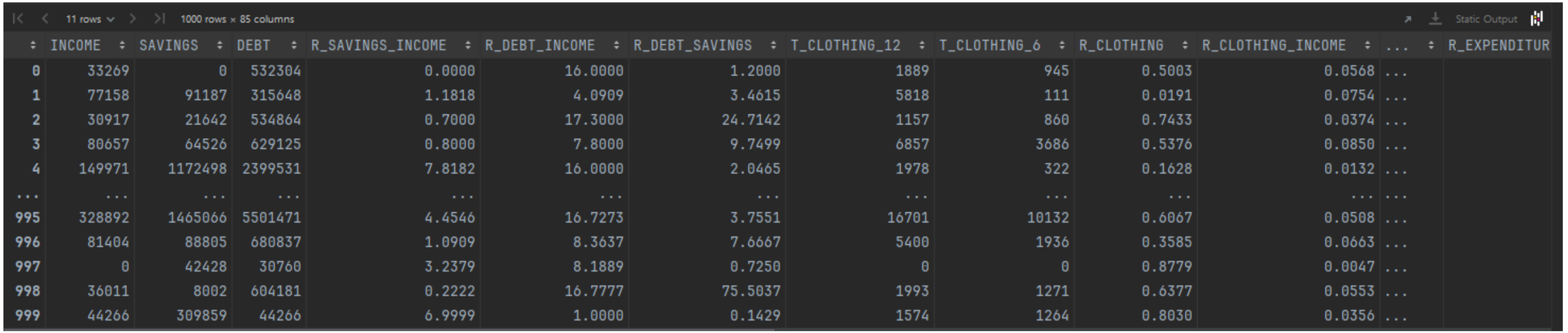
KOMU POWINNIŚMY UDZIELIĆ KREDYTU?

Projekt nr 1 z przedmiotu „Wstęp do uczenia maszynowego”

Anna Ostrowska, Dominika Gimzicka, Norbert Frydrysiak

Dane



11 rows | 1000 rows × 85 columns | Static Output

	INCOME	SAVINGS	DEBT	R_SAVINGS_INCOME	R_DEBT_INCOME	R_DEBT_SAVINGS	T_CLOTHING_12	T_CLOTHING_6	R_CLOTHING	R_CLOTHING_INCOME	...	R_EXPENDITUR
0	33269	0	532304	0.0000	16.0000	1.2000	1889	945	0.5003	0.0568	...	
1	77158	91187	315648	1.1818	4.0909	3.4615	5818	111	0.0191	0.0754	...	
2	30917	21642	534864	0.7000	17.3000	24.7142	1157	860	0.7433	0.0374	...	
3	80657	64526	629125	0.8000	7.8000	9.7499	6857	3686	0.5376	0.0850	...	
4	149971	1172498	2399531	7.8182	16.0000	2.0465	1978	322	0.1628	0.0132	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	328892	1465066	5501471	4.4546	16.7273	3.7551	16701	10132	0.6067	0.0508	...	
996	81404	88805	680837	1.0909	8.3637	7.6667	5400	1936	0.3585	0.0663	...	
997	0	42428	30760	3.2379	8.1889	0.7250	0	0	0.8779	0.0047	...	
998	36011	8002	604181	0.2222	16.7777	75.5037	1993	1271	0.6377	0.0553	...	
999	44266	309859	44266	6.9999	1.0000	0.1429	1574	1264	0.8030	0.0356	...	

- Głównie dane na temat na co ludzie wydawali pieniądze w ciągu ostatnich 12/6 miesięcy, przychody i oszczędności
- Ale też mniej oczywiste np. czy ktoś bawi się w hazard czy nie
- Dużo kolumn
- target = ‘DEFAULT’ (1 jeśli klient nie spłacił kredytu, 0 jeśli spłacił)

źródło danych: strona Kaggle, https://www.kaggle.com/datasets/conorsully1/credit-score

Cel Biznesowy

Części projektu

1. EDA

przeanalizowanie dostępnych danych

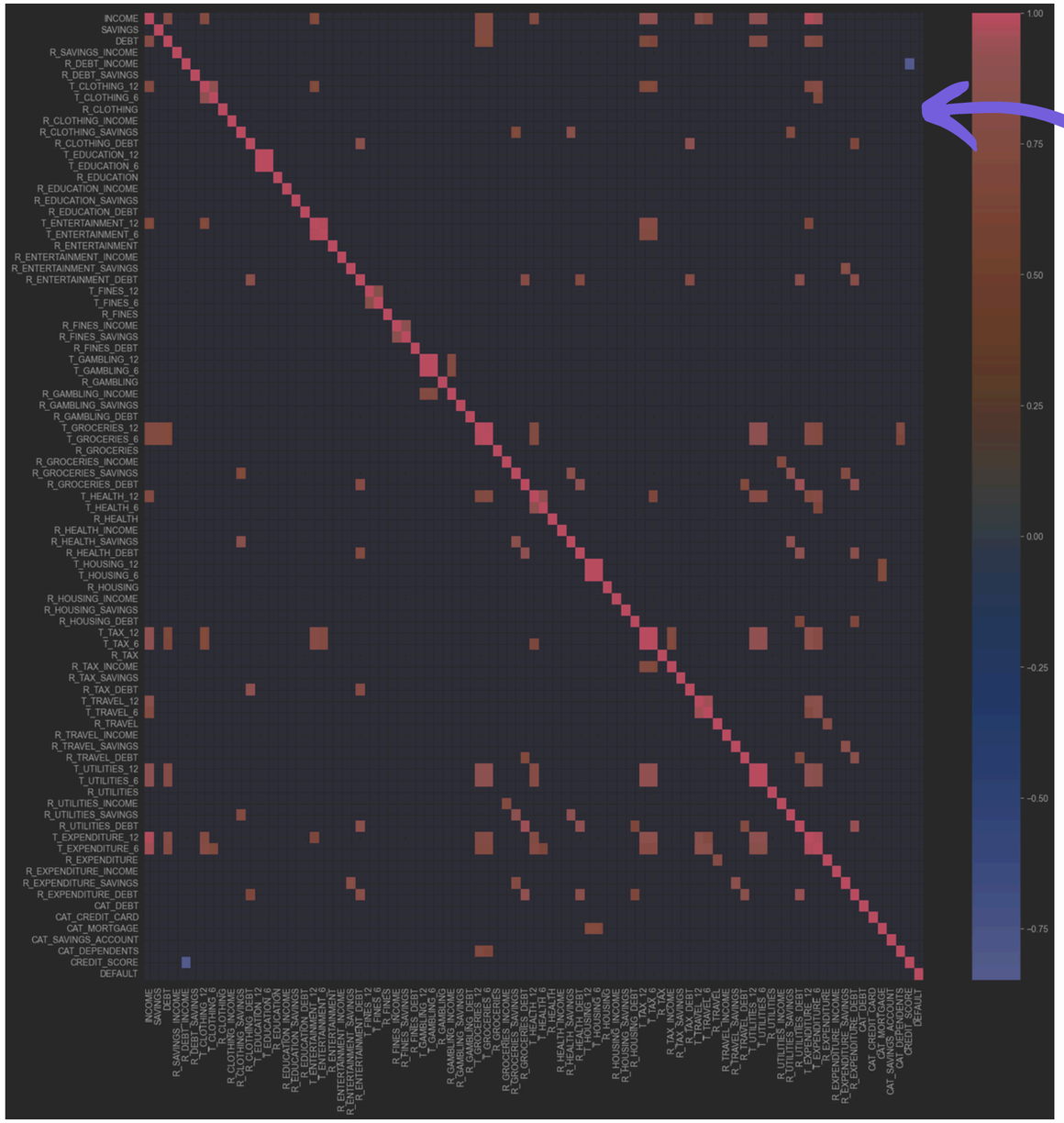
2. feature engineering

preprocessing danych, wstępne modelowanie

3. final

dodanie bardziej zaawansowanych modeli, kroswalidacja, strojenie hiperparametrów, metody wyjaśnialności

EDA



1. Podział zbioru na dane walidaycjne i testowe
2. Brak wartosci nullowych w zbiorze
3. 1 zmienna kategoryczna (CAT_GAMBLING), zmapowana używając ordinal encodingu
4. Dużo kolumn, część należy usunąć (correlation matrix)
5. Brak zrównoważenia targetu (utrudnione zadanie)
6. Należy usunąć outliery i zrobić transformacje zmiennych na rozkład normalny

```
DEFAULT
0    450
1    178
Name: count, dtype: int64
```

Inżynieria cech

Zmienne kategoryczne

‘CAT_GAMBLING’

↓

Ordinal Encoding

(No = 0 , Low = 1, High=2)

Kolumny z duża korelacją

"T_CLOTHING_12","T_ENTERTAINMENT_12",
"T_GROCERIES_12", "T_GROCERIES_6", "T_HEALTH_12", "T_TAX_12",
"T_TAX_6", "T_TRAVEL_12", "T_TRAVEL_6","T_UTILITIES_12",
"T_UTILITIES_6", "T_EXPENDITURE_12", "T_EXPENDITURE_6"

↓

Usuwamy

Brak NULLów

Outliery

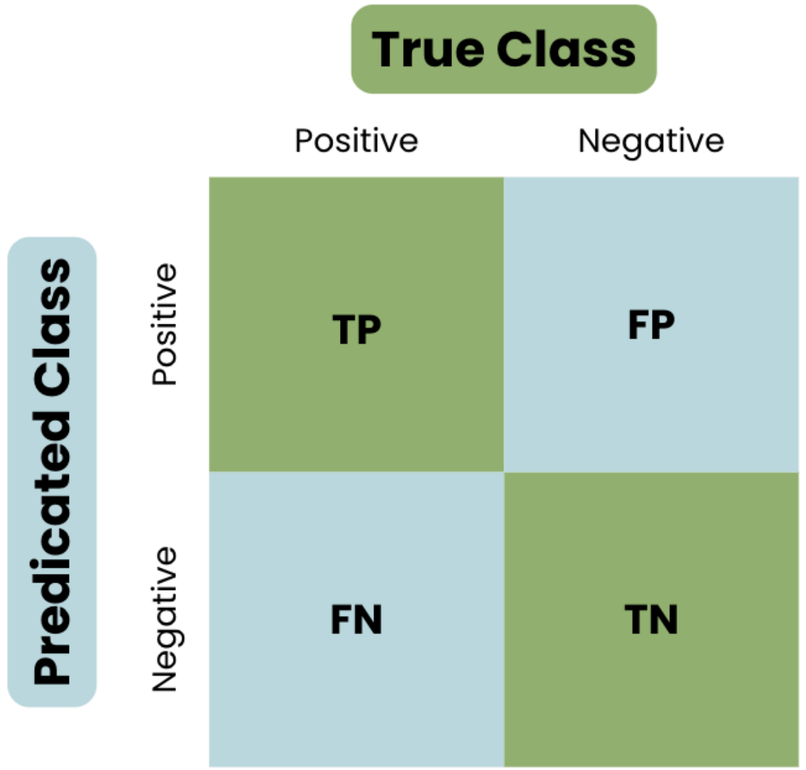
Automatyczna detekcja (PyOD)
(zakładamy, że jest ich 4%)

Transformacja

Box-Cox + standaryzacja

Na jakie metryki patrzymy?

$$PRECISION = \frac{TP}{TP + FP} = \frac{TP}{\text{TOTAL PREDICTED POSITIVE}} \quad RECALL = \frac{TP}{TP + FN}$$



Wstępne modele

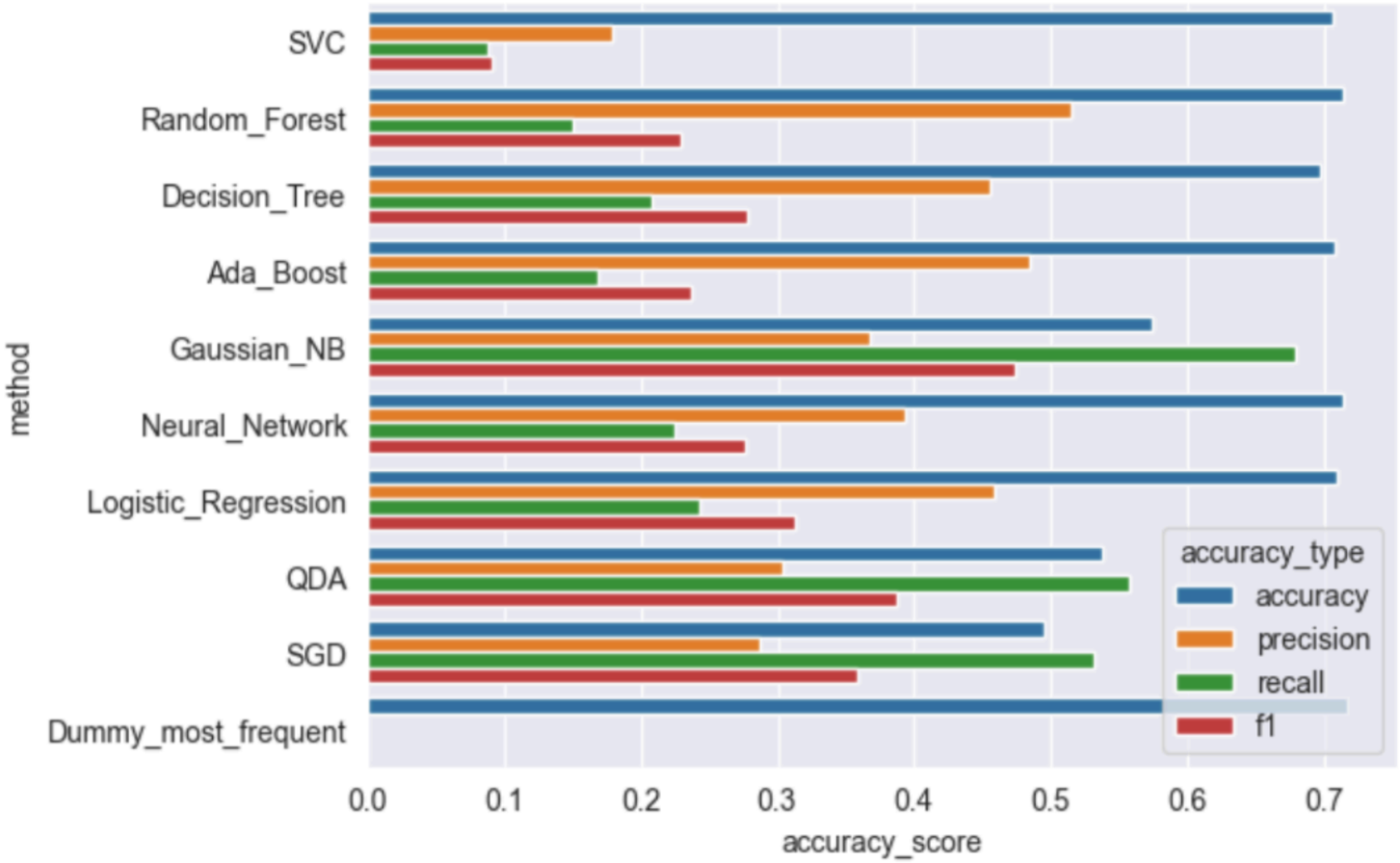


Figure 10: Wyniki różnych metryk z drugiego modelowania dla różnych modeli dla zbioru potraktowanego transformacją Box Cox oraz z automatycznym usuwaniem outliernerów

Bardziej zaawansowane modele i techniki

- Bagging
- Hard Voting
- Soft Voting
- Soft Voting z różnymi wagami
- Stacking
- RandomForestClassifier
- AdaBoostClassifier
- GradientBoosting
- XGBClassifier
- TPOT
- AutoML

	precision	recall	f1-score	support
0	0.80	0.28	0.42	99
1	0.31	0.82	0.45	39
accuracy			0.43	138
macro avg	0.56	0.55	0.43	138
weighted avg	0.66	0.43	0.43	138

Figure 16: classification_report dla BaggingClassifier z parametrami: estimator=model6, n_estimators=10, random_state=0

	precision	recall	f1-score	support
0	0.83	0.30	0.44	99
1	0.32	0.85	0.47	39
accuracy			0.46	138
macro avg	0.58	0.57	0.46	138
weighted avg	0.69	0.46	0.45	138

Figure 13: classification_report dla Soft Voting z wagami [1,1,1,5,1,1,25,15]

Strojenie hiperparametrów

- GridSearch
- RandomizedSearch
- BayesSearch

- DecisionTreeClassifier
- GradientBoostingClassifier
- RandomForestClassifier
- GaussianNB
- QuadraticDiscriminantAnalysis

```
Best F1-score: 0.485993 using {'var_smoothing': 1e-05}
              precision    recall  f1-score   support

           0       0.80      0.37      0.51        99
           1       0.33      0.77      0.46        39

    accuracy                           0.49       138
   macro avg       0.57      0.57      0.48       138
weighted avg       0.67      0.49      0.50       138
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
Best F1-score: 0.466625 using {'priors': [0.3, 0.7], 'reg_param': 0.3, 'store_covariance': True, 'tol': 0.001}
              precision    recall  f1-score   support

           0       0.82      0.40      0.54        99
           1       0.34      0.77      0.47        39

    accuracy                           0.51       138
   macro avg       0.58      0.59      0.50       138
weighted avg       0.68      0.51      0.52       138
```

- GaussianNB

```
              precision    recall  f1-score   support

           0       0.81      0.21      0.34        99
           1       0.30      0.87      0.45        39

    accuracy                           0.40       138
   macro avg       0.56      0.54      0.39       138
weighted avg       0.67      0.40      0.37       138

Best F1-score: 0.906260 using {'var_smoothing': 1e-12}
```

Crossvalidation

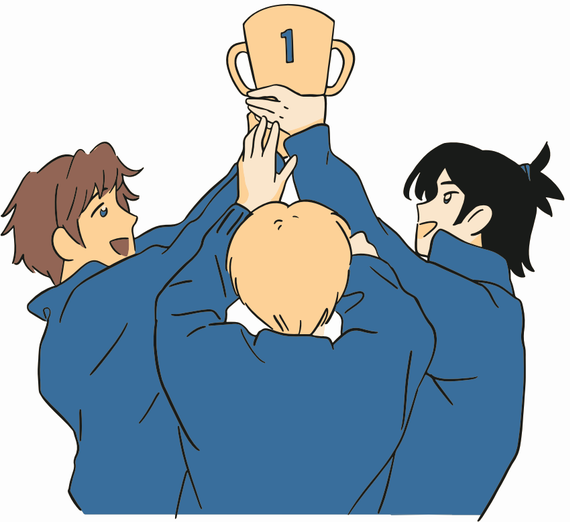
model	accuracy	accuracy_std	precision	precision_std	recall	recall_std	f1	f1_std
Missing 0 / Count 8 / 8 Unique values	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8	Missing 0 / Count 8
LogisticRegression	0.697410	0.036087	0.438797	0.120167	0.285873	0.090187	0.345532	0.102926
XGBClassifier	0.727683	0.036433	0.629841	0.296587	0.123492	0.058368	0.205325	0.096108
GradientBoostingClassif…	0.694311	0.042004	0.403015	0.173062	0.179683	0.086070	0.247700	0.113528
GaussianNB(1e-12)	0.407644	0.035635	0.313954	0.010005	0.916190	0.063067	0.467024	0.012299
GaussianNB(1e-05)	0.500000	0.048000	0.339397	0.024949	0.798095	0.068055	0.475298	0.030544
SoftVotingClassifier(we…	0.447479	0.036680	0.323191	0.016509	0.865714	0.074734	0.470042	0.024493
BaggingClassifier	0.444305	0.037775	0.321038	0.019391	0.860000	0.069656	0.467160	0.028247
QDA	0.522273	0.027367	0.342200	0.014017	0.741746	0.072915	0.467296	0.022470

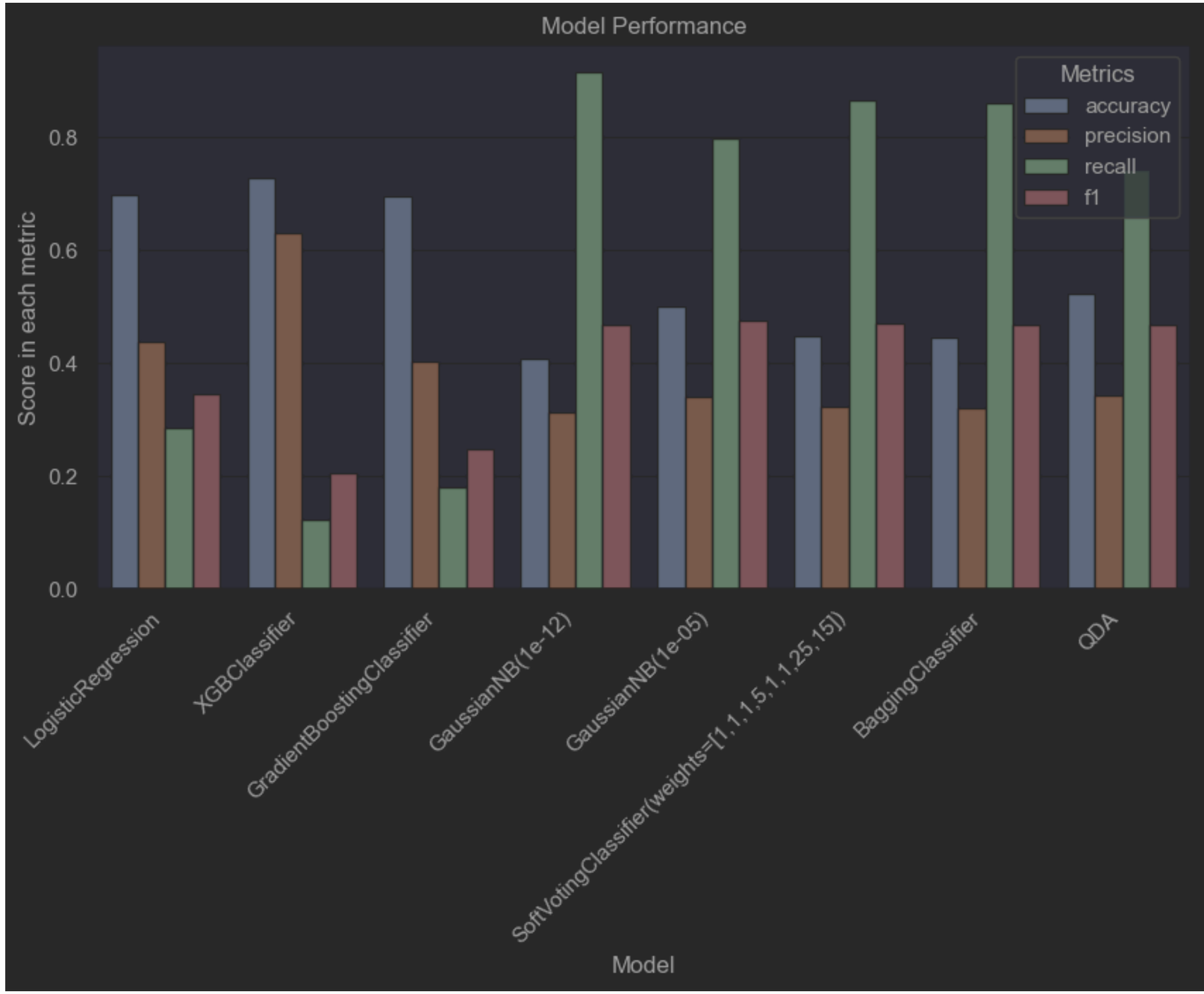
Następujące modele są bardzo niestabilne:

- LogisticRegression
- XGBClassifier
- GradientBoostClassifier

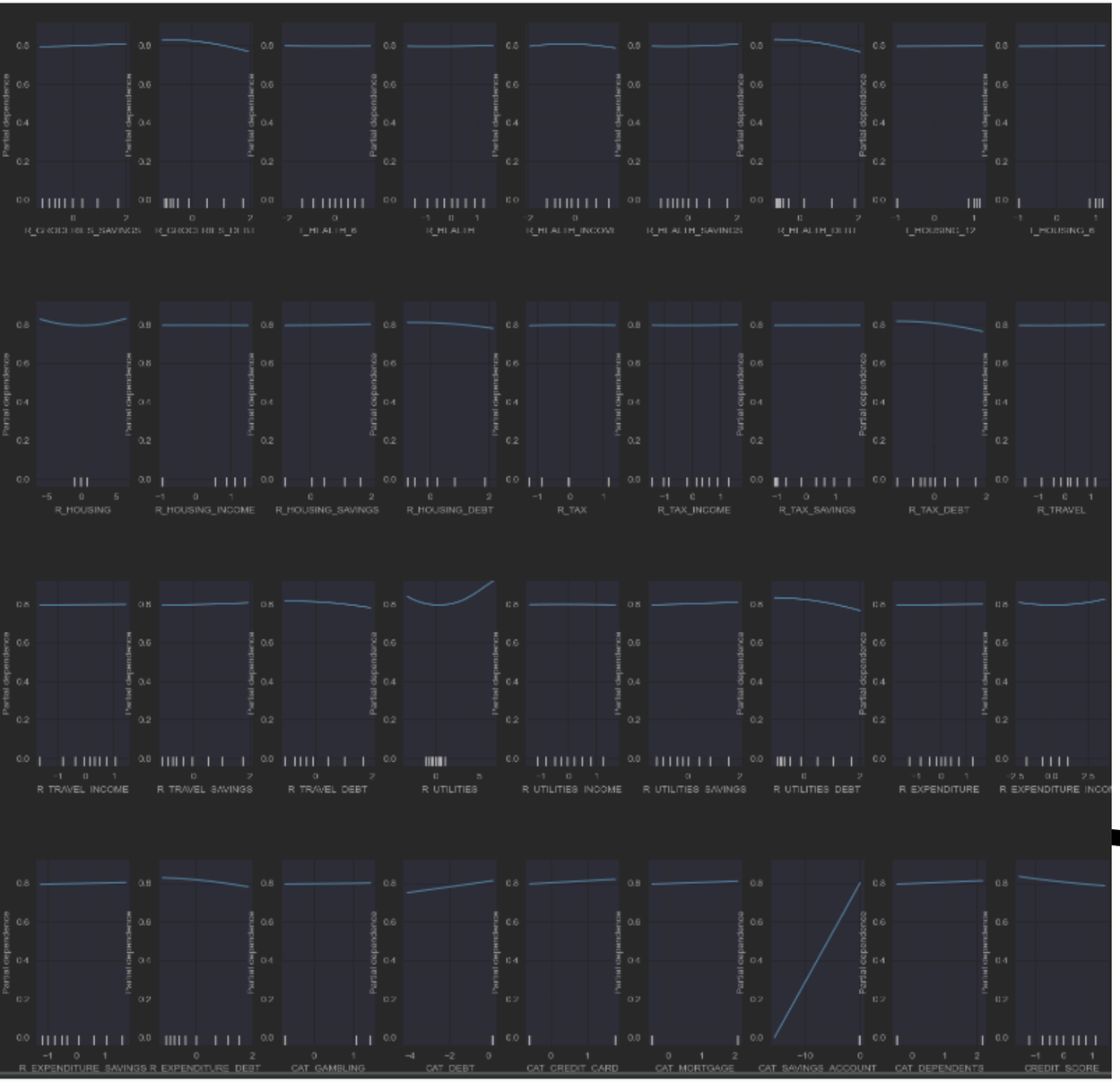
Wybrany model

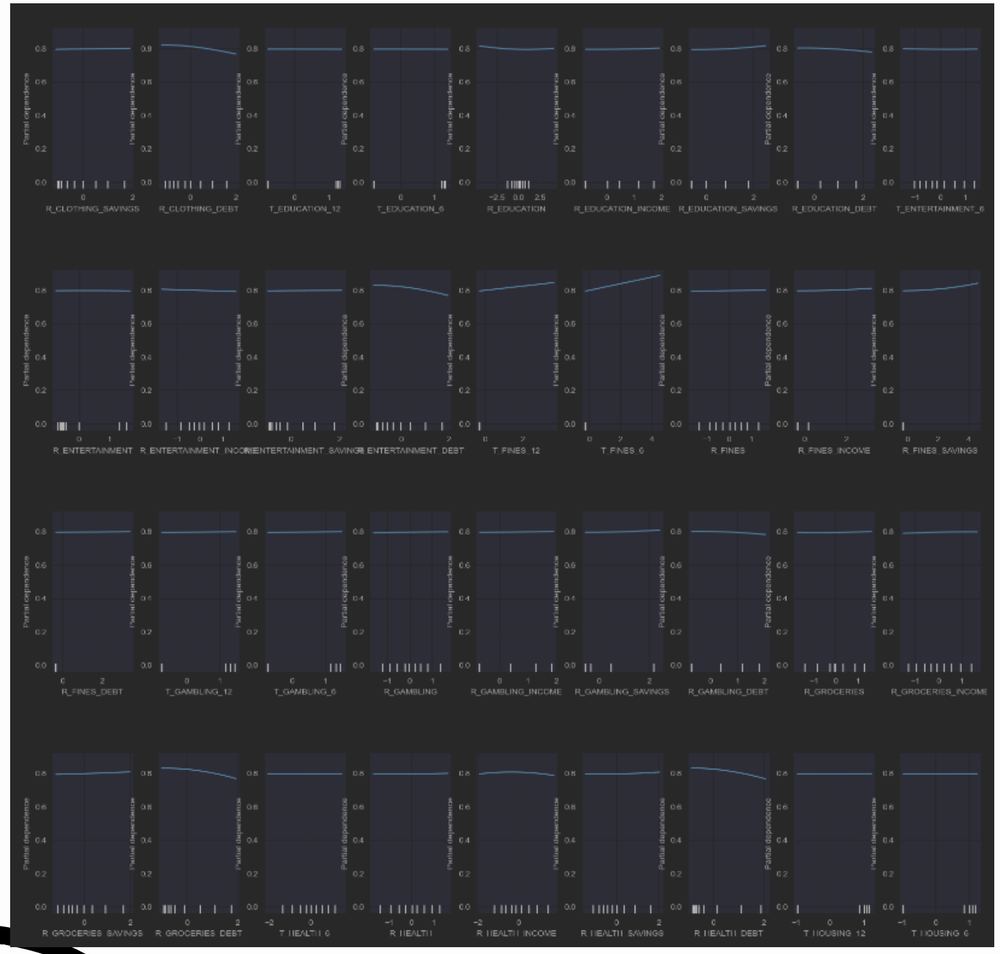
GaussianNB(1e-12)





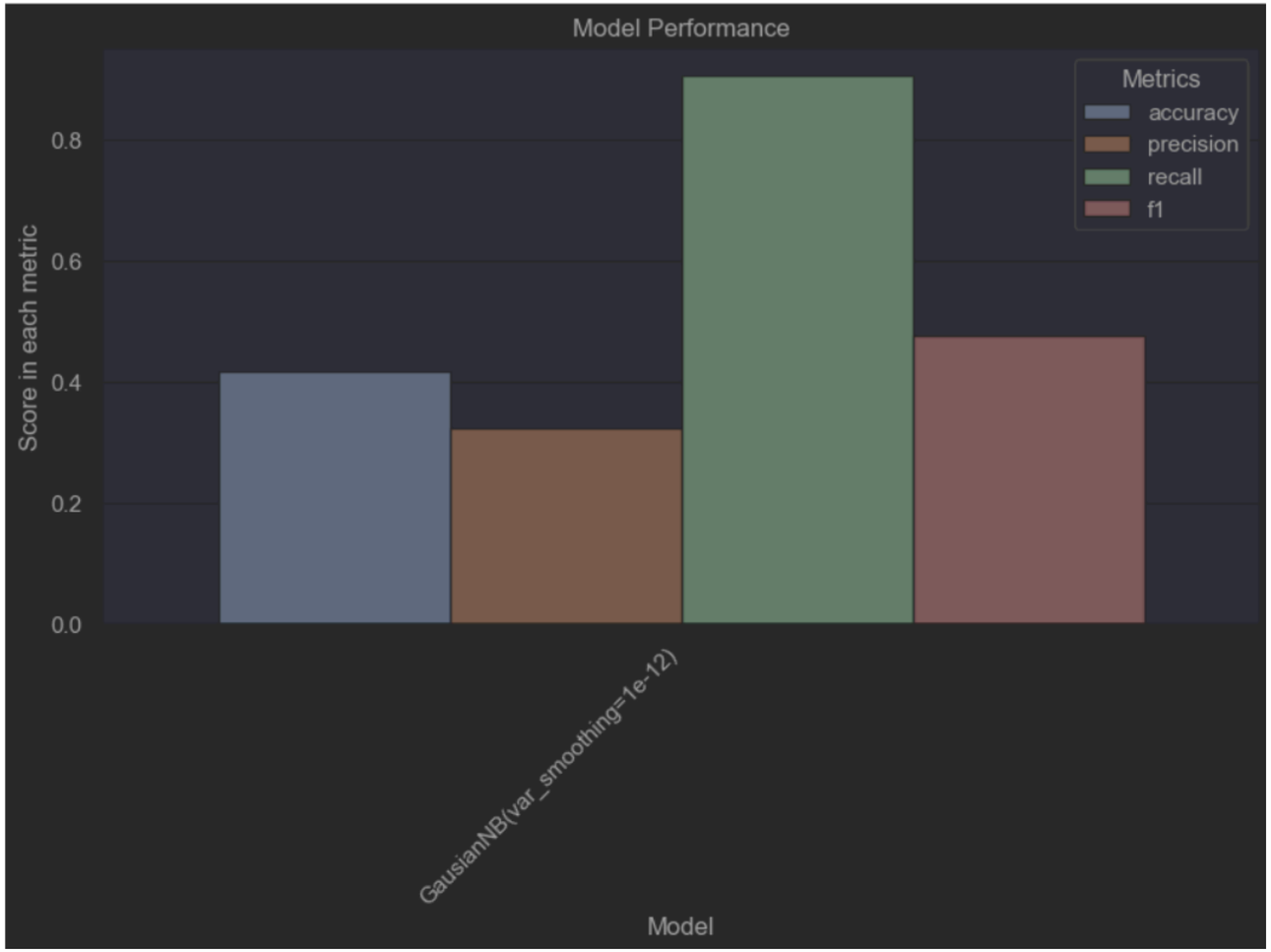
Metody wyjaśnialności - Partial Dependance Plots





-większość zmiennych ma marginalny wpływ na wynik modelu
-jedyna zmienna o dużym wpływie to ‘CAT_SAVINGS_ACCOUNT’

Czy osiągnięto Cel?



- sukces - znaleźliśmy model, który maksymalizuje recall i to na niezłym poziomie w miarę stabilnie
- wyłapujemy znaczną większość osób, które nie spłacają kredytu
- teraz możemy sprzedać ten klasyfikator jako bardzo bezpieczny klasyfikator dla finansów banku, szczególnie dla banku, który ma problemy finansowe i nie chce ryzykować.

Co byśmy zrobili bez walidatorów?

Nie wiemy!! ❤️

Sprawdźcie na zbiorze testowym tylko finalny wybrany model.	Tak
Fajny pomysł z wykresem porównującym wyniki indywidualne, zwiększa czytelność :)	Tak
Dla SVC, można użyć funkcji classification_report, a wyniki będą ładnie widoczne razem, ogólnie dla każdego modelu warto wywołać tę funkcję, to ładne i czytelne podsumowanie. Szczególnie w przypadku SVC, łatwo się pogubić w tym, co jest czym	Tak

Dziękujemy za uwagę i pozdrawiamy Natalię oraz Karolinę jako fajnych walidatorów

link do projektu - https://github.com/fantasy2fry/credit-score-classification-ml