Przewidywanie, czy ktoś spłaci kredyt z wykorzystaniem różnych modeli uczenia maszynowego

Anna Ostrowska, Dominika Gimzicka, Norbert Frydrysiak

April 2024

1 Wstęp

Na zajęcia „Wstęp do uczenia maszynowego" na 2 roku kierunku Inżynieria i Analiza Danych na wydziale Matematyki i Nauk Informacyjnych Politechniki Warszawskiej przygotowaliśmy projekt, nad którym pracowaliśmy od początku marca do końca kwietnia 2024 roku.
Dostaliśmy ramkę danych ze strony Kaggle (link do ramki danych w rozdziale „Źródła"), zawierającą kolumny z różnymi informacjami pochodzącymi z transakcji finansowych i bieżącej sytuacji finansowej 1000 klientów banku, wraz z informacją, czy dany klient ma niespłacony kredyt.
Głównym celem projektu było wykorzystanie tego zbioru danych do szacowania ryzyka kredytowego i przewidywania potencjalnych przypadków niewypłacalności z wykorzystaniem różnych modeli uczenia maszynowego przedstawionych na zajęciach laboratoryjnych i zdecydowanie, który modeli z jakimi hiperparametrami najdokładniej przewiduje, czy klient spłaci kredyt.
Projekt podzielony był na 3 części:

1. Eksploracyjna analiza danych (EDA)
2. Inżynieria cech i wstępne modelowanie
3. Final (bardziej zaawansowane modele, kroswalidacja, strojenie hiperparametrów modeli i metody wyjaśnialności)

Cały kod napisany przez nas w celu zrealizowania projektu umieszczony został na naszym repozytorium na Githubie: https://github.com/fantasy2fry/credit-score-classification-ml.

1.1 Walidacja

Nasza grupą walidatorów, która po każdej części analizowała nasze postępy i dzieliła się przemyśleniami, co możemy poprawić i jak ulepszyć nasz projekt, składała się z 2 osób z naszego kierunku: Natalia Choszczyk i Karolina Dunal.

1.2 Plan biznesowy

Jako cel postawiliśmy sobie jak najdokładniejsze przewidywanie, czy klient spłaci kredyt, na podstawie danych zawartych w ramce danych ze strony Kaggle. Mogłoby to się przydać w bankach przy decydowaniu, czy warto udzielić komuś kredytu, czy też nie.
Czysto biznesowo chcemy mieć klasyfikator, który bardzo szybko i trafnie zdecyduje na podstawie wprowadzonych danych czy powinniśmy dać kredyt temu klientowi. Jeśli uda się takie trafne narzędzie stworzyć moglibyśmy znacząco zautomatyzować i ułatwić taki proces wielu bankom.
Problem może być taki, że w naszych danych możemy nie mieć wystarczającego dostępu do potrzebnych informacji - potencjalne niespłacenie kredytu może wynikać z informacji niedającej się wyrazić liczbami czy tego typu danymi (a nawet z przyczyn powstałych dopiero po udzieleniu kredytu, których nie da się przewidzieć, takich jak problemy prywatne, choroby, zmiany na rynku pracy czy inne zdarzenia nagłe, często niezwiązane z pozycją finansową klienta). Jakość produktu jaki powinniśmy dostać, aby był on przydatny, powinna być lepsza od obecnych praktyk, dzięki którymi banki wybierają kredytobiorców. Zadanie jest arcytrudne i możliwe, że prawie niemożliwe jest stworzenie takiego narzędzia.

1.3 Dane

Wykorzystywana przez nas ramka danych zawiera 84 kolumny, w tym numeryczne: CUSTOMER_ID (numer identyfikacyjny klienta), INCOME (całkowity dochód w ciągu ostatnich 12 miesięcy), SAVINGS (całkowite oszczędności w ciągu ostatnich 12 miesięcy), DEBT (całkowita wartość istenijącego długu), wydatki w ciągu 6 i 12 miesięcy w różnych kategoriach (artykuły spożywcze, odzież, edukacja, mieszkanie, zdrowie, podróże itp.), CREDIT_SCORE (zdolność kredytowa) oraz kategoryczne: informacja o kategorii hazardu (brak, niski, wysoki), czy klient jest zadłużony, czy klient ma kartę kredytową, kredyt hipoteczny, konto oszczędnościowe i czy ma jakiekolwiek osoby na utrzymaniu. Jako kolumnę „target" przyjęliśmy kolumnę "DEFAULT": binarną zmienną wskazująca, czy klient nie spłacił kredytu (1), czy też wywiązał się ze zobowiązań (0).

1.4 Brak zrównoważenia klas "DEFAULT"

Z racji, że będziemy przewidywać czy ktoś spłaci kredy czy nie. Należy rozważyć czy mamy zrównoważony zbiór danych na zawierający dwie klasy:

- Ludzie, którzy spłacą kredyt,
- Ludzie, którzy nie spłacą kredytu.

Z racji dużego braku zrównoważenia, nasze zadanie jest trochę utrudnione. Modele mogą trochę źle uczyć się, bo mają stosunkowo znacznie mniej przykładów mniejszej z klas.

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

Figure 1: Rozkład klas w części zbioru dla modelarzy - zbiór treningowy oraz zbiór walidacyjny (bez zbioru testowego).

2 EDA

W pierwszej części naszego projektu sprawdzaliśmy, z jakimi danymi mamy do czynienia oraz podzieliliśmy dane na testowe i walidacyjne w przedstawiony poniżej sposób (pamiętając o ustaweniu losowego ziarna):

1. Podzielenie całego zbioru danych na dwie części: dla modelarzy i dla walidatorów (70% i 30%).
2. Podzielenie pierwszej części na dwie części: do treningu i do walidacji (70% i 30%).
3. Podzielenie części walidacyjnej z pierwszej części na dwie części: do walidacji i do testowania (66% i 34%).
4. Zapisanie wszystkich części w osobnych plikach.

Po przeanalizowaniu ramki danych wyciągnęliśmy następujące wnioski:

- w ramce danych nie ma brakujących wartości (NULLów)
- kolumnę kategoryczną CAT_GAMBLING należy zmapować, ponieważ ma 3 kategorie. Resztę kolumn kategorycznych można zostawić jak wartości binarne
- ramka danych ma dużo kolumn, należy zdecydować, które trzeba zostawić, a które można usunać z ramki
- nasz target (kolumna "DEFAULT") nie ma równo rozłożonych wartości - jest prawie 3 razy więcej zer, niż jedynek

2.1 Mapowanie

Zmapowaliśmy kolumnę CAT_GAMBLING za pomocą słownika, przypisując kategoriom następujące wartości: No: 0, Low: 1, High: 2.

2.2 Usuwanie kolumn

Na poczatku zdecydowaliśmy się usunąć kolumnę CUSTOMER_ID, ponieważ dla każdego wiersza jest ona inna i nie przedstawia żadnych informacji, które mogłyby pomóc w osiągnięciu naszego celu.
Następnie przedstawiliśmy resztę kolumn w formie macierzy korelacji w celu sprawdzenia, których kolumn jeszcze moglibysmy się pozbyć. Ze względu na duża liczbę kolumn w ramce danych zdecydowaliśmy się jaednak patrzeć na macierz korelacji, która pokazuje tylko korelacje, których wartości bezwględne są $>$0,7.

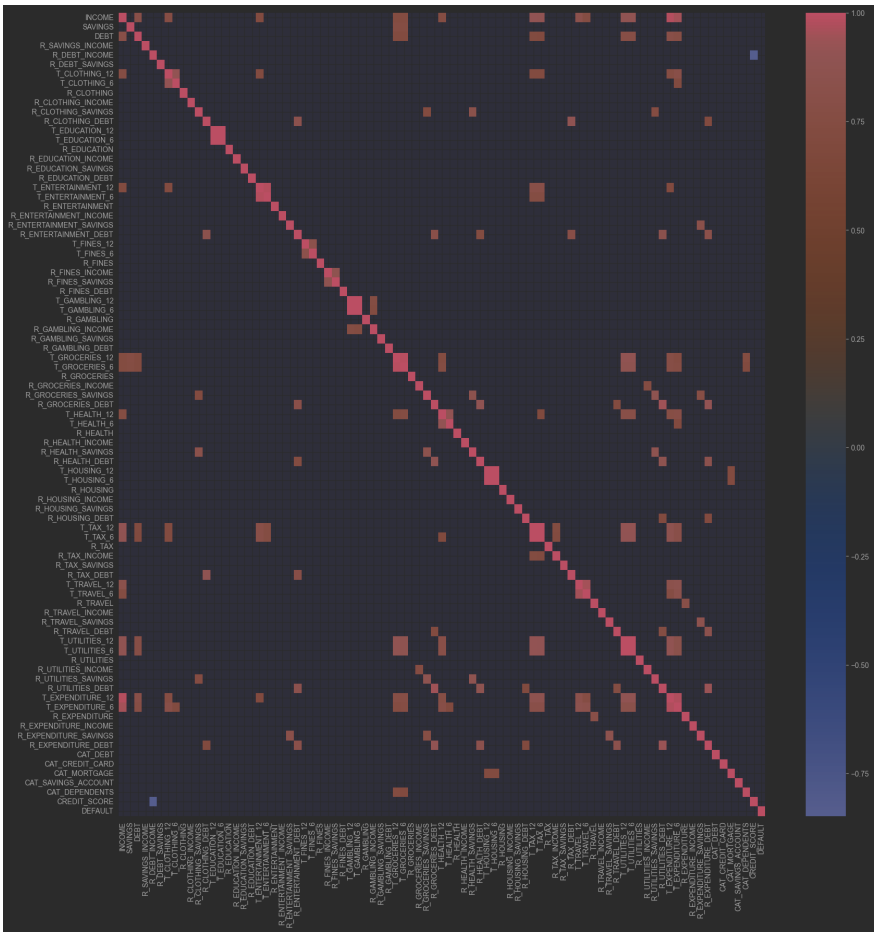


Figure 2: Macierz korelacji kolumn w używanej przez nas ramce danych.

Po analizie naszej ramki danych i powyższej macierzy korelacji zdecydowaliśmy się, że naszymi głownymi kandydatami do usunięcia bedą kolumny o nazwach:

- T_TAX_6
- T_TAX_12
- T_UTILITIES_6
- T_UTILITIES_12
- T_EXPENDITURE_6
- T_EXPENDITURE_12
- T_GROCERIES_6
- T_GROCERIES_12

Rozważymy również kolumny takie jak:

- T_HEALTH_6
- T_HEALTH_12
- T_TRAVEL_6
- T_TRAVEL_12

Powyższe kolumny sa mocno skorelowane z innymi kolumnami, więc nie ma potrzeby przechowywania informacji o wszystkich z nich.
Zdecydowaliśmy się również usunąć kolumnę CAT_DEBT (zmienna binarna informująca, czy ktoś ma kredyt), ponieważ ta informacja jest podana w kolumnie DEBT (wartość kredytu: 0 jeśli CAT_DEBT = 0).
Zauważyliśmy również, że kolumna CREDIT_SCORE jest mocno skorelowana z naszym targetem, więc na pewno nie będziemy jej usuwać - pomoże nam ona w wyznaczeniu wyników (przewidywaniu, czy ktoś spłaci kredyt).

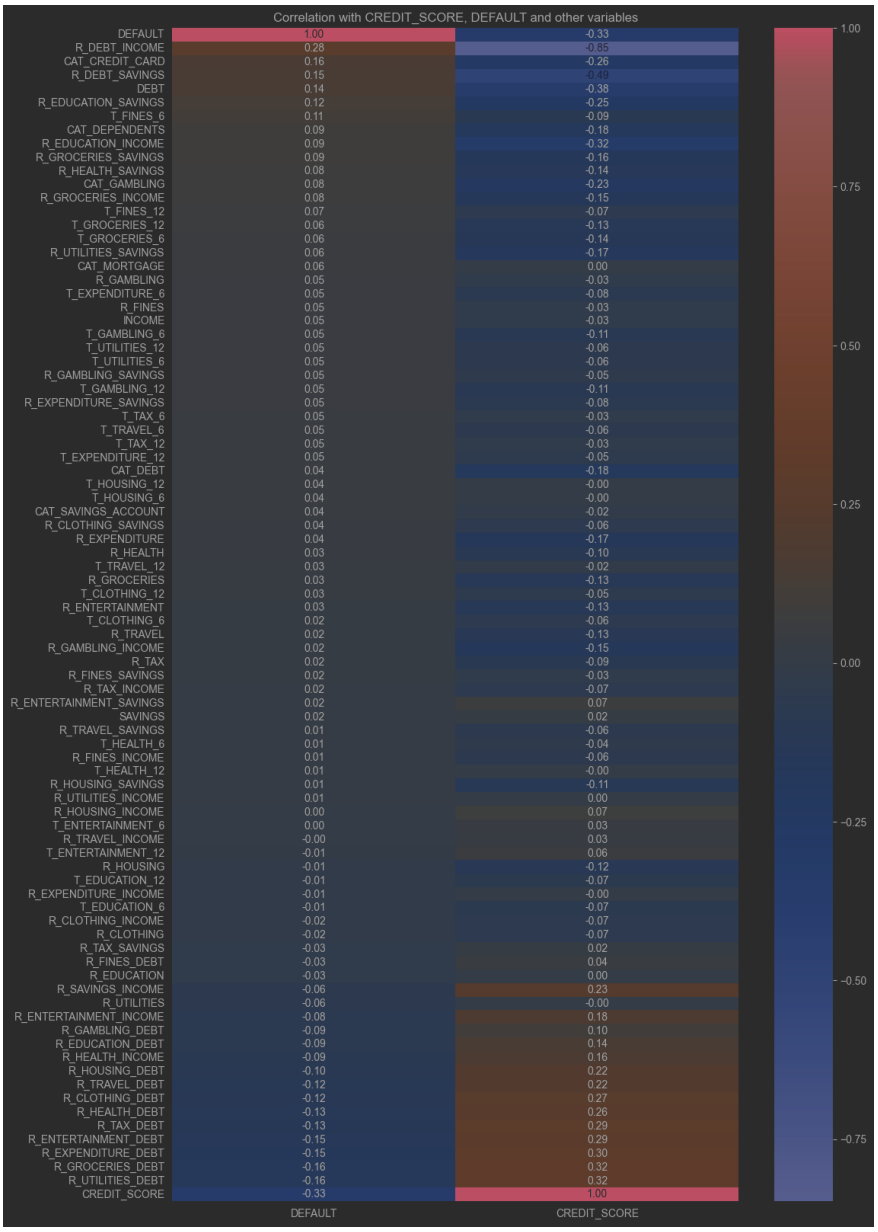


Figure 3: Korelacja między CREDIT_SCORE, DEFAULT i innymi zmiennymi.

2.3 Rozkład wartości w kolumnach

Sprawdziliśmy rozkład wszystkich wartości, oprócz tych gdzie wartości są z przedziału tylko 0 lub 1, w celu zdecydowania, czy i jaką transformację należy

wykonać w następnej części projektu.

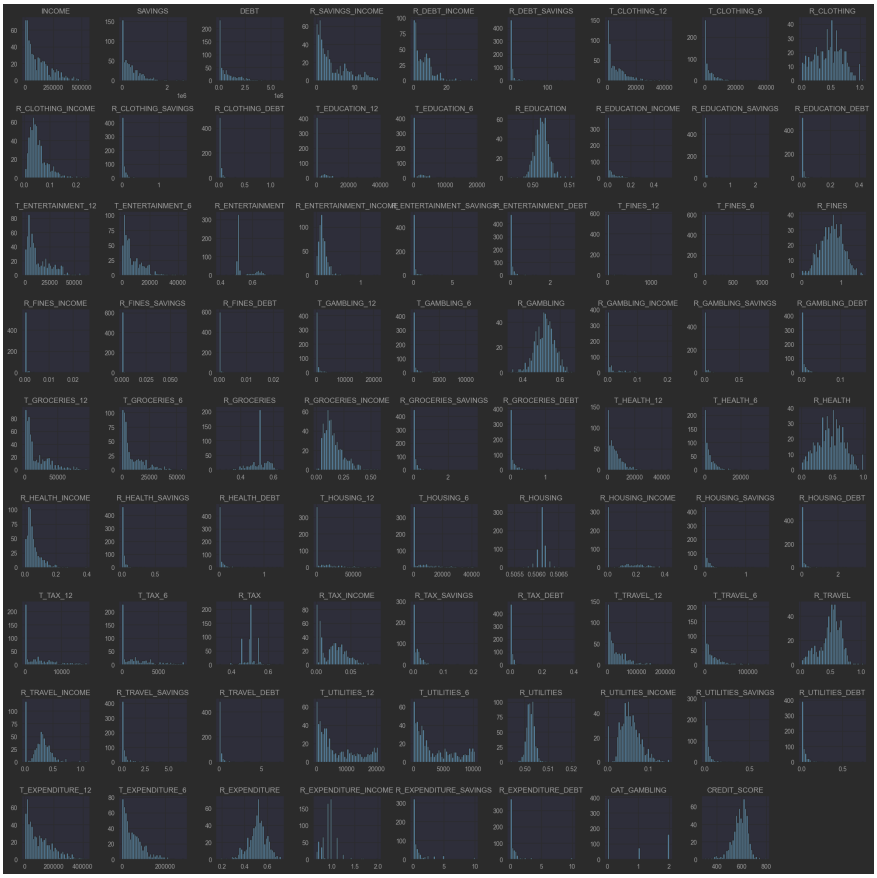


Figure 4: Rozkład zmiennych w ramce danych.

Można zauważyć, że niektóre kolumny mają rozkłady podobne do rozkładu normalnego,ale nie wszystkie. Będzie więc potrzebne przeprowadzenie transformacji zmiennych.

2.4 Wnioski i spostrzeżenia

Dodatkowe spostrzeżenia:

- im wyższy "CAT_GAMBLING", tym wyższy "DEFAULT" (a więc większa szansa na niespłacenie długu).
- na początku może się wydawać, że INCOME i SAVINGS nie mają wpływu na to, czy klient spłaca kredyt - korelacje między tymi dwoma zmiennymi a DEFAULT sa prawie zerowe, jednak można sprawdzić, że jak

podzielimy ramkę na różne kategorie w zależności od wysokości kwoty wziętego kredytu, to wtedy w każdej kategorii SAVINGS i INCOME są wysoko skorelowane z DEFAULT. Z tego wynika, że po prostu osoby z wyższym przychodem i oszczędnościami biorą wyższe kredyty, co wiąże się prawdopodobnie z mniejszym prawdopodobieństwem spłacenia kredytu. Wniosek z tego taki, że czasami pomimo tego, że jakaś kolumna wydaje się mieć niską korelację z targetem, nie należy jej ignorować - w połączeniu z inną kolumną może mieć ona duże znaczenie.

Wnioski z 1. etapu projektu (co należy zrobić na następnych etapach):

- wiele nieoczekiwanych kolumn jest ważnych
- CREDIT_SCORE jest najważniejszą cechą
- musimy zdecydować, które kolumny usunąć (na razie usunęliśmy tylko CUSTOMER_ID i CAT_DEBT)
- jest tylko jedna zmienna kategoryczna, którą należy zmapować - "CAT_GAMBLING" (użyliśmy ordinal encoding dla tej zmiennej)
- są pewne wartości odstające (outliery), ale nie tak wiele (należy zdecydować czy, i w jaki sposób je usuniemy)
- należy przeprowadzić transformację zmiennych

2.5 Bonus - Niektóre modele zwracają, które kolumny są ważne

W międzyczasie poszliśmy na wykład i dowiedzieliśmy się, że niektóre modele zwracają ważność kolumn w ich predykcji, zastasowaliśmy tę wiedzę, co zweryfikowało naszą chęć usunięcia wszystkich kolumn zaczynających się od R_.

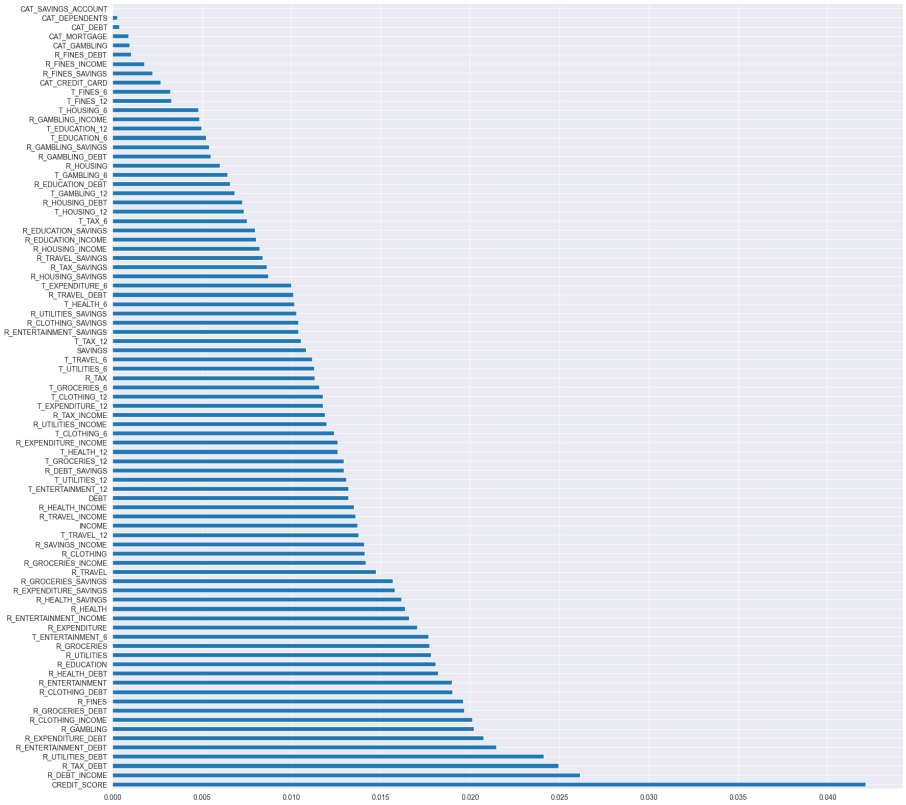


Figure 5: Wykres przedstawiający ważność poszczególnych feature według modelu RandomForestClassifier

2.6 Walidacja

Zgodnie z radami naszej grupy walidatorów zdecydowaliśmy się po tym etapie wprowadzić następujące zmiany w naszym projekcie:

- usunęliśmy zmienną kategoryczną CAT_GAMBLING w macierzy korelacji
- zmieniliśmy scatterplot na boxploty do sprawdzenia rozrzutu wartości kategorycznych (CAT_GAMBLING)
- zmienne binarne przeanalizowaliśmy za pomocą wykresów słupkowych

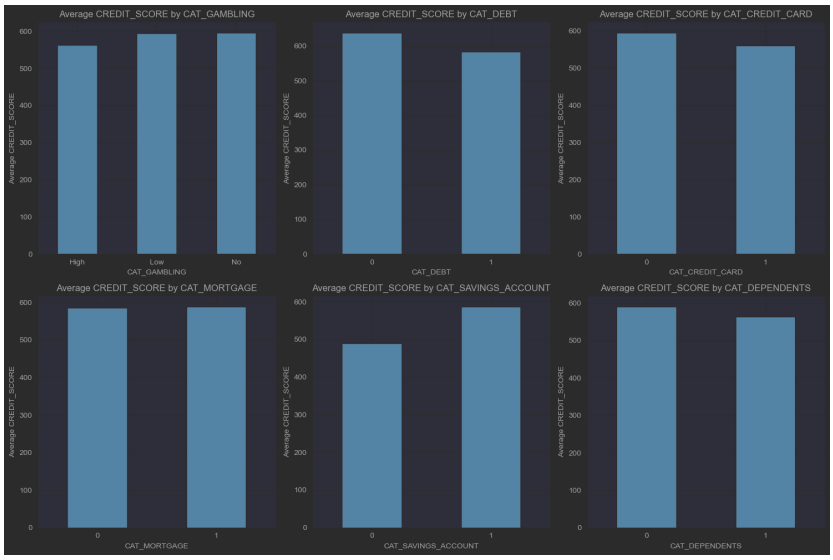


Figure 7: Wykresy rozkładów wartości binarnych.

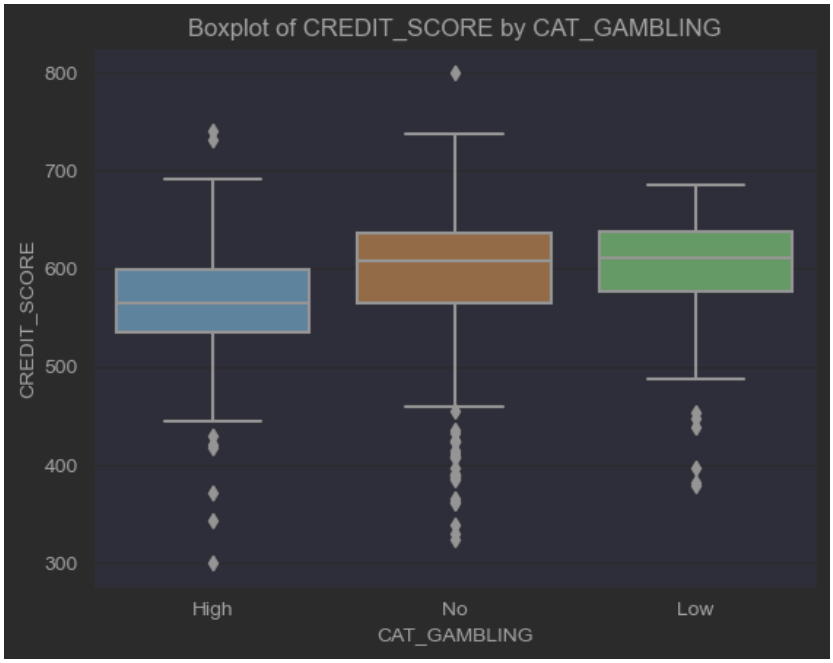


Figure 6: Rozkład wartości zmiennej CAT_GAMBLING.

3 Inżynieria cech i wstępne modelowanie

3.1 Inżynieria cech

W tym etapie zajęliśmy się przygotowaniem zbioru danych do modelowania.

3.1.1 Wnioski z EDA

Na początku zwróciliśmy uwagę na wnioski z EDA:

- nie ma brakujących wartości (NULLów)
- jest jedna zmienna kategoryczna 'CAT_GAMBLING' (użyliśmy ordinal encoding)
- jest 86 kolumn i usunęliśmy do tej pory tylko jedną - 'ID'
- dobrymi kandydatami do usunięcia są kolumny: "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" (zajmiemy się nimi później)

3.1.2 Outliery

Do detekcji outlierów przetestowaliśmy 2 strategie:

1. Wykrywanie ręczne

   Spróbowaliśmy użyć do tego najpierw wykresów typu boxplot, jednak w naszym zbiorze danych jest za dużo kolumn, aby były one czytelne.

   Później użyliśmy wykresów puktowych, dla wszystkich kolumn oprócz 'DEFAULT' (ponieważ jest naszym celem - tym, co będziemy chciec przewidzieć), zmiennych binarnych oraz 'CAT_GAMBLING'(ponieważ była początkowo zmienną kategoryczną). Z otrzymanych wykresów wypisaliśmy ręcznie widoczne outliery, otrzymjąc granice dla nich.

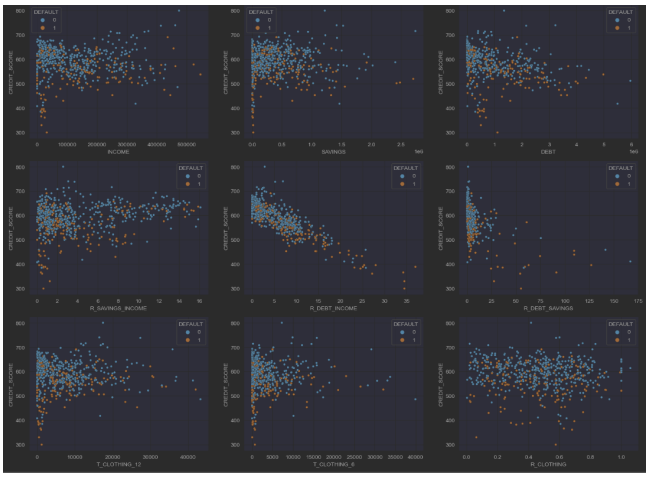


Figure 8: Przykładowe wykresy jakie uzyskaliśmy dla pierwszych 9 zmiennych z naszego zbioru danych

2. Wykrywanie automatyczne

   Drugim spsosobem jaki wypróbowaliśmy do usuwania outlierów jest PYOD, czyli automatyczne wykrywanie outlierów. Aby móc go użyć należy podać procent, jaki uważamy, że stanowią outliery. Po przetestowaniu rożnych wartości (nie możemy usunąć ani za dużo ani za mało danych), zdecydowaliśmy, że 4% daje nam najlepsze wyniki (wykrywa w ten sposób 20 wartości odstających i 470 do pozostawienia).

Na ten moment nie zdecydowaliśmy jeszcze, która strategia będzie dawała lepsze rezultaty - sprawdzimy to przy wstępnym modelowaniu.

3.1.3 Transformacja danych

Jako, że wśród naszych danych (zgodnie z histogramami przedstawionymi na Figure 6) mało zmiennych ma rozkłady normalne, ale za to dużo ma rozkłady podobne do normalnego oraz wszystkie nasz dane są dodatnie, uznaliśmy że ostatecznie użyjemy transformacji Boxa-Coxa wraz z Standard Scaler. Dzięki takiej transformacji uznaliśmy, że modele będą najefektywniej działać, jednak sprawdzimy naszą tezę jeszcze w trakcie wstępnego modelowania.

3.2 Wstępne modelowanie

Wstępne modelowanie bardzo pokazało nam przed jak bardzo trudnym zadaniem stoimy. Uznaliśmy, że jest każda obserwacja w zbiorze treningowym jest na wagę złota i nie powinniśmy za bardzo usuwać wielu obserwacji przy wykrewaniu outlinerów automatycznie. Uznaliśmy, że 4 % to outlinery. Zauważyliśmy, że nie powinniśmy prawie w ogóle polegać na metryce Accuracy. Uznaliśmy, że z punktu widzenia biznesowego powinniśmy patrzeć na metryki:

1. F1
2. Recall
3. Precision

Metryka F1 jest bardzo ważna, ponieważ nie chcemy zminimalizaować liczbę udzielonych kredytów, które miałyby nigdy nie zostać spłacone - w takim przypadku nasz bank straciłby dużo pieniędzy, a nawet zbankrutował.
Z drugiej strony chcemy udzielać jak największą liczbę kredytów, jeśli wiemy, że zostaną one spłacone. Odmawiając zbyt wielu klientom udzielenia kredytu, niewiele osób chciałoby korzystać z naszych usług i jednocześnie bank nie miałby na czym zarabiać.
Na poniższych wykresach przedstawiono wyniki kilku podstawowych modeli, sprawdzonych w ramach tego etapu projektu, przedstawione za pomocą różnych metryk:

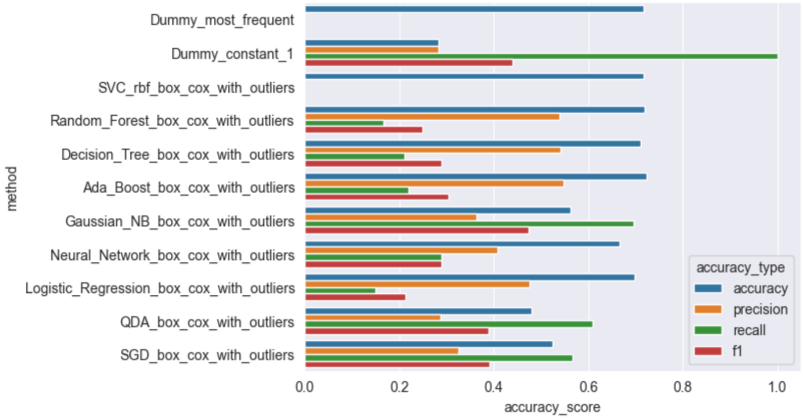


Figure 9: Wyniki różnych metryk z pierwszego modelowania dla różnych modeli dla zbioru potraktowanego transformacją Box Cox oraz bez usuwania outlierów.

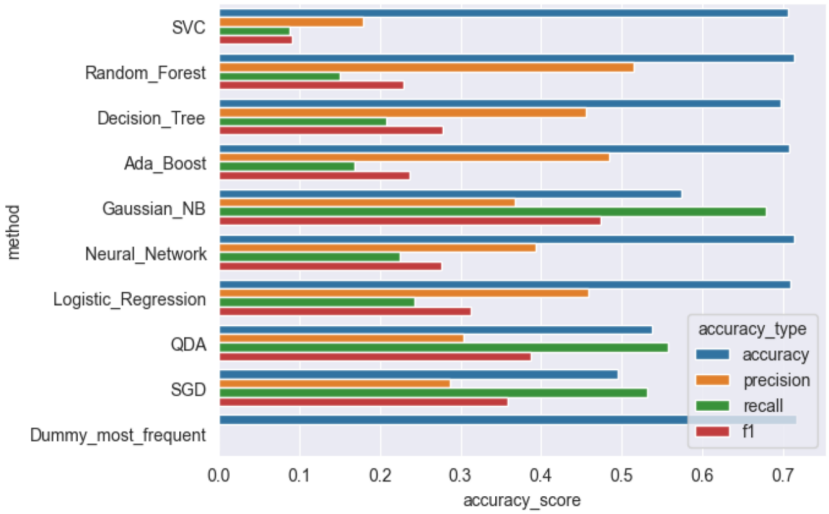


Figure 10: Wyniki różnych metryk z drugiego modelowania dla różnych modeli dla zbioru potraktowanego transformacją Box Cox oraz z automatycznym usuwaniem outlierów

Jak widać te modele nie są w stanie dokładnie przewidywać targetu, czyli czy klient spłaci kredyt. Widać też, że poleganie na wyłącznie jednej metryce jest bardzo mylące: na przykładzie Dummy_most_frequent czy Maszyny Wektorów Nośnych z kernelem "linear", które wyniki accuracy mają bardzo wysokie, ale nie mają sensu, gdyż w reszcie metryk dają wyniki równe lub bliskie 0 .
Na wykresach 10 oraz 9 widać, że naciekawszym dla nas modelem wydaje się Naiwny Klasyfikator Bayesowski. Szczególnie wysokie wyniki ma dla metryk recall i f1.

3.3 Wnioski

Sprawdziliśmy, które kolumny są mocno skorelowane i na tej podstawie wybraliśmy, które z nich będziemy usuwać i podobnie uznaliśmy, że będziemy automatycznie usuwać outlinery.
Niestety wyniki pierwszych modeli pokazały, że zadanie jest bardzo trudne, ale też intuicyjnie dla banków często też jest to trudne i w rzeczywistości dają one kredyty osobom, które mają ogromne prawdopodobieństwo, że spłacą kredyt i przez to wiele osób denerwuje się, że nie dostają kredytu. My próbujemy inaczej podejść do sprawy - chcemy często dawać kredyty, ale też nie przesadzić w drugą stronę. Z wstępnych modeli widać, że Naiwny Bayes jest prawdopodobnie celnym wyborem przy tym problemie. Jednak trzeba przeszukać różne parametry, ponieważ wyniki są dalej dalekie od zadowalających. Na następnym etapie musimy pokombinować z tworzeniem komitetów i doborem hiperparametrów.
Jeżeli chodzi o zagadnienia feature engineeringu, to mieliśmy tylko jedną zmi-

enną nienumeryczną i to było CAT_GAMBLING, na której wykonaliśmy ordinal encoding. Zmapowaliśmy 'No' jako 0, 'Low' jako 1 oraz 'High' jako 2 i jest to proste, ale naszym zdaniem najlepsze rozwiązanie tutaj.

3.4 Walidacja

Uwagi od walidatorów	**Czy wzięto pod uwagę?**
PyOD - czy dodano wyjaśnienie dlaczego przewidujesz, że 4% będzie odstępstwami? (zanieczyszczenie = 0.04, jeśli dobrze rozumiem, sami to ustawiliśmy na podstawie prognozy)	Tak
Może odstępstwa będą bardziej widoczne na wykresach pudełkowych? Niż na wykresach punktowych (nie wiem)	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
Fajny pomysł z wykresem porównującym wyniki indywidualne, zwiększa czytelność :)	Tak
Może spróbuj sprawdzić, czy pominięcie niektórych kolumn poprawi wyniki	Tak
Byłoby miło dodać kilka podsumowujących uwag, podsumowujących które opcje są najlepsze i dlaczego	Tak

Table 1: Podsumowanie uwag od grupy walidatorów

4 Final

W ostatniej części projektu wypróbowaliśmy bardziej zaawansowane modele, zrobiliśmy strojenie hiperparametrów i kroswalidację. Natępnie porównaliśmy wyniki, jakie otrzymaliśmy za pomocą classification_report i tworząc wykresy i zdecydowaliśmy, które modele w naszym wypadku są najbardziej sensowneDodalismy do nich metody wyjaśnialności i zdecydowaliśmy, który model będzie naszym ostatecznym, dającym najlepsze wyniki. Patrzyliśmy głównie na recall i f1, ponieważ, jak wspomniane w poprzednim rozdziale, są one dla nas najważniejsze.

4.1 Bardziej zaawansowane modele

W celu znalezienia najlepszego modelu wypróbowaliśmy poniższe metody i modele:

- Hard Voting
- Soft Voting

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

4.1.1 Hard i Soft Voting

Dla Hard i Soft Votingu zdecydowaliśmy się jako estymatory wybrać modele, które dawały najlepsze wyniki w poprzedniej części projektu, jak i dołączyć kilka modeli, których nie sprawdzaliśmy wcześniej lub które myśleliśmy, że mogą dawać wysokie wyniki, jeśli coś się w nich poprawi lub zmieni.
Wybrane modele to:

- model0 = RandomForestClassifier(random_state=1)
- model1 = DecisionTreeClassifier(random_state=1)
- model2 = KNeighborsClassifier()
- model3 = LogisticRegression(random_state=1, max_iter=1000) - jeden z lepszych w KM2
- model4 = MLPClassifier(random_state=42, max_iter=1000) - jeden z lepszych w KM2
- model5 = GradientBoostingClassifier(random_state=1)
- model6 = GaussianNB() - najlepiej się sprawdzał w KM2
- model7 = QuadraticDiscriminantAnalysis() - jeden z lepszych w KM2

Wyniki były następujące:

- Hard Voting

```
f1 score:  0.2711864406779661
model.score:  0.6884057971014492
              precision    recall  f1-score   support

           0       0.74      0.88      0.80        99
           1       0.40      0.21      0.27        39

    accuracy                           0.69       138
   macro avg       0.57      0.54      0.54       138
weighted avg       0.64      0.69      0.65       138
```

Figure 11: classification_report dla Hard Voting

- Soft Voting bez wag

```
              precision    recall  f1-score   support

           0       0.76      0.75      0.75        99
           1       0.38      0.38      0.38        39

    accuracy                           0.64       138
   macro avg       0.57      0.57      0.57       138
weighted avg       0.65      0.64      0.65       138
```

Figure 12: classification_report dla Soft Voting bez wag

- Soft Voting z wagami - z wypróbowanych wag najlepiej sprawdzały się takie, które najwyższe wagi przyporządkowywały modelom, które najlepiej sprawdzały się w poprzedniej części - najlepsze wypróbowane wagi to: weights=[1,1,1,5,1,1,25,15]

```
              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]

Z powyższych metod, patrząc na recall i f1, zdecydowanie najlepsze wyniki daje Soft Voting z wagami [1,1,1,5,1,1,25,15].

4.1.2 Stacking

Używając StackingClassifier wypróbowaliśmy dla niego dwa parametry, używając tych samych estymatorów, co do Soft i Hard Votingu:

- final_estimator=LogisticRegression()

	precision	recall	f1-score	support
0	0.75	0.91	0.82	99
1	0.50	0.23	0.32	39
accuracy			0.72	138
macro avg	0.62	0.57	0.57	138
weighted avg	0.68	0.72	0.68	138

Figure 14: classification_report dla StackingClassifier z final_estimator=LogisticRegression()

- final_estimator=GaussianNB()

	precision	recall	f1-score	support
0	0.77	0.81	0.79	99
1	0.44	0.38	0.41	39
accuracy			0.69	138
macro avg	0.61	0.60	0.60	138
weighted avg	0.68	0.69	0.68	138

Figure 15: classification_report dla StackingClassifier z final_estimator=GaussianNB()

4.1.3 Bagging

BaggingClassifier wypróbowaliśmy używając wszystkich modeli z tych wybranych jako estymatory i model6 (GaussianNB) dawał zdecydowanie najlepsze wyniki:

	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

4.1.4 RandomForestClassifier

Ten Classifier wypróbowaliśmy tylko dla parametrów użytych na zajęciach laboratoryjnych: n_estimators=1000, max_depth=3, min_samples_split = 2, max_features = 3, random_state=0, n_jobs = -1. Z powodu bardzo jego niskich wyników zdecydowaliśmy się porzucić dalsze badanie go i jego parametrów.

	precision	recall	f1-score	support
0	0.72	0.99	0.83	99
1	0.50	0.03	0.05	39
accuracy			0.72	138
macro avg	0.61	0.51	0.44	138
weighted avg	0.66	0.72	0.61	138

Figure 17: classification_report dla RandomForestClassifier z parametrami n_estimators=1000, max_depth=3, min_samples_split = 2, max_features = 3, random_state=0, n_jobs = -1

4.1.5 AdaBoostClassifier

Ten Classifier również nie dawał satysfakcjonujących wyników:

	precision	recall	f1-score	support
0	0.76	0.83	0.79	99
1	0.43	0.33	0.38	39
accuracy			0.69	138
macro avg	0.60	0.58	0.58	138
weighted avg	0.67	0.69	0.67	138

Figure 18: classification_report dla AdaBoostClassifier z random_state=1

4.1.6 GradientBoosting

Dla GradientBoostingu wypróbowaliśmy różne wartości parametrów, jednak dla żadnej w nich wynik nie był nawet bliski satysfakcjonującego. Dlatego też zdecydowalismy się zostawić ten model i nie używać go.

	precision	recall	f1-score	support
0	0.78	0.88	0.82	99
1	0.54	0.36	0.43	39
accuracy			0.73	138
macro avg	0.66	0.62	0.63	138
weighted avg	0.71	0.73	0.71	138

Figure 19: classification_report dla GradientBoosting z parametrami random_state=1, learning_rate=0.5

4.1.7 XGBClassifier

Dla tego modelu wypróbowaliśmy dwa różne, zmienione parametry:

- booster='gbtree'

	precision	recall	f1-score	support
0	0.75	0.95	0.84	99
1	0.58	0.18	0.27	39
accuracy			0.73	138
macro avg	0.66	0.56	0.56	138
weighted avg	0.70	0.73	0.68	138

Figure 20: classification_report dla XGBClassifier z parametrami random_state=1, learning_rate=0.01, booster='gbtree', max_depth=4

- booster='gblinear'

	precision	recall	f1-score	support
0	0.75	0.92	0.83	99
1	0.53	0.23	0.32	39
accuracy			0.72	138
macro avg	0.64	0.57	0.57	138
weighted avg	0.69	0.72	0.68	138

Figure 21: classification_report dla XGBClassifier z parametrami random_state=1, learning_rate=0.01, booster='gblinear', max_depth=4

4.1.8 AutoML i TPOT

Oba te narzędzia dawały słabe, niesatysfakcjonujące nas wyniki. Zdecydowaliśmy się więc je porzucić.

4.2 Strojenie hiparametrów

Do wybrania najlepszych paramatrów modeli wypróbowaliśmy takie metody, jak:

- GridSearch
- RandomizedSearch
- BayesSearch

Jako parametr „scoring" dla tych technik używaliśmy 'f1', ponieważ to i recall są dla nas najważniejsze, ale jednak nie chcemy szukać wyniku po samym recall, ponieważ chcemy też często dawać kredyty.

4.2.1 GridSearch

GridSearch użyliśmy do takich metod, jak:

- DecisionTreeClassifier
  W tej metodzie przeszukaliśmy takie parametry, jak: max_depth=[3, 5, 6, 7, 8]
  ccp_alpha=[0,0.01, 0.02, 0.05, 0.1, 0.2]
  splitter=["best", "random"]
  min_samples_split=[2, 5, 10]
  min_samples_leaf=[1, 2, 4]
  max_features=["auto", "sqrt", "log2"]
  Nasze wyniki:

```
Best: 0.386175 using {'ccp_alpha': 0, 'max_depth': 8, 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
```

Figure 22: najlepsze parametry przy scoring='f1' wg GridSearch dla DecisionTreeClassifier

Sprawdziliśmy jeszcze, czy te same parametry wychodzą jako najlepsze, jeśli zmienimy scoring na 'recall':

```
Best: 0.396238 using {'ccp_alpha': 0, 'max_depth': 8, 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Figure 23: najlepsze parametry przy scoring='recall' wg GridSearch dla DecisionTreeClassifier

Classification_report dla najlepszych parametrów wg scoring='f1' przedstawia się następująco:

	precision	recall	f1-score	support
0	0.71	0.86	0.78	99
1	0.22	0.10	0.14	39
accuracy			0.64	138
macro avg	0.47	0.48	0.46	138
weighted avg	0.57	0.64	0.60	138

Figure 24: classification_report dla najlepszych parametrów wg GridSearch przy scoring='f1 w metodzie DecisionTreeClassifier

Model ten przedstawia się bardzo słabo w naszym projekcie, wyniki wg scoring='recall' są równie niskie.

- GradientBoostingClassifier
  Tutaj przeszukaliśmy takie parametry, jak: max_depth = [3, 5, 6, 7, 8] loss = ["deviance", "exponential"] learning_rate = [0.001, 0.01, 0.1] n_estimators = [10, 50, 100, 200] subsample = [0.5, 0.75, 1.0] max_features = ["auto", "sqrt", "log2"]

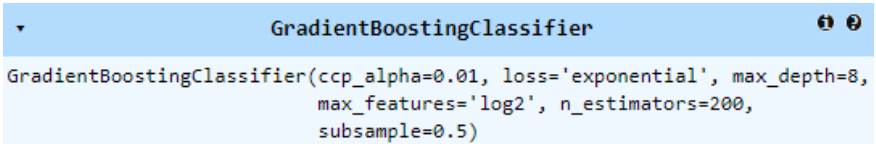
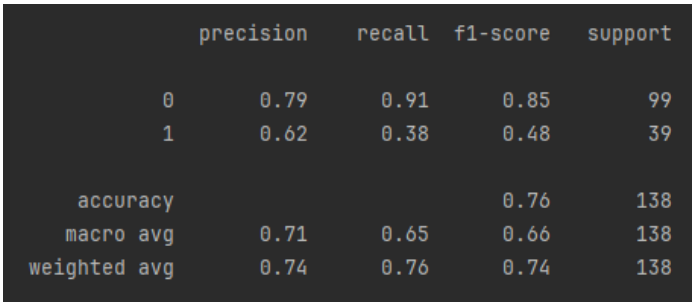


Figure 25: najlepsze parametry dla GradientBoostingClassifier wg GridSearch



	precision	recall	f1-score	support
0	0.79	0.91	0.85	99
1	0.62	0.38	0.48	39
accuracy			0.76	138
macro avg	0.71	0.65	0.66	138
weighted avg	0.74	0.76	0.74	138

Figure 26: classification_report dla najlepszych parametrów wg GridSearch dla GradientBoostingClassifier

- RandomForestClassifier
  Przeszukiwane parametry: max_depth = [3, 5, 6, 7, 8] criterion = ["gini", "entropy"] ccp_alpha = [0, 0.01, 0.02, 0.05, 0.1, 0.2] n_estimators = [10, 50, 100, 200] min_samples_split = [2, 5, 10] min_samples_leaf = [1, 2, 4] bootstrap = [True, False] Wyniki:

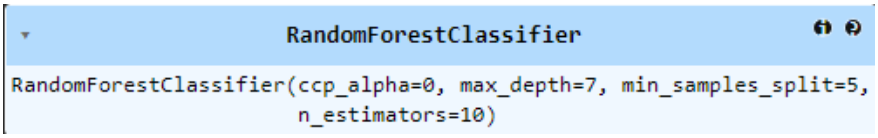


Figure 27: najlepsze parametry dla RandomForestClassifier wg GridSearch

	precision	recall	f1-score	support
0	0.74	0.94	0.83	99
1	0.50	0.15	0.24	39
accuracy			0.72	138
macro avg	0.62	0.55	0.53	138
weighted avg	0.67	0.72	0.66	138

Figure 28: classification_report dla najlepszych parametrów wg GridSearch dla RandomForestClassifier

- GaussianNB
  Tutaj nie było wiele parametrów do sprawdzenia, ale ze względu na to, że model ten dawał nam do tej pory najbardziej satysfakcjinujące nas wyniki, chcieliśmy spróbować ulepszyć go jeszcze bardziej, dobierając jak najlepsze parametry (sprawdziliśmy je aż trzema technikami: GridSearch, RandomSearch i BayesSearch).
  Przeszukiwane parametry: 'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5]
  Wyniki:

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

Figure 29: najlepsze parametry dla GausianNB wg GridSearch i classification_report tego modelu z tymi parametrami

- QuadraticDiscriminantAnalysis
  Ten model również dawał nam wcześniej w miarę dobre wyniki w porównaniu z resztą modeli, więc postanowiliśmy przeszukać i dobrać jego parametry:

'reg_param': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
'tol': [1e-3, 1e-4, 1e-5]
'priors': [None, [0.5, 0.5], [0.7, 0.3], [0.3, 0.7]]
'store_covariance': [True, False]
Wyniki:

```
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
```

Figure 30: najlepsze parametry dla QuadraticDiscriminantAnalysis wg GridSearch i classification_report tego modelu z tymi parametrami

Niektóre z nich wykazywały się wysokimi, w porównaniu z resztą, wynikami w poprzedniej części projektu, dlatego chcieliśmy spróbować sprawić, aby wyniki te były jeszcze lepsze, za pomocą doboru odpowiednich parametrów. Inne z nich sprawdziliśmy ze względu na chęć sprawdzenia, czy mają szansę dać wyższe wyniki niż z poprzednio użytymi parametrami. Byliśmy ciekawi, jak mocno dobranie dobrych parametrów wpływa na wyniki modeli.

4.2.2 RandomizedSearch i BayesSearch

Za pomocą metod RandomizedSearchCV oraz BayesSearchCV sprawdzaliśmy jedynie parametry dla modelu GaussianNB, ponieważ jak na razie dawał on najlepsze wyniki.
Jako parametr param_distribution dla Random Search daliśmy param_grid = 'var_smoothing': [1e-12, 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2], a dla Bayes Search 'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5].
Nasze wyniki:

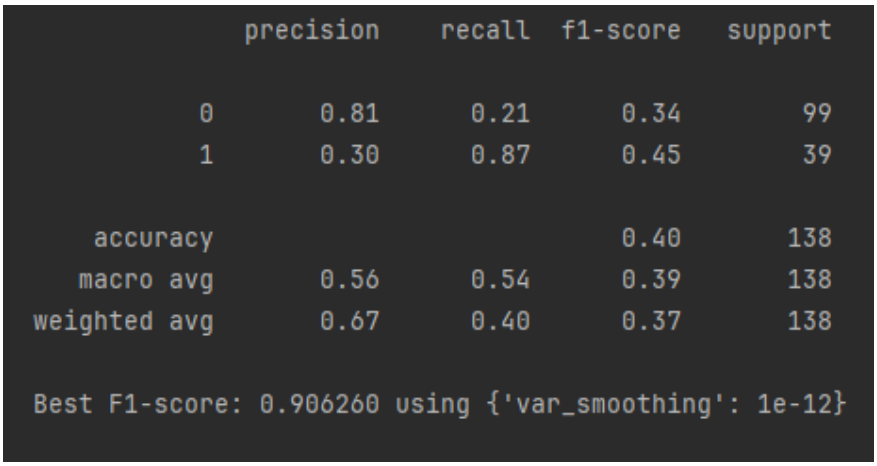


Figure 31: najlepsze parametry dla GausianNB wg Random Search i classification_report dla tego modelu ze znalezionymi parametrami

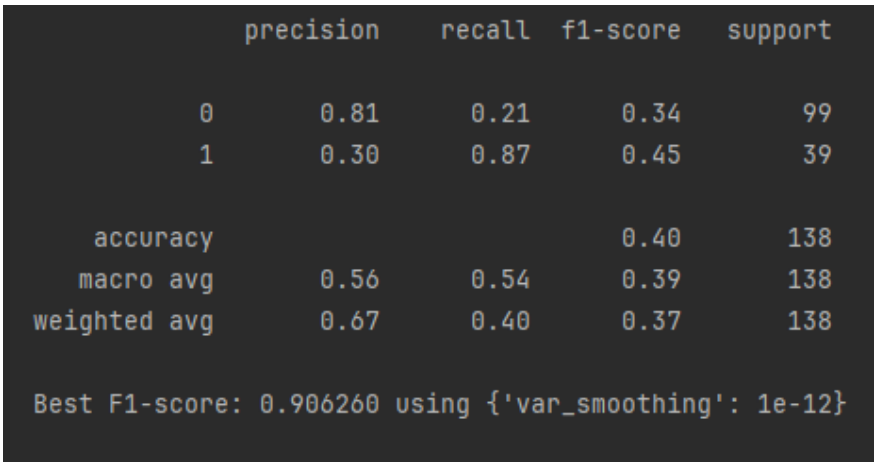


Figure 32: najlepsze parametry dla GausianNB wg Bayes Search i classification_report dla tego modelu ze znalezionymi parametrami

4.3 Podsumowanie wyników modeli

Najlepsze wyniki z przestestowanych przez nas modeli i ich parametrów miały:

- VotingClassifier(estimators=estimators[0:8], voting='soft', weights=[1,1,1,5,1,1,25,15])
- BaggingClassifier(estimator=GaussianNB() ,n_estimators=10, random_state=0) (troche gorsze niz 1)

- GausianNB(var_smoothing = 1e-05)
- GausianNB(var_smoothing = 1e-12)
- QuadraticDiscriminantAnalysis(priors =[0.3, 0.7], reg_param = 0.3, store_covariance =True, tol = 0.001)

4.4 Kroswalidacja

Zadanie kroswalidacji jest sprawdzenie stabilności naszych modeli dla różnych rodzajów danych. Jesteśmy w stanie zobaczyć jak te modele zachowują się w różnych przypadkach i uniknąć sytuacji, że zbiór treningowy zawsze przypadkowo daje dużo informacji modelowi jak nauczyć się zależności o zbiorze walidacyjnym.

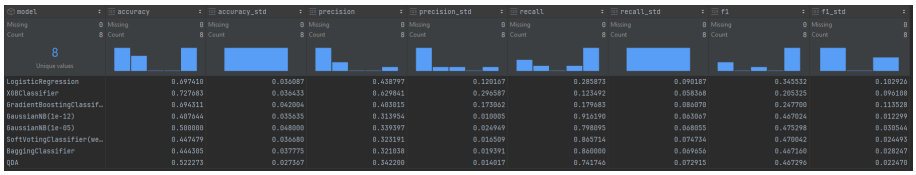


Figure 33: Tabelka z informacjami z crossvalidation dla każdego modelu, średnia z każdej metryki oraz odchylenie standardowe, co może ocenić stabilność predykcji

Widzimy, że patrząc na odchylenie standardowe precision i miary f1 następujące modele są bardzo niestabilne:

- LogisticRegression
- XGBClassifier
- GradientBoostClassifier

Zauważamy, że komitety i inne sposoby na łączenie modeli może poradzić sobie z problemem.

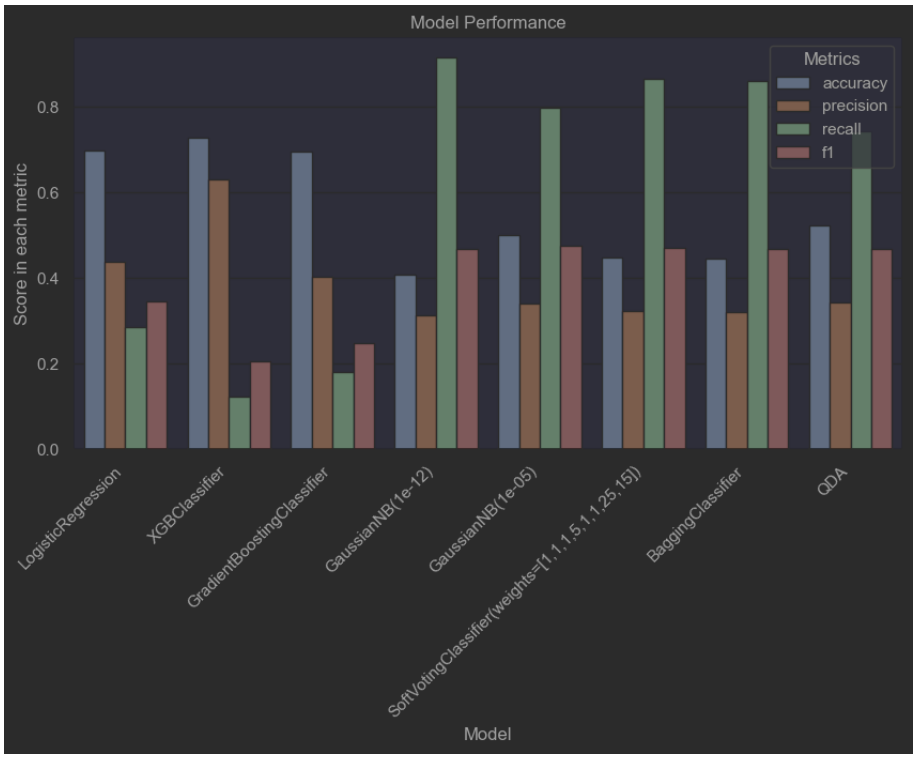


Figure 34: Wykres z informacjami z crossvalidation dla każdego modelu, średnia z każdej metryki.

Jeśli spojrzymy na średnią z metryk możemy zauważyć, że następujące modele są najlepsze:

- GaussianNB(1e-12)
- GaussiaNB(1e-05)
- SoftVotingClassifier
- Bagging Classifier
- QDA

Ale faworytem jest pierwszy, ponieważ maksymalizuje metrykę recall, co częściowo realizuje nasz problem biznesowy, ponieważ wyłapujemy (w miarę stabilnie - odchylenie standardowe na poziomie 6%) wtedy 90% ludzi którzy nie spłacą kredytu, co zminimalizuje straty finansowe banku.

4.5 Wybrane modele i metody wyjaśnialności

Początkowo (przed walidacją) modele, które wybraliśmy, aby "przeszły" do kolejnego etapu (czyli metod wyjaśnialności) były to: XGBoost, Naive Bayes oraz

soft voting. To właśnie te modele sprawdzaliśmy jak działają, używając różnych metod.
Jednkaże ostatecznie, po etapie walidacji i poprawieniu rożnych elementów w modelowaniu i samym już wyborze najlepszych modeli, uznaliśmy że najlepszym modelem (i tym którego efekty z metod wyjaśnialności pokaże tutaj) jest GaussianNB(1e-12). Jest to model, który ma zdecyowanie najwyższy recall i jeden z wyższych f1, czyli spełnia dokładnie to na czym nam zależy.

Z różnych metod wyjaśnialności, głównie skupialiśmy sie na SHAP oraz Partial Dependance Plots, gdyż według nas dają one najbardziej czytelne i przydatne rezultaty (testowaliśmy też m.in. metodę LIME, którą zasugerowali nam walidatorzy, jednak według nas nie dawał on użytecznych, w naszym projekcie, wyników). W przypadku jednak modelu GaussianNB metoda SHAP nie jest możliwa, a przynajmniej mało efektywna, ponieważ mając tak duży zbiór danych jak nasz, wykonanie się tego zajmie zdecydowanie bardzo długo.

4.5.1 Partial Dependance Plots

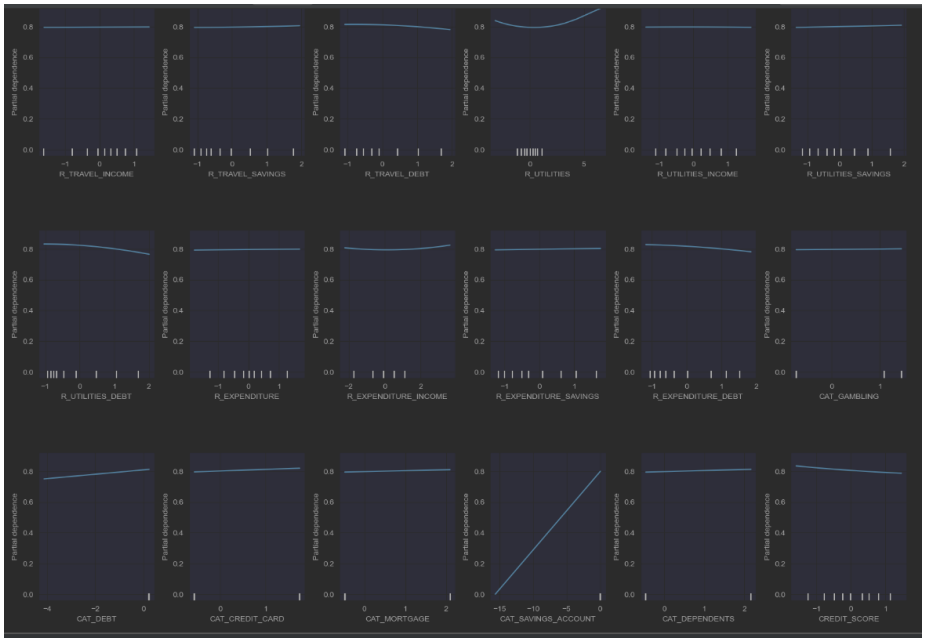


Figure 35: Część wykresów wygenerowanych dzięki PDP dla modelu GaussianNB(1e-12)

Na powyżej przedstawionym zdjeciu widać część wykresów wygenerowanych dzięki PDP dla tego modelu. Niestety wszytskich zmiennych (72) jest za dużo, aby wszystkie wykresy móc pokazać w raporcie. Jednakże większość wykresów

niestety wyglądała podobnie - tzn. większość zmiennych ma marginalny wpływ na wynik modelu. Jedyną zmienną, która ma duży wpływ na wynik modelu jest 'CAT_SAVINGS_ACCOUNT' (której wykres jest widoczny w zamieszczonym zdjęciu).Jest to zaskakujące odkrycie, gdyż jednak jest to model dający najlepsze rezultaty i wyniki.

4.6 Walidacja

Na tym etapie uwagi walidatorów były bardzo cenne, spowodowały one zmianę modelu, który uznaliśmy za ostateczny i najlepszy. Pozwoliły osiągnąć lepsze wyniki, niż się spodziewaliśmy przed otrzymaniem informacji zwrotnej od grupy.

Table 2: Sugestie od walidatorów na etapie KM3

Sugestie	Czy wzięto pod uwagę?
Warto uwzględnić wyjaśnienie wyboru zestawu estymatorów i brak testowania innych kombinacji. Sprawdźcie, czy identyczne wyniki w głosowaniu są wynikiem błędu w kodzie. Rozważcie eksperymentowanie z różnymi wagami.	Tak
Eksperymentujcie z różnymi kombinacjami estymatorów.	Tak
Rozważcie większą różnorodność w wyborze modeli lub zmiennej liczby kolumn.	Tak
Brakujący opis może być przyczyną niezrozumienia tego, co jest sprawdzane w tej sekcji. Badanie wartości NaN dla średnich i odchyleń standardowych wymaga zgłębienia. Sprawdźcie problem z wykonaniem kodu dla Random Forest Classifier.	Tak
Dobre wyniki otrzymane z klasyfikatora TPOT, dlaczego nie jest on brany pod uwagę?	Tak
Dodajcie komentarze wyjaśniające wybór konkretnych modeli i który z nich jest ostatecznym kandydatem.	Tak
Rozważcie dodanie wykresu ROC i macierzy pomyłek dla najlepszych modeli.	Nie
Sprawdźcie na zbiorze testowym tylko finalny wybrany model.	Tak
Skoncentrujcie się na wykresach PDP, rozważcie użycie LIME dla lepszych wizualizacji.	Tak

5 Podsumowanie

5.1 Finalne wyniki na zbiorze testowym

Użyliśmy tutaj zbioru testowego, którego mieliśmy użyć dopiero na końcu.

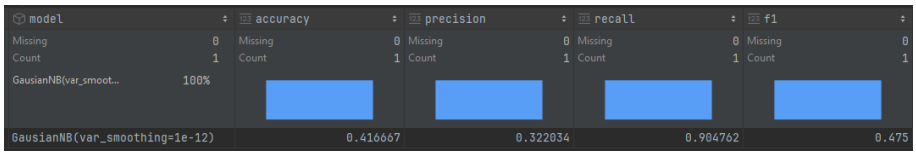


Figure 36: Tabelka z informacjami za pomocą zbioru testowego dla finalnego modelu.

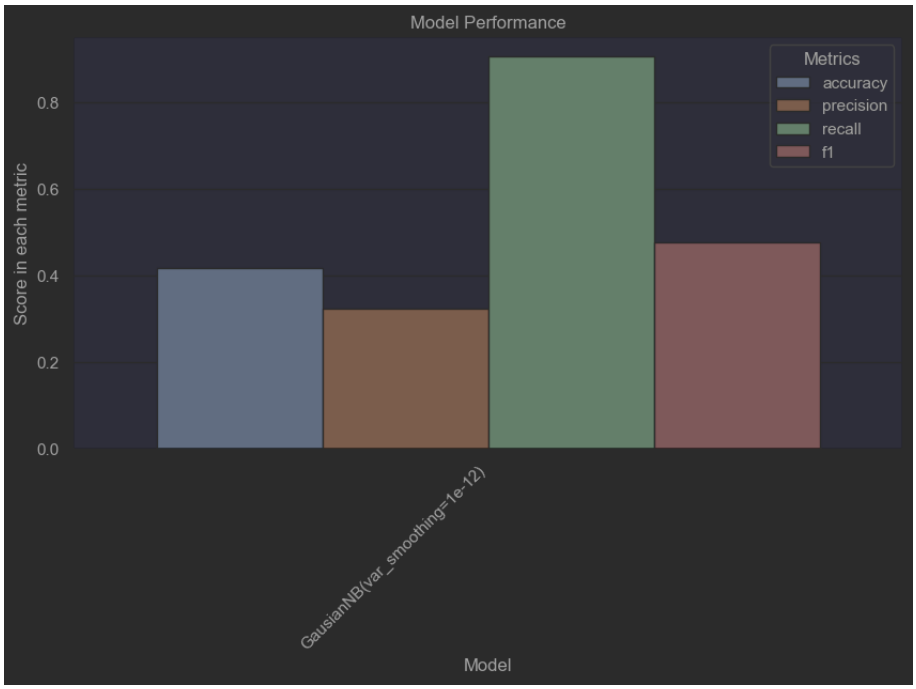


Figure 37: Wykres z informacjami za pomocą zbioru testowego dla finalnego modelu.

5.2 Czy zrealizowano cel biznesowy?

Jak wcześniej zauważono rozwiązanie problemu może się nie udać z wielu powodów. De facto każdy przypadek niespłacenia kredytu jest kompletnie inny, do wielu cech klientów nie mieliśmy dostępu, a niektóre nie są nawet mierzalne czy możliwe do przewidzenia.
Jednakże osiągnęliśmy mały sukces, bo znaleźliśmy model, który maksymalizuje recall i to na zadawalającym poziomie i w miarę stabilnie. Wyłapujemy znaczną większość osób, które nie spłacają kredytu. Moglibyśmy teraz sprzedać ten klasyfikator jako bardzo bezpieczny klasyfikator dla finansów banku, szczególnie dla banku, który ma problemy finansowe i nie chce ryzykować.

6 Źródła

link do repozytorium naszego projektu - https://github.com/fantasy2fry
link do ramki danych, z których korzystaliśmy - https://www.kaggle.com/datasets/conorsully1/credit-score