Projekt 1 – Prezentacja końcowa

Przemysław Olender, Dominik Pawlak

```
1 grades_df = pd.read_csv('school_grades_dataset.csv')
2
3 print(grades_df.shape)
4 grades_df.head()
```

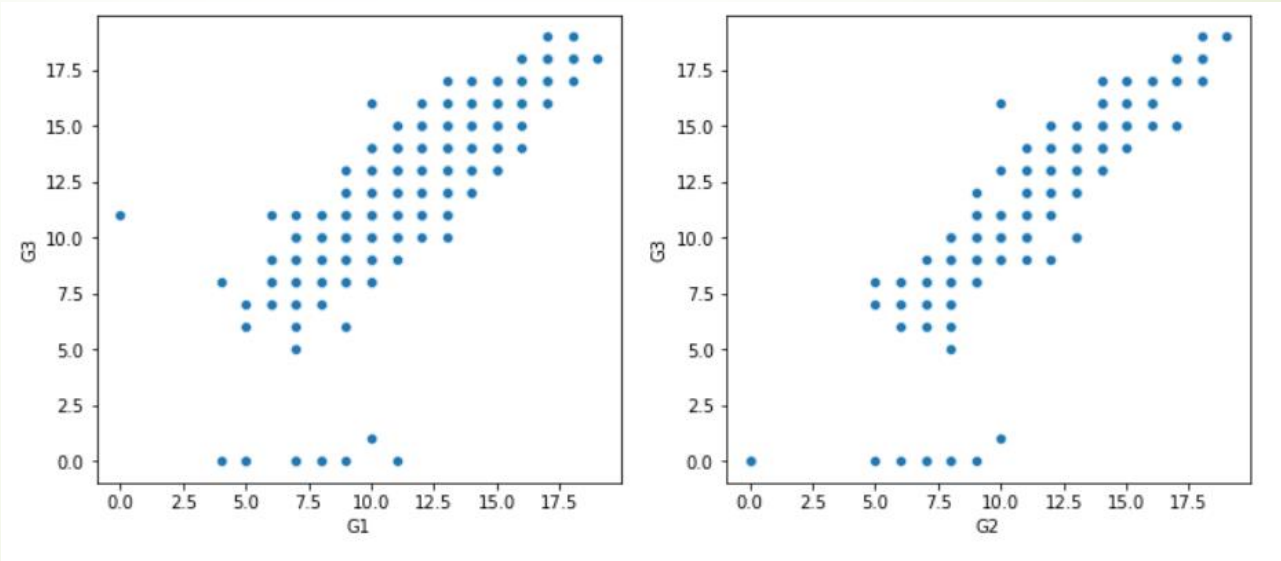
```
(649, 33)
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	4	0	11	11
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	2	9	11	11
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	6	12	13	12
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	0	14	14	14
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	0	11	13	13

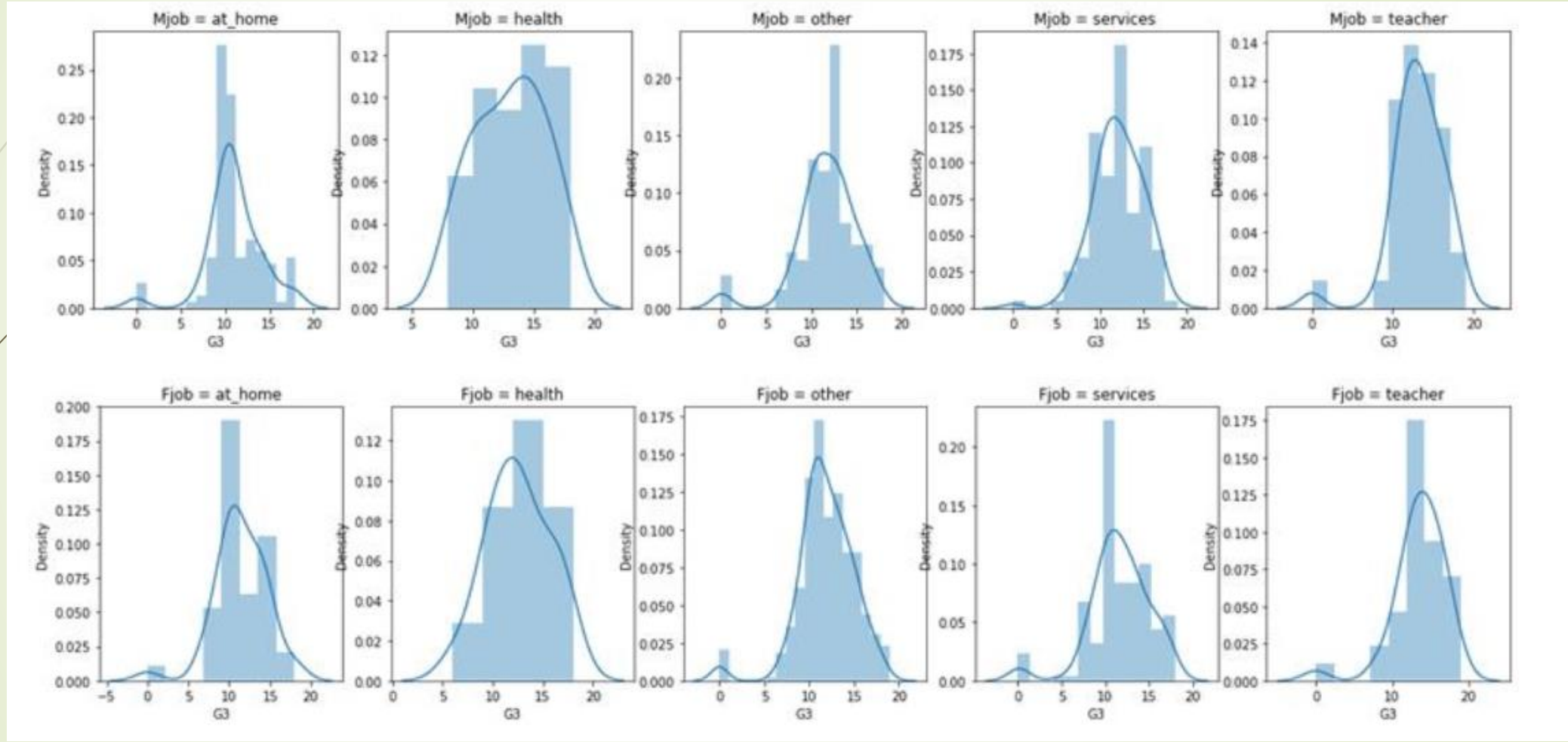
5 rows × 33 columns

Pierwsze zależności

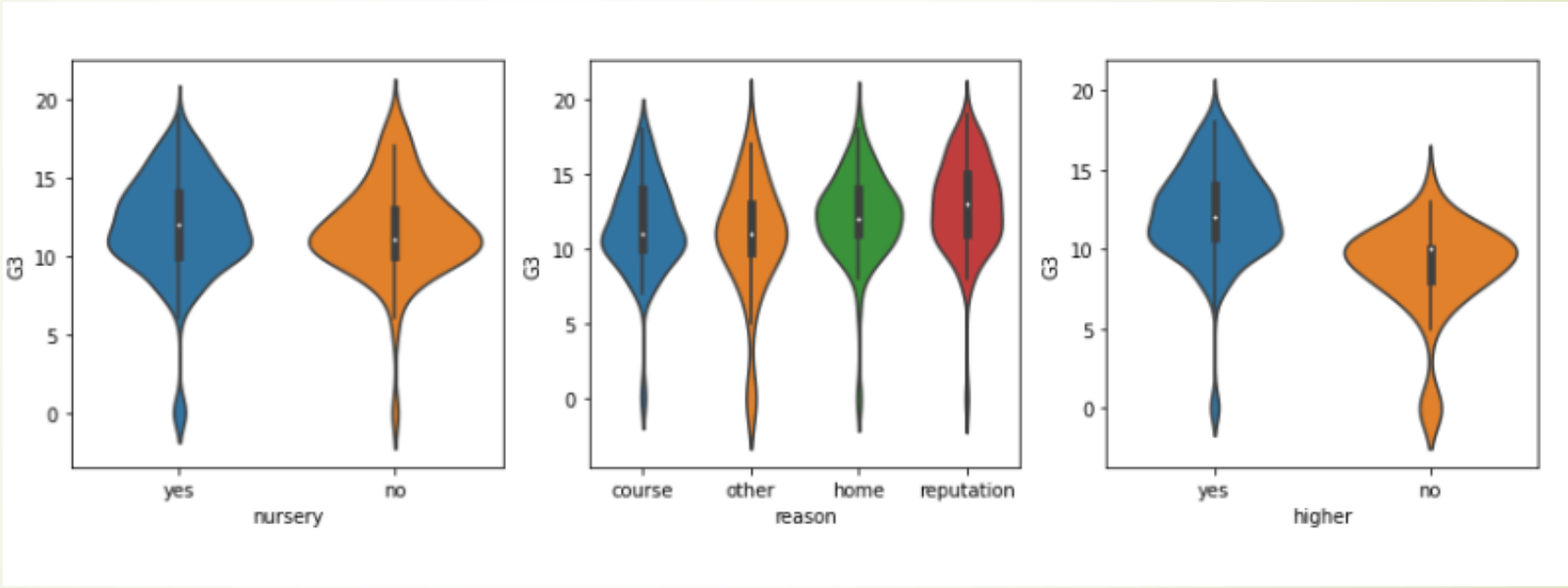
- Niemal liniowa korelacja między G1, a G2



Ważne cechy


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

Ważne cechy


G3
20
15
10
5
0
yes
no
nursery
G3
20
15
10
5
0
course
other
home
reputation
reason
G3
20
15
10
5
0
yes
no
higher

Usuwanie danych

- W zbiorze nie ma braków, nie trzeba więc wykonywać imputacji.
- 15 uczniów w zbiorze danych otrzymało 0 punktów z egzaminu końcowego.

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
163	GP	M	18	U	LE3	T	1	1	other	other	...	2	3	5	2	5	4	0	11	9	0
440	MS	M	16	U	GT3	T	1	1	at_home	services	...	5	4	5	4	5	3	0	7	0	0
519	MS	M	16	R	GT3	T	2	1	other	services	...	5	2	1	1	1	2	0	8	7	0
563	MS	M	17	U	GT3	T	2	2	other	other	...	1	2	1	2	3	5	0	7	0	0
567	MS	M	18	R	GT3	T	3	2	services	other	...	2	3	1	2	2	5	0	4	0	0

5 rows × 33 columns

Encoding

- W ramce znajduje się bardzo dużo kolumn kategorycznych, część z nich to odpowiedzi tak lub nie, w innych możemy znaleźć podział na kilka kategorii.
- Aby pozbyć się zmiennych kategorycznych użyliśmy encodingu, binarne odpowiedzi przekształciliśmy na 0 (no) i 1 (yes).
- Inne kategorię przykształciliśmy mapując średnią liczbę punktów zdobytą na egzaminie przez osoby do niej należące.

Grupowanie wyników

Aby nie przewidywać dokładnej liczby punktów jaką otrzymał uczeń podzieliliśmy wyniki na przedziały:

wynik	grupa	oznczenie w ramce	liczność
0-9	niezaliczony	1	85
10-11	słaby	2	201
12-13	średni	3	154
14-15	dobry	4	112
16-20	bardzo dobry	5	82

Pierwsze modele

- Stworzyliśmy pierwsze modele, zaczęliśmy modelowanie dokładnej wartości wyniku egzaminu przy uzyciu wszystkich cech:

```
xgboost accuarcy: 0.36649214659685864
Logistic Regression accuracy: 0.3141361256544502
RandomForestClassifier accuracy: 0.450261780104712
DecisionTreeClassifier accuracy: 0.4554973821989528
```

- Spróbowaliśmy też nie uzywając G1 I G2:

```
xgboost accuarcy: 0.17801047120418848
Logistic Regression accuracy: 0.16753926701570682
RandomForestClassifier accuracy: 0.17277486910994763
DecisionTreeClassifier accuracy: 0.18848167539267016
```

- Wyniki były bardzo słabe

Regresja liniowa

- Jako, że kolumny G1, G2 I G3 są liniowo zależne spróbowaliśmy użyć regresji liniowej do przewidywania wyniku. Wytrenowany model ma niskie RMSE I bardzo wysoki score.

```
RMSE: 0.9450989370465289
R-squared: 0.8596527332651904
```

- Jeśli zaokrąglimy wyniki do liczb całkowitych, to jakość modelu spada, nadal jednak wygląda lepiej niż przewidywanie wyniku za pomocą klasyfikacji.

```
Odesetek dobrze predykowanych zaokrąglonych wyników:      0.4816753926701571
```

Ponowne modelowanie

- Po podziale zmiennej celu na kubełki, ponownie użyliśmy przedstawionych wcześniej modeli. Wyniki były dużo lepsze.

```
xgboost accuarcy: 0.3193717277486911
Logistic Regression accuracy: 0.28272251308900526
RandomForestClassifier accuracy: 0.31413612565445026
DecisionTreeClassifier accuracy: 0.32460732984293195
```

- Również wybraliśmy najważniejsze kolumny:

```
xgboost accuarcy: 0.2879581151832461
Logistic Regression accuracy: 0.2931937172774869
RandomForestClassifier accuracy: 0.28272251308900526
DecisionTreeClassifier accuracy: 0.27225130890052357
```

Dalsze modelowanie

- Jeszcze raz powtórzyliśmy trenowanie modeli, tym razem sprawdziliśmy też jak zachowują się dla 7 najważniejszych cech (już z G1 I G2) przy użyciu RFE.
- Zdecydowaliśmy się rozwinąć analogiczne dwa modele: regresji logistycznej i lasu losowego. W tym celu używaliśmy narzędzi GridSearch i RFE.

```
Logistic regression accuracy: 0.6544502617801047
Logistic regression accuracy: 0.6910994764397905. (po zastosowaniu RFE)
Decision Tree accuracy: 0.6178010471204188
Decision Tree accuracy: 0.6387434554973822. (po zastosowaniu RFE)
Random Forest accuracy: 0.7277486910994765
Random Forest accuracy: 0.6910994764397905. (po zastosowaniu RFE)
XGBoost accuracy: 0.6649214659685864
XGBoost accuracy: 0.6492146596858639. (po zastosowaniu RFE)
```

Regresja logistyczna

- W pierwszej kolejności użyliśmy narzędzia **Grid Search**, które pozwoliło nam znaleźć najlepsze parametry dla naszego modelu. Zaprogramowaliśmy w nim parametry: **C**, **class_weight**, **fit_intercept**, **l1_ratio**, **solver**. Najlepszy model uzyskano dla podanych poniżej parametrów.

```
Best: 0.693066 using {'C': 0.4, 'class_weight': None, 'fit_intercept': True, 'l1_ratio': 0.0, 'solver': 'newton-cg'}
```

- Dla uzyskanego modelu stworzyliśmy następnie pętle znajdującą optymalną liczbę zmiennych użytych w narzędziu RFE.

```
Wynik dla regresji logistycznej: 0.7486910994764397. (po zastosowaniu RFE dla 11 zmiennych)
```

- Otrzymany model oceniliśmy również innymi miarami

```
F1-score: 0.7449279510229163
Precision: 0.7513956018937493
Recall: 0.7486910994764397
```

Las losowy

- Znów zaczęliśmy od użycia GridSearch, aby znaleźć najlepsze parametry modelu. Szukaliśmy optymalnych wartości dla parametrów criterion, class_weight oraz max_depth.

```
Best: 0.747120 using {'class_weight': 'balanced_subsample', 'criterion': 'gini', 'max_depth': 5}
```

- Następnie użyliśmy RFE aby sprawdzić dla których cech model jest najbardziej optymalny.
- Inne miary dla ostatecznego modelu to:

```
F1-score: 0.7272386623852753
Precision: 0.7345616330948538
Recall: 0.7329842931937173
```