: absences, Mjob_health, Mjob_services, Mjob_teacher, romantic_yes. Czyli żeby poprawić wyniki naszego modelu regresji liniowej należy usunąć Mjob_other, paid_yes oraz schoolsup_yes. Nasz nowy model powstał w następujący sposób:

```
from sklearn.feature_selection import SequentialFeatureSelector
selector2 = SequentialFeatureSelector(model, n_features_to_select =
8, direction = 'backward')
selector2.fit(X_train, y_train)
selector2.get_feature_names_out()
```

```
array(['absences', 'Mjob_health', 'Mjob_services', 'Mjob_teacher',
       'romantic_yes', 'failures_1', 'failures_2', 'failures_3'],
      dtype=object)
```

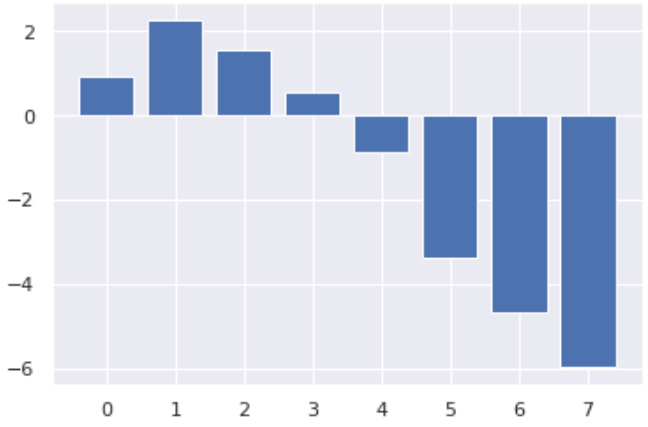
Sprawdźmy jakie teraz daje wyniki:

```
y_train_pred_szac3 = model3.predict(X3_train)
y_test_pred_szac3 = model3.predict(X3_test)
y_train_pred_szac3 = np.round(y_train_pred_szac3)
y_test_pred_szac3 = np.round(y_test_pred_szac3)
print('Zbiór testowy')
ocen_model(y_test, y_test_pred_szac3)
```

```
Zbiór testowy
Trafność:  17.797 %
Trafność z dopuszczalnym odstępstwem o 1:  33.898 %
Średni błąd bezwzględny:  3.093
```

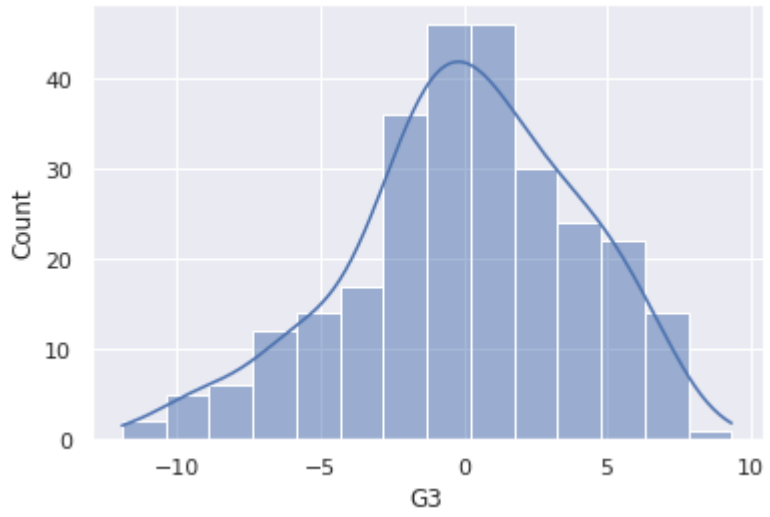
Udało nam się poprawić trafność o ponad **4%**! Zredukował się także średni błąd bezwzględny - o **0.06**. Poniżej ważność zmiennych ostatecznego modelu regresji liniowej:

	wspolczynnik modelu
absences	0.947563
Mjob_health	2.278935
Mjob_services	1.545814
Mjob_teacher	0.555303
romantic_yes	-0.883275
failures_1	-3.354777
failures_2	-4.658058
failures_3	-5.981035

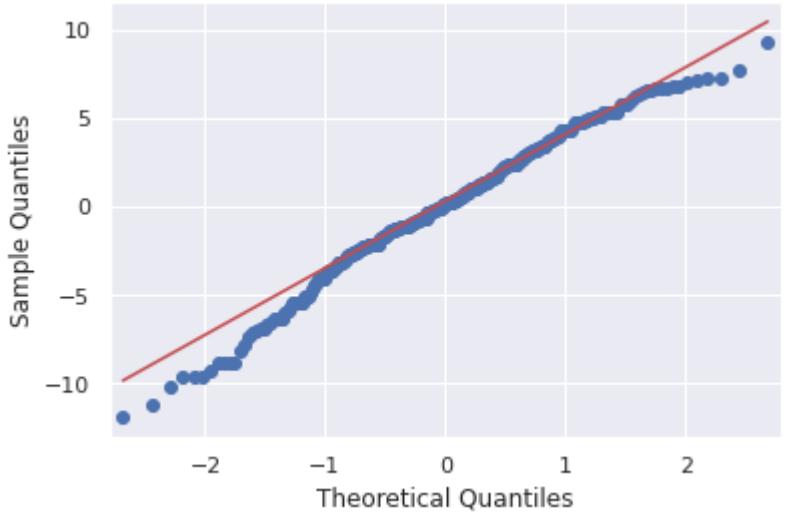


Sprawdzimy teraz założenia naszego ostatecznego modelu. Zaczniemy od sprawdzenia najpierw czy reszty (błędy modelu) mają rozkład normalny ze średnią 0 i stałą wariancją. Narysujemy **histogram**, **wykres kwantyl-kwantyl** i wykonamy **test Shapiro-Wilka**.

```
import seaborn as sns
yy_pred = model3.predict(X3_train)
reszty = y_train - yy_pred
sns.histplot(x=reszty, kde=True)
```




```
import statsmodels.api as sm
sm.qqplot(data = reszty, line='q');
```




```
from scipy.stats import shapiro
shapiro(reszty)
```

```
ShapiroResult(statistic=0.9849081039428711, pvalue=0.005385767202824354)
```

Histogram nie jest zbliżony do histogramu rozkładu normalnego, jest trochę lewoskośny. Jeżeli chodzi o wykres Q-Q to niebieskie punkty układają się mniej więcej wzdłuż linii czerwonej co sugeruje nam, że rozkład reszt jest normalny. Hipotezą zerową testu była normalność rozkładu, jak widzimy, p-wartość testu jest mniejsza od 0.05 (czyli standardowego poziomu istotności). Odrzucamy hipotezę zerową, więc **rozkład reszt nie jest normalny**.

Zbadamy teraz niezależność reszt za pomocą **testu Durbina-Watsona**:

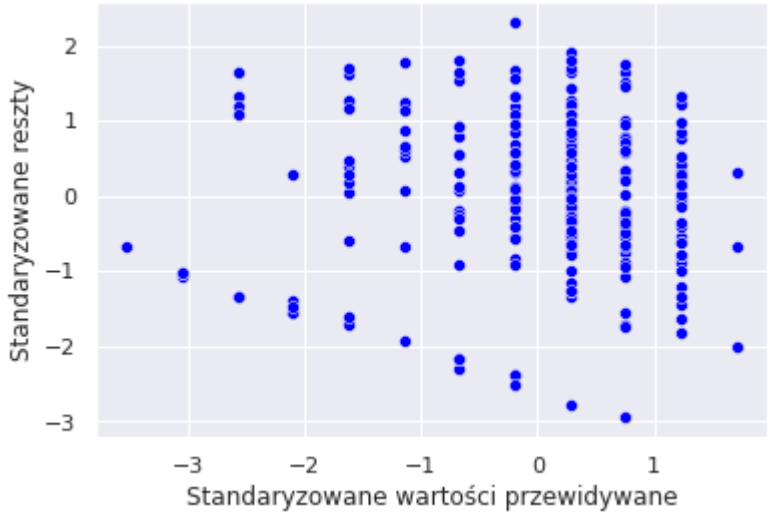
```
from statsmodels.stats.stattools import durbin_watson
durbin_watson(reszty)
```

```
2.059710961357093
```

Zastosujemy tzw. **regułę kciuka**, która mówi, że wartość statystyki testowej powinna należeć do przedziału [1.5, 2.5]. Nasz wynik nie wykracza poza ten przedział, co świadczy o tym, że **reszty nie są ze sobą skorelowane**.

Teraz zajmiemy się homoskedastycznością reszt, czyli równością ich wariancji. Sprawdzamy ją rysując **wykres rozrzutu standaryzowanych reszt względem standaryzowanych wartości przewidywanych**. Jeżeli wariancje są równe na wykresie nie powinny być widoczne żadne wyraźne wzorce.

```
import matplotlib.pyplot as plt
from scipy.stats import zscore
sns.scatterplot(x = zscore(y_train_pred_szac3), y=zscore(reszty),
color = 'blue')
plt.xlabel('Standaryzowane wartości przewidywane')
plt.ylabel('Standaryzowane reszty')
```



Nie są widoczne żadne wzorce, **zachowana jest homoskedastyczność**.

Sprawdźmy jeszcze czy w zbiorze występują obserwacje odstające:

```
abs(zscore(reszty)) > 3
```

Widzimy, że **obserwacji odstających nie ma**.

7 Porównanie modeli

- Model klasyfikacji (**lasy losowe**)

```
Zbiór testowy
Trafność:  11.864 %
Trafność z dopuszczalnym odstępstwem o 1:  29.661 %
Średni błąd bezwzględny:  3.017
```

- Model szacowania (wieloraka regresja liniowa, klasa **LinearRegression**)

```
Zbiór testowy
Trafność:  17.797 %
Trafność z dopuszczalnym odstępstwem o 1:  33.898 %
Średni błąd bezwzględny:  3.093
```

Oba modele nie dają jakiś rewelacyjnych wyników, trafność nie przekracza dla obu nawet 20%, trafność z odstępstwem daje już lepsze wyniki, ale dalej są one raczej przeciętne.

Zdecydowanie lepszy okazał się model regresji liniowej, trafność lepsza o około 6%, trafność z dopuszczalnym odstępstwem o 1 lepsza o około 4%. Co ciekawe, to model klasyfikacji osiągnął lepszy wynik dla średniego błędu bezwzględnego, błąd jest niższy o 0.076

Przy modelu klasyfikacji najważniejszą zmienną była zmienna absences, natomiast przy modelu szacowania zmienna failures. W modelu klasyfikacji zmienna failures natomiast była brana pod uwagę na drugi miejscu. Natomiast w modelu szacowania, zmienna absences była jedną z najmniej ważnych zmiennych.

Podczas eksploracyjnej analizy danych zauważyliśmy że największy wpływ na ocenę G3 może mieć zmienna failures, potem Mjob a najmniej ważna jest zmienna absences. Model szacowania potwierdza to założenie, sam bowiem uznał zmienną failures za najważniejszą w naszym modelu. Inaczej sprawa się ma co do modelu klasyfikacji. Przy badaniu korelacji zauważyliśmy że absences jest zmienną najsłabiej skorelowaną ze zmienną celu, ale w modelu klasyfikacyjnym ta zmienna jest najbardziej istotna.

8 Grupowanie

Najpierw zdefiniujemy funkcję do wizualizacji grup i do wizualizacji wykresu sylwetki:

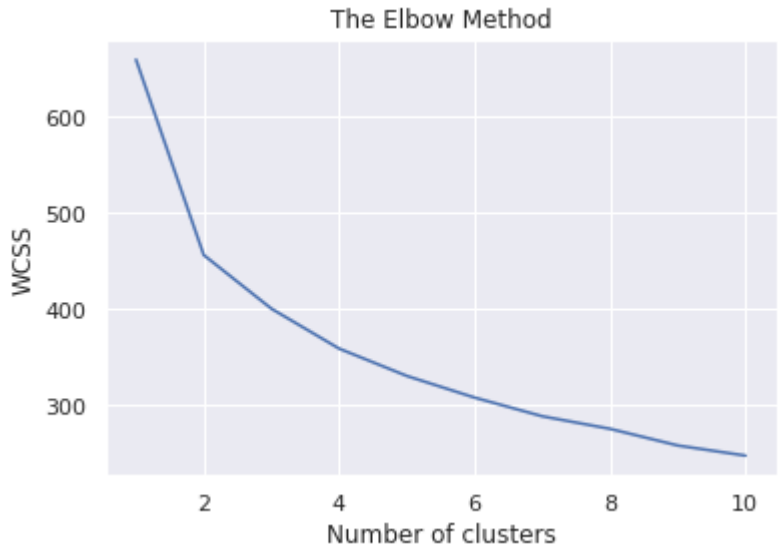
```
import matplotlib.pyplot as plt
import seaborn as sns

def wizualizuj_grupy(model):
```

```
  for i in range(model.n_clusters):
    plt.figure(figsize = (15,3))
    pd.Series(model.cluster_centers_[i], index =
model.feature_names_in_).plot(kind = 'bar')
    plt.title('Grupa ' + str(i))
    plt.color_sequences = ["#611423","#361b82","#5248db"]
```

```
def silhouette_plot(X, y_pred, n_clusters):
  fig, ax1 = plt.subplots(1, 1)
  fig.set_size_inches(18, 7)
  ax1.set_xlim([-0.1, 1])
# The (n_clusters+1)*10 is for inserting blank space between
silhouette
# plots of individual clusters, to demarcate them clearly.
  ax1.set_ylim([0, X.shape[0] + (n_clusters + 1) * 10])
# The silhouette_score gives the average value for all the samples.
# This gives a perspective into the density and separation of the
formed
# clusters
  silhouette_avg = silhouette_score(X, y_pred)
  print("For n_clusters =", n_clusters, "The average silhouette_score
is :", silhouette_avg)
# Compute the silhouette scores for each sample
  sample_silhouette_values = silhouette_samples(X, y_pred)
  y_lower = 10
  for i in range(n_clusters):
# Aggregate the silhouette scores for samples belonging to
# cluster i, and sort them
    ith_cluster_silhouette_values = sample_silhouette_values[y_pred
== i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i
    color = cm.nipy_spectral(float(i) / n_clusters)
    ax1.fill_betweenx(np.arange(y_lower, y_upper),0,
ith_cluster_silhouette_values,facecolor=color, edgecolor=color,
alpha=0.7)
# Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
# Compute the new y_lower for next plot
    y_lower = y_upper + 10 # 10 for the 0 samples
  ax1.set_title("Wykres sylwetki")
  ax1.set_xlabel("Wartość współczynnika sylwetki")
  ax1.set_ylabel("Numer klastra")
# The vertical line for average silhouette score of all the values
  ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
  ax1.set_yticks([]) # Clear the yaxis labels / ticks
  ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
```

Teraz za pomocą **metody Elbow**, zobaczymy na ile będzie najbardziej optymalnie podzielić nasz zbiór (szukamy miejsca w którym coraz to większa ilość klastrów nie poprawia znacząco wyników):

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
                    random_state = 297110)
    kmeans.fit(X_train)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



Podzielimy nasz zbiór na 4 grupy:

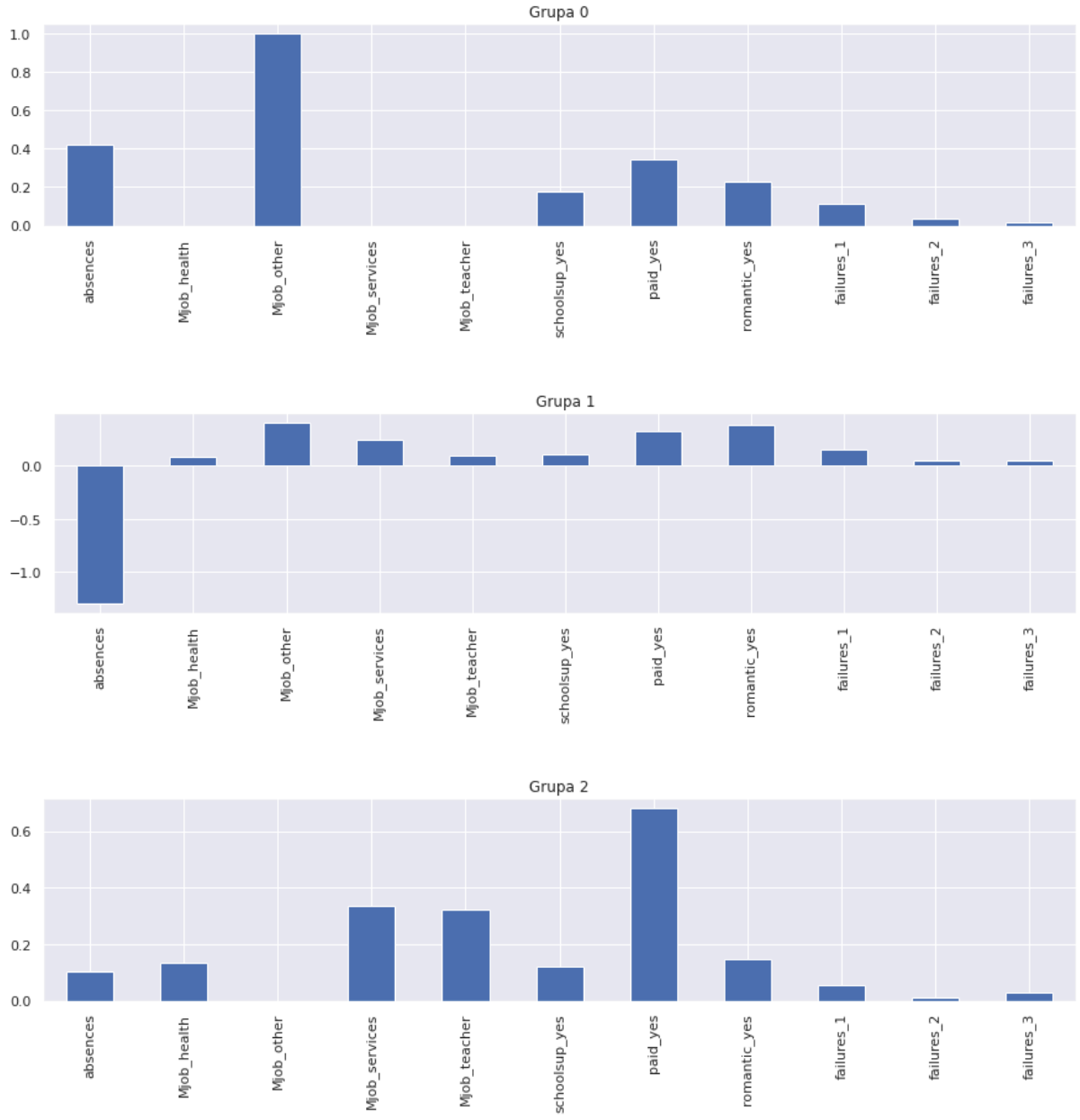
```
km4 = KMeans(n_clusters = 4, random_state = 297110)
km4.fit(X_train)
km4.predict(X_train)
print(km4.score(X_train))
print(km4.inertia_)
```

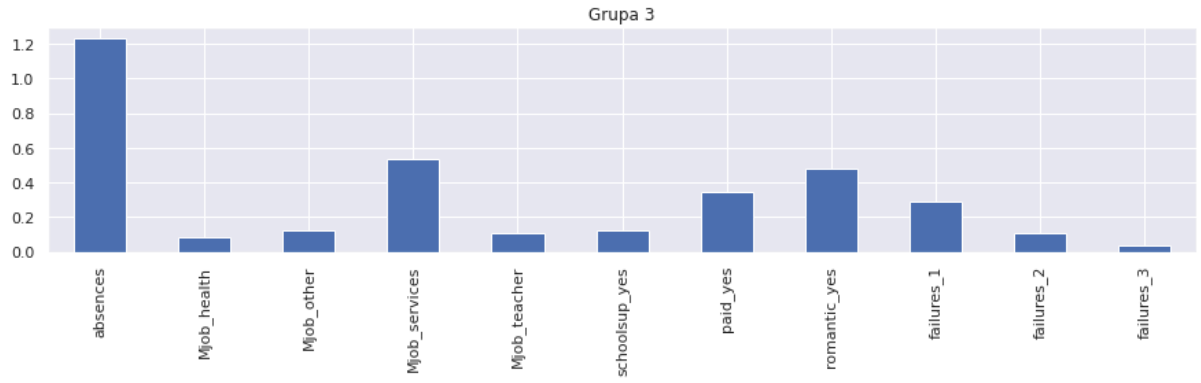
```
-358.93237695042285
358.93237695042285
```

```
y4 = km4.predict(X_train)
pd.Series(y4).value_counts().sort_index()
```

```
0    61
1    81
2    75
3    58
dtype: int64
```

```
wizualizuj_grupy(km4)
```





Otrzymaliśmy następujące profile:

- Grupa 0 - są to studenci których matka pracuje w sektorze innym niż wymienione w zbiorze danych, mają też trochę nieobecności w szkole i czasem pobierają dodatkowe płatne zajęcia z przedmiotu
- Grupa 1 - są to studenci którzy regularnie chodzą do szkoły
- Grupa 2 - są to studenci którzy pobierają dodatkowe płatne zajęcia z przedmiotu, a ich matka najczęściej pracuje w usługach lub jako nauczyciel
- Grupa 3 - są to studenci którzy mają dużo nieobecności, często też ich matka pracuje w usługach i są oni często w romantycznej relacji

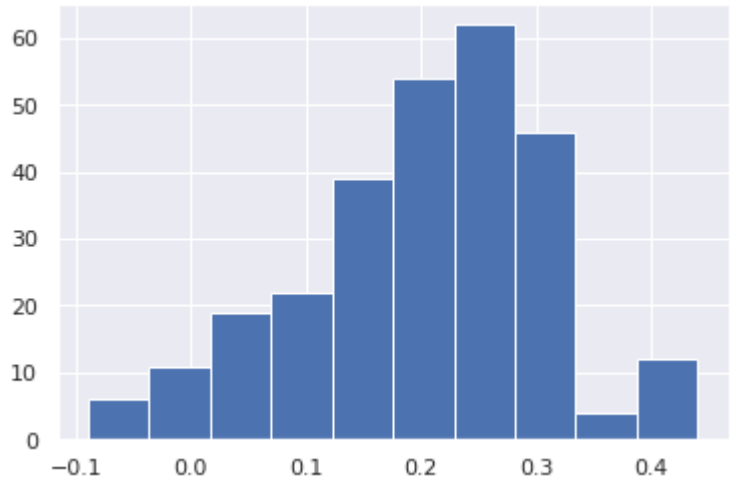
Sprawdźmy teraz jakość naszego grupowania:

```
from sklearn.metrics import silhouette_samples, silhouette_score
print('dla k = 4, s =', silhouette_score(X_train, y4))
```

```
dla k = 4, s = 0.20456746779476717
```

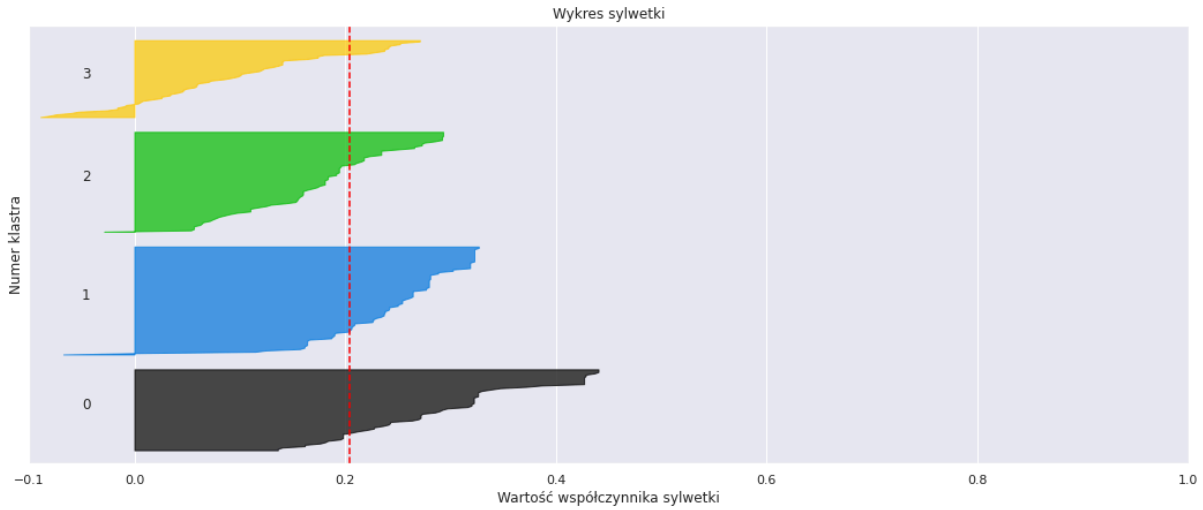
Wynik powyżej 0.2 można uznać za poprawny, choć nie rewelacyjny. Sprawdźmy wyniki dla poszczególnych obserwacji:

```
pd.Series(silhouette_samples(X_train, y4)).hist()
```



Widzimy, że zdarzają się obserwacje dla, których wynik jest ujemny (co sugeruje, że bardziej pasowałyby do innych grup). Narysujmy teraz wykresy sylwetki:

```
silhouette_plot(X_train, y4, 4)
```



Widzimy, że w grupie 3 i 2 spora część obserwacji ma wartości mniejsze od średniej. Dla grup 3, 2, 1 część obserwacji ma ujemne wartości miary sylwetki. Grupa 3 i 2 wyglądają na niezbyt dobrze dopasowaną większość elementów ma wartości miary mniejsze od średniej dla całego zbioru. Lepiej sprawa wygląda w przypadku grupy 1 i 0, choć też nie jest perfekcyjnie.

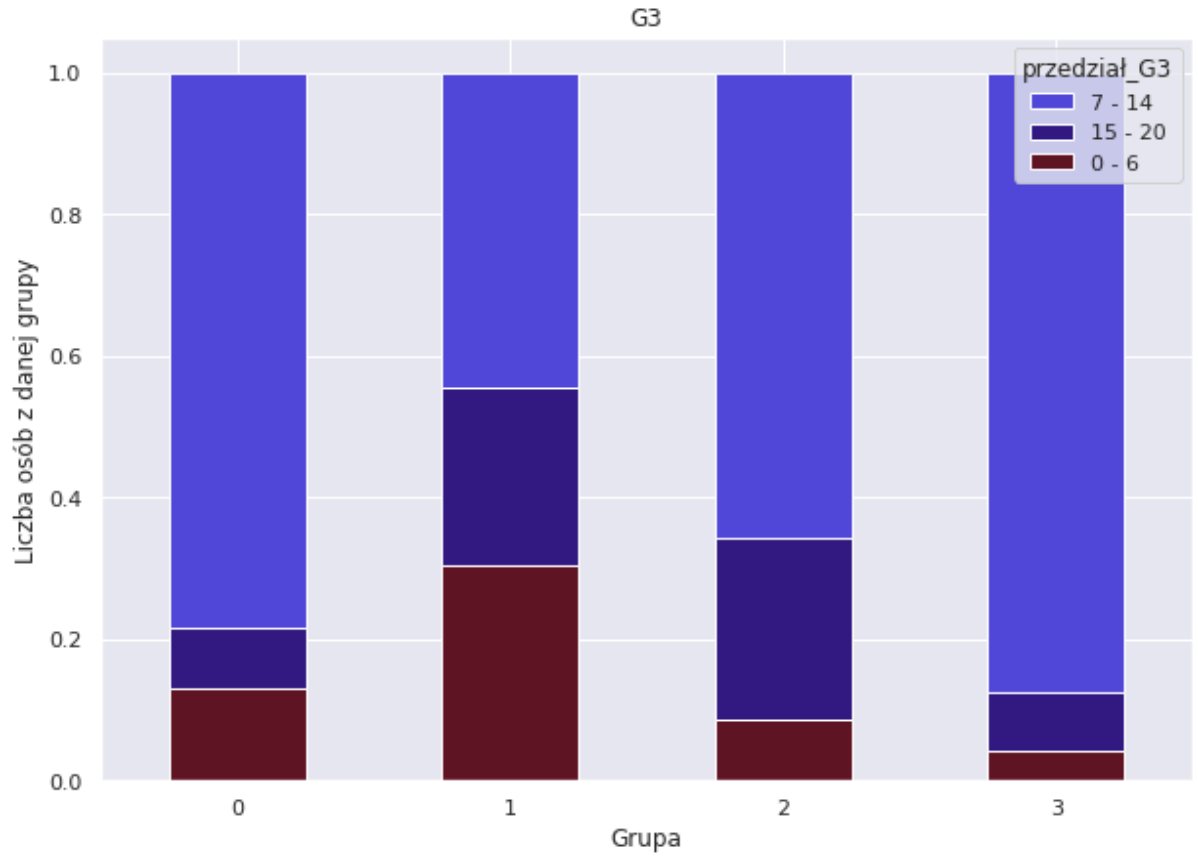
Pogrupujmy teraz naszym modelem zbiór testowy:

```
y4_test = km4.predict(X_test)
pd.Series(y4_test).value_counts().sort_index()
```

```
0    23
1    36
2    35
3    24
dtype: int64
```

Sprawdzimy teraz czy podział uczniów na grupy ma jakiś związek z oceną G3. W tym celu ponownie skorzystamy z przedziału ocen, aby lepiej móc wywnioskować coś z wykresu:

```
tabela1 = pd.crosstab(dane_testowe['grupa'],
dane_testowe['przedział_G3'], normalize = 'index')
tabela1.plot(kind='bar', stacked = True,
title='G3', xlabel='Grupa',
ylabel='Liczba osób z danej grupy', rot=0, figsize = (10,7), legend =
'reverse', color = ["#611423","#361b82","#5248db"])
```



Możemy zauważyć pewne zależności jeśli chodzi o podział na grupy a ocenę końcową, choć nie ma też ich dużo, podział uczniów na grupy nie ma dużego związku z oceną G3. Więcej ocen z przedziału 15-20 wpada do Grupy 1 niz do Grupy 3 i 0 (czyli uczniowie regularnie chodzący do szkoły mają lepsze oceny niż ci, którym zdarza się opuszczać zajęcia). Ale do tej samej grupy wpada też najwięcej słabych ocen, a ma to związek z tym że w naszym zbiorze mamy głównie studentów z

małą ilością nieobecności. Dużo najlepszych ocen wpada również do Grupy 2 która charakteryzuje się tym że studenci pobierają dodatkowe płatne zajęcia z przedmiotu. Najmniej słabych ocen trafia do grupy 3 czyli do studentów którzy nie chodzą raczej regularnie na zajęcia (co jest zaskakujące, ale pamiętajmy że nasz zbiór składa się głównie z regularnie chodzących studentów), częściej do tej grupy trafiają przeciętne oceny.