Przewidywanie niewypłacalności kredytobiorców na podstawie danych Credit Score

Pahasian Milanna *
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

Badzeika Hleb *
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

Bokhan Katsiaryna *
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

* - równoważność

23rd April 2024

1 Zbiór danych

Dane przydzielone naszemu zespołowi - Credit Score [1]. To ramka danych zawierająca 87 kategorii (kolumn) i 1000 obserwacji, które są sztucznie wygenerowanymi danymi, może być, według ustalonych rozkładów. Celem było przewidywanie niewypłacalności (DEFAULT) - zmiennej binarnej, gdzie 1 oznacza, że osoba zalegała z płatnościami, 0 że nie.

Ramka danych również zawiera informacje o zdolności kredytowej (CREDIT_SCORE), które mogłyby być również targetem, ale skupiliśmy się jednak na niewypłacalności, oraz pozbyliśmy się tej kolumny, bo tak naprawdę zrobiony przez nas model w jakimś sensie sam musi każdej obserwowanej osobie przyporządkować 'credit_score', żeby na jego podstawie zrobić wnioski o DEFAULT.

1.1 Opis zmiennych (kolumn)

Jak już wspomniano wcześniej, zbiór danych składa się z 87 kolumn zawierających informacje o

- niewypłacalności (DEFAULT)
- zdolności kredytowej (CREDIT_SCORE)
- identyfikatorze osoby (CUST_ID)
- dochodach (INCOME), oszczednościach (SAVINGS), zadłużeniach (DEBT)
- wydatkach na odzież (T_CLOTHING), edukację (T_EDUCATION), rozrywkę (T_ENTERTAINMENT), mandaty (T_FINES), kasyno (T_GAMBLING), artykuły spożywcze (T_GROCERIES), zdrowie (T_HEALTH), mieszkanie (T_HOUSING), podatki (T_TAX), podróży (T_TRAVEL), usługi komunalne (T_UTILITIES), ?? (T_EXPENDITURE) w ciągu ostatnich 6 i 12 miesięcy
- posiadaniu karty (CAT_CREDIT_CARD, (0 lub 1)), obecności długów (CAT_DEBT, (0 lub 1)), poziom uzależnienia osoby od gier (CAT_GAMBLING, ('No', 'Low', 'High')), kredytu hipotecznym (CAT_MORTGAGE, (0 lub 1)), koncie oszczędnościowym (CAT_SAVINGS_ACCOUNT, (0 lub 1)), obecność osób na utrzymaniu (CAT_DEPENDENTS, (0 lub 1))
- wszystkie pozostałe kolumny to różne stosunki wyżej opisanych zmiennych (na przykład R_HEALTH_SAVINGS = HEALTH/SAVINGS)

2 Exploratory Data Analysis

2.1 Braki danych

W naszym zbiorze nie było żadnych braków danych.

2.2 Rozkład zmiennej celu oraz zdefiniowanie zbioru, na którym my będziemy pracować

Na początku podzieliliśmy nasz zbiór danych (1000 obserwacji) na zbiory treningowy (800 obserwacji) i walidacyjny (200 obserwacji). Zbiór walidacyjny został przekazany naszemu zespołowi walidacyjnemu. Rozkład zmiennej DEFAULT w tych zbiorach można zobaczyć na Fig.1. Dalej będziemy pracować wyłącznie na zbiorze treningowym.

Dodatkowo na tych barplotach widzimy, że mamy styczność z niezbalansowanymi danymi: DEFAULT = 0 w zbiorze jest znacznie więcej niż DEFAULT = 1. To teoretycznie może mieć wielki wpływ na trenowanie modeli.

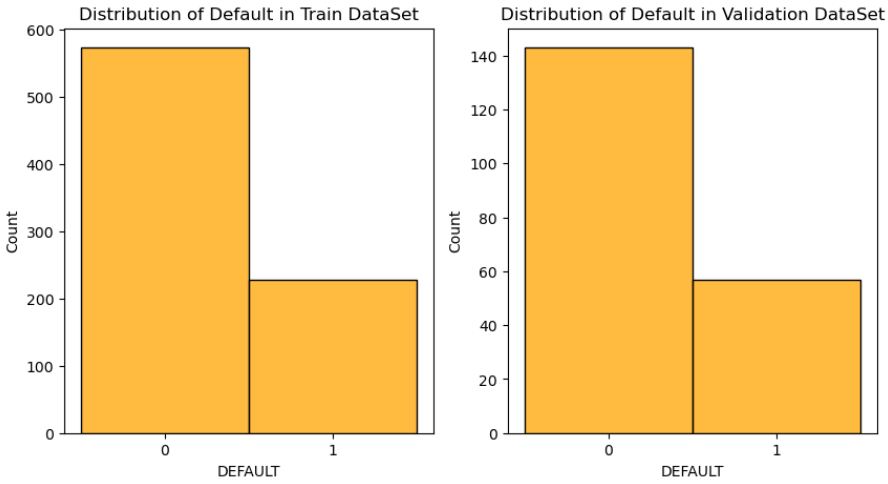

Figure 1: Rozkład zmiennej celu

2.3 Rozkład zmiennej celu (DEFAULT) w zależności od innych zmiennych

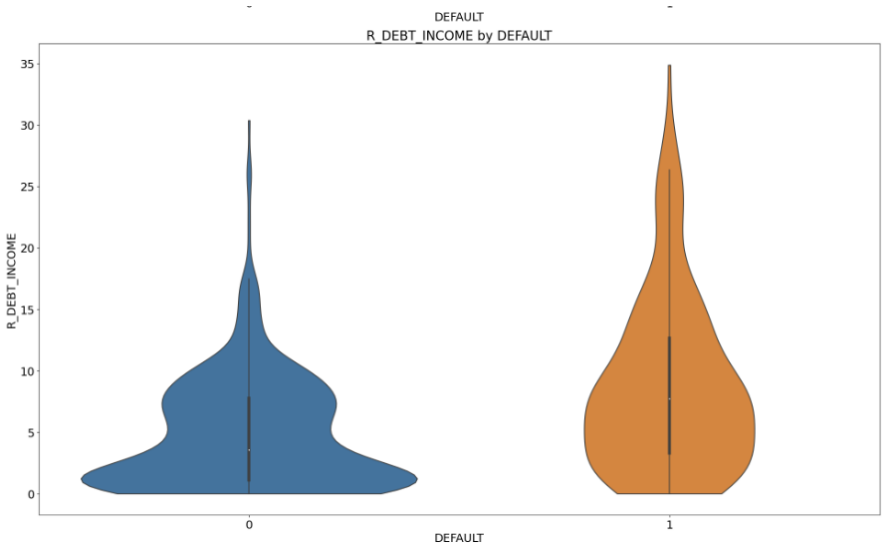

Figure 2: Rozkład zmiennej celu w zależności od R_DEBT_INCOME

Chcieliśmy zrozumieć, jakie zmienne mają wpływ na DEFAULT. W tym celu zrobiliśmy violinploty (rozkład zmiennej DEFAULT w zależności od innej zmiennej). Najciekawszym się okazał wykres na Fig.2. Więc prawdopodobnie, zmienna R_DEBT_INCOME będzie jedną z najważniejszych dla modeli, przewidujących DEFAULT.

2.4 Rozkłady zmiennych i je przekształcenia/tranformacje

Wśród wszystkich naszych danych były tylko 2 kolumny kategoryczne: CUST_ID oraz CAT_GAMBLING, wszystkie inne były numeryczne. Pierwsza została usunięta z powodu braku informacji istotnych, a druga (przyjmuje wartości "Low", "No", "High") została przekształcona za pomocą Ordinal Encoding (bo "No" $<$ "Low" $<$ "High"), czyli 'No' -> 1, 'Low' -> 2, 'High' -> 3.

Zrobiliśmy histploty dla każdej kolumny (jest ich 86, więc tutaj nie umieszczamy je), które później zostały użyte do wykrywania outlierów. Prawie wszystkie kolumny mają rozkłady normale lub podobne do eksponencjalnych, więc żadna transformacja nie była potrzebna.

Przez nas została zauważona ciekawa rzecz: niektóre wartości R_{GROUP} (na Fig.3 jest przykład na kolumnie R_FINES) miały wartość większą niż 1, co nie zgadza się z tym, że R_{GROUP} = T_{GROUP}_6/T_{GROUP} bo kwota wydana w ostatnich 6 miesięcach nie może być większa niż w ciągu ostatnich 12.

Zaczęliśmy próbować zrozumieć, dlaczego tak się zdarzyło. Zauważyliśmy, że te mylące dane są spowodowane dzieleniem przez 0, czyli wartością NaN w R_{GROUP} (gdy T_{GROUP}_12 = 0), i jej późniejszym zastąpieniem za pomocą jakiegoś Imputera.

Spróbowaliśmy użyć różnych imputerów, żeby samodzielnie zastępować powstające podczas dzielenia przez 0 wartości NaN: zastosowaliśmy KNNImputer oraz IterativeImputer, ale to w żadny sposób nie zmieniło wyników (było sprawdzone na Gradient Boosting Classifier). Domyślamy się, przyczyną jest to, że dane choć są sztucznie wygenerowane, są zgodne rozkładom.

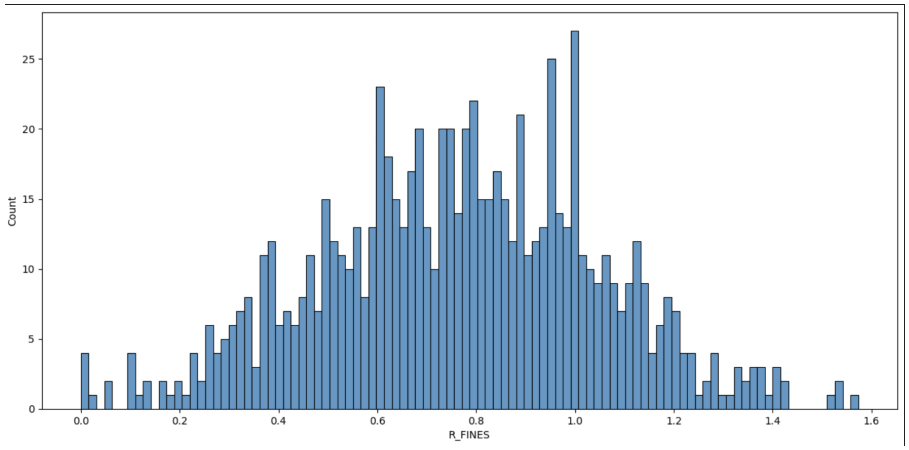

Figure 3: Rozkład zmiennej R_FINES

Następnym pytaniem badania było: W których kolumnach, przez które dzielimy, jest tyle zer, że w końcu z powodu imputacji danych okazało się za dużo danych zastąpionych imputerem? Wyniki można zobaczyć na Fig.4. Więc było postanowiono, że dla kolumny R_FINES, R_GAMBLING, R_EDUCATION, R_HOUSING będą usunięte z powodu dużej liczby braków danych.

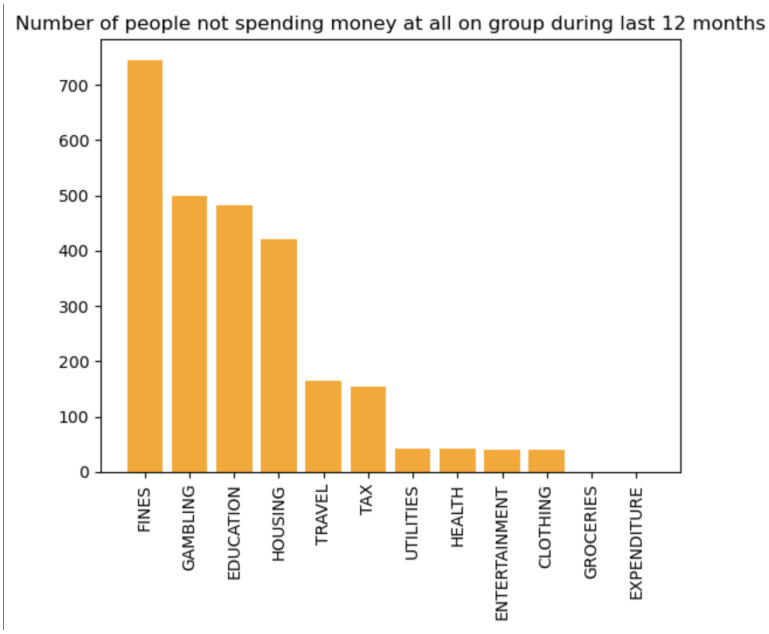

Figure 4: Liczba zer w kolumnie T_{GROUP}_12

2.5 Wartości odstające (Outliers)

Korzystając z rozkładów zmiennych zauważyliśmy, że outlierów w naszym zbiorze prawie nie ma. Tylko w kolumnach SAVINGS, DEBT, T_CLOTHING_12, T_CLOTHING_6, T_HEALTH_12, T_HEALTH_6, T_TRAVEL_12, T_TRAVEL_6 jest kilka sztuk. Więc w tych kolumnach, jeżeli była spotykana bardzo duża wartość, to ona była zastąpiona największą wartością, która nie była wartością odstającą.

2.6 Zmienne o nizkiej wariancji

W naszym zbiorze danych wszystkie zmienne mają wystarczająco duże warości wariancji do pracy.

2.7 Skorelowane kolumny

Zrobiliśmy heatmapę dla 86 kolumn (tutaj nie możemy jej umieścić z powodu dużego rozmiaru) zbioru danych, gdzie były zauważone mocno skorelowane kolumny. Więc było postanowiono, że jeżeli 2 kolumny mają korelację >95%, to jedna z nich musi być wyrzucona. Są to kolumny: T_EXPENDITURE_12, T_EDUCATION_6, T_ENTERTAINMENT_6, T_GAMBLING_6, T_GROCERIES_6, R_UTILITIES_DEBT, T_HOUSING_6, T_TAX_6, T_UTILITIES_6.

3 Eksperymentowanie z różnymi modelami uczenia maszynowego

Celem wynikowego modelu jest predyktowanie DEFAULT, więc długo się zastanawili nad tym, jaka metryka jest dla nas najważnejsza. Po szczegółowej analizie postanowiliśmy, że naszą główną metryką jest Accuracy (balans dla prawidłowego zgadywania 0, i 1 wartości DEFAULT), ale jednak też jest dla nas ważna metryka recall 1 (pozytywna predykcja niewypłacalności) - jest to spowodowane tym, że predykcja defaultu jest istotna dla n.p. banku, bo minimalizyje ryzyko. Ale jednocześnie jeżeli recall 1 będzie duży, to jak zobaczyliśmy podczas trenowania modeli, accuracy bardzo spada, czyli bank bardzo bardzo często nie wydaje kredytów tym osobom, które go spłacą . Więc trzeba bardzo sensownie webierać model: musi mieć dobrą accuracy i nie za małą recall 1.

Krótkie opisy niektórych trenowanych przez nas modeli:

Random Forest: Pomysłem na tę metodę było użycie PCA(Principal Components Analysis) do redukcji rozmierności. Dlatego oprócz wymienionych w części EDA skorelowanych kolumn, wyrzuciliśmy pozostałe kolumny typu T_GROUP_6 (i zostawiliśmy T_GROUP_12) orza kolumny R_GROUP, bo pokazują stosunki T_GROUP_6 i T_GROUP_12 (rozkład podobny do normalnego o wartości oczekiwanej 0.5) i nie dostarczają istotnych dla PCA informacji. W punkcie 2.3 jest opisane zastąpenie wartości odstających. Metoda PCA jest wrażliwa na różnicy w skalach, więc użyliśmy StandardScaler. Za pomocą RandomizedSearchCV wyznaczyliśmy hyperparametry dwóch modeli; użyliśmy metodę kroswalidacji.

Support Vector Classification: najpierw użyliśmy PipelineBasic (z basic_pipeline_functions), który przekształca dane według zasad ustalonych podczas EDA (ważną rzeczą jest to, że z danych które służą do przewidywania DEFAULT trzeba ręcznie usunąć kolumny 'CREDIT_SCORE','DEFAULT'). Do selekcji zmiennych użyliśmy SelectKBest, wybraliśmy hyperparametry za pomocą GridSearchCV. Zwalidowaliśmy wyniki metodą cross-validation.

Logistic Regression: najpierw został użyty PipelineBasic (stworzony przez nas), który robi wszystkie zmiany z danymi, o których postanowiliśmy podczas EDA (outliery, bezsensowne kolumny itp.). Potem został użyty StandardScaler, bo Logistic Regression jest czujna na różniy w scalach. Do feature selection została użyta metoda Recursive Feature Elimination oraz dla przeszukiwania najlepszych hyperparametrów (feature_selection__n_features_to_se classifier__solver, classifier__C) podczas cross walidacji został użyty GridSearch.

XGBOOST Classifier: najpierw został użyty PipelineBasic (stworzony przez nas), nie normalizowaliśmy naszych danych, bo XGBoost jest oparty na drzewach decyzyjnych, które nie są czujne na różnicy w danych. Do feature selection został użyty PCA. Oraz za pomocą BayesSearch podczas cross walidacji, najlepsze hyperparametry (pca__n_components, classifier__learning_rate, classifier__n_estimators, classifier__max_depth, classifier__min_child_weight, classifier__subsample, classifier__colsample_bytree, classifier__reg_alpha, classifier__reg_lambda, classifier__gamma) zostały znalezione.

Gradient Boosting Classifier: najpierw został użyty PipelineBasic (stworzony przez nas), nie normalizowaliśmy naszych danych, bo Gradient Boosting Classifier jest oparty na drzewach decyzyjnych, które nie są czujne na różnicy w danych. Do feature selection został użyty PCA. Oraz za pomocą BayesSearch podczas cross walidacji, najlepsze hyperparametry (pca__n_components, classifier__learning_rate, classifier__n_estimators, classifier__max_depth, classifier__min_samples_split, classifier__min_samples_leaf, classifier__subsample) zostały przeszukane.

Stacking: Stacking to technika łączenia różnych modeli w celu zwiększenia dokładności predykcji. W tym przypadku, jako bazowe modele do stacking wykorzystano XGBoost, Logistic Regression, SVC oraz Gradient Boosting Classifier. Każdy z tych modeli jest trenowany niezależnie, a ich prognozy są używane jako cechy dla finalnego modelu meta-klasyfikatora (również Logistic Regression).

Meta-klasyfikator jest trenowany na prognozach z modeli bazowych, co pozwala mu nauczyć się, który model bazowy jest najbardziej wiarygodny w zależności od danych wejściowych. To sprawia, że stacking jest bardzo skuteczną metodą do poprawy ogólnej wydajności modelu, szczególnie w skomplikowanych problemach klasyfikacyjnych.

Model	Accuracy (c-v)	Recall 1 (c-v)	Accuracy (T)	Recall 1 (T)
Logistic Regression	0.734375	0.231	0.74375	0.22
Random Forest	0.734	0.127	0.742	0.16
Stacking	0.732	0.143	0.74	0.22
Support Vector Classification	0.731	0.094	0.744	0.11
Gradient Boosting Classifier	0.7297	0.1264	0.73125	0.11
XGBOOST Classifier	0.7234375	0.115	0.71875	0.089

Table 1: Best accuracy for models

Próbowaliśmy również stosować GaussianNB i KNeighborsClassifier, ale wyniki były jeszcze gorsze.

4 Transformacja zmienncyh do rozkładów podobnych do normalnych

Naszą kolejną próbą polepszyć wyniki naszych modeli została transformacja zmiennych do rozkładów podobnych do rozkładów normalnych, bo niektóre metody w uczeniu maszynowym (między innymi i PCA, użyta w niektórych przez nas pipelinach) zakładają, że zmienne pochodzą z rozkładów normlanych. Więc stworzyliśmy drugi PipelineBasic, który robi przekształcenia wszystkich zmiennych numerycznych do rozkładów podobnych do normalnych. Zastosowaliśmy PowerTransformer(method = 'box-cox') oraz PowerTransformer(method='yeo-johnson') w zależności od rozkładu początkowego zmiennej.

Po ponownym przeszukiwaniu odpowiednich hyperparametrów z użyciem nowego BasicPipeline dla modeli GradientBoostingClassifier oraz XGBoost (bo tam stosujemy PCA) wynki i na kros walidacji, i na zbiorze testowym miejscami były takie same, jak w tabeli 1, a miejsczmi nawet gorsze.

Zauważmy, że metoda Regresji Logistycznej i metoda Recursive Feature Elimination nie zakładają, że dane pochodzą z rozkładów normalnych. Więc nasz końcowy najlepszy model to Regresja Logistyczna, która jest wytrenowana na zbiorze treningowym bez transformacji do rozkładów podobnych do normalnych.

5 Model Top 1

Na podstawie wyników w tabeli 1, postanowiliśmy, że dla naszego datasetu Logistic Regression to najlepszy model. On ma najwyższy recall 1 oraz ma accuracy na dobrym poziomie.

	Accuracy	Percision 0	Recall 0	Percision 1	Recall 1
Cross-validation	0.7344	0.7535	0.9345	0.5833	0.2308
Testing	0.7438	0.7569	0.9478	0.6250	0.2222

Table 2: Metryki Logistic Regression

6 SMOTE and OverSampling

Jak już było mówiono na początku, nasze dane są bardzo niezbalansowane. Z tego powodu spróbowaliśmy zastosować metodę SMOTE oraz OverSampling dla Logistic Regression.

6.1 SMOTE

Po zastosowaniu SMOTE dla zbioru kros walidacyjnego (zbiór testowy pozostał bez zmian) otrzymaliśmy wyniki zapisane tabeli 2:

	Accuracy	Recall 1
Cross-validation	0.774	0.6179
Testing	0.70625	0.044

Table 3: wyniki po zastosowaniu SMOTE dla Logistic Regression

6.2 Over Sampling

	Accuracy	Recall 1
Cross-validation	0.627	0.70
Testing	0.64375	0.667

Table 4: wyniki po zastosowaniu OverSampling dla Logistic Regression

Jak widzimy, zastsowanie SMOTE i OverSampling w żadny sposób nie poprawiło wyników. Analogicznie próbowaliśmy dopasowywać parametr class_weight dla niektórych modeli, ale wartość "None" dawała najlepsze wyniki.

7 TPOT

Nie byliśmy za bardzo zadowoleni wynikami naszych modeli w stosunku do zrobionej pracy, bo we wszystkich naszych modelach, maksymalne wyniki accuracy są bardzo podobne i w okolicah 71-75%. Żeby jeszcze raz sprawdzić otrzymane przez nas wyniki, skorzystaliśmy z TPOTClassifier dla poszukiwania najlepszej modeli dla maksymalizacji Accuracy.

```
TPOTClassifier(generations=30, population_size=40, verbosity=2,
        random_state=42, scoring = 'accuracy')
```

Po działaniu tego kodu, przez TPOTClassifier został znaleziony najlepszy model. To był

```
ExtraTreesClassifier(
    GaussianNB(
        RobustScaler(input_matrix)),
        bootstrap=False,
        criterion=gini,
        max_features=0.35000000000000003,
        min_samples_leaf=18,
        min_samples_split=20,
        n_estimators=100)
```

Ten model pokazał Accuracy na zbiorze walidacyjnym 0.74375. Niestety na zbiorze testowym Accuracy już spadła do 0.725.

8 XAI

Do interpretacji najlepszej względem accuracy modeli zdecydowaliśmy wykorzystać *dalex*.

8.1 Ważność zmiennych

Z analizy zrozumieliśmy że dla LogisticRegression najważniejszą zmienną była R_DEBT_INCOME. Faktycznie takimi i byly nasze oczekiwania na podstawie 2.3

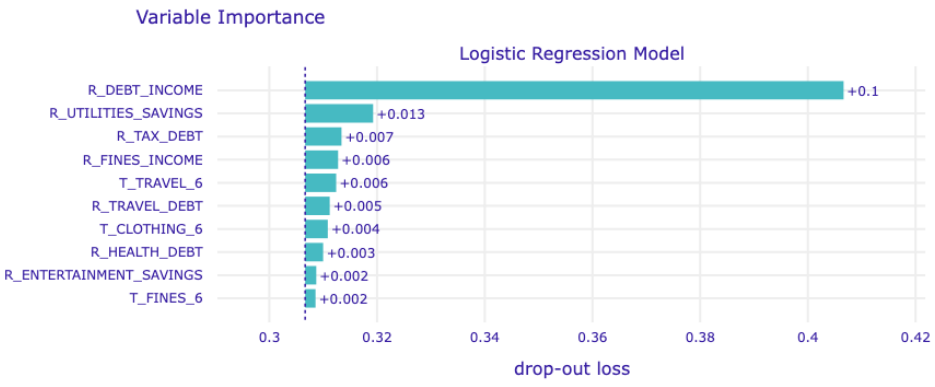

Figure 5: Ważności zmiennych

8.2 Ocena wydajności modelu

Na wykresie niżej można zobaczyć porównanie wybranej modeli z Random Classifier'om, widać ż wydajność jest wyżej.

Figure 6: Wydajność modelu

9 Zastosowanie biznesowe

Z naszego punktu widzenia bank szuka model, który skutecznie optymalizuje stosunek prawdziwie pozytywnych wyników do fałszywie pozytywnych, przyczynia się do dwóch kluczowych aspektów działalności banku: zwiększenia dochodowości oraz zmniejszenia ryzyka kredytowego.

- Zmniejszenie ryzyka kredytowego: Poprzez zwiększenie dokładności w identyfikowaniu klientów, którzy mogą nie być w stanie spłacić kredytu (True Positive), bank minimalizuje potencjalne straty wynikające z niewypłacalności. Mniejsza liczba fałszywie pozytywnych wyników oznacza, że mniej niewypłacalnych klientów otrzymuje finansowanie, co bezpośrednio przekłada się na stabilność finansową banku.
- Zwiększenie dochodowości: Udoskonalając zdolność do rozróżniania między niewypłacalnymi a wypłacalnymi klientami, bank może bardziej efektywnie kierować swoje produkty finansowe do osób, które najprawdopodobniej będą w stanie terminowo spłacić swoje zobowiązania. To nie tylko zwiększa bazę kredytową banku, ale także pozwala na bardziej celowane i efektywne kampanie marketingowe, które są skierowane do klientów o potwierdzonej zdolności kredytowej.
- Optymalizacja ofert produktowych: Analizując zmienne, które najbardziej wpływają na predykcje modelu (np. stosunek długów do dochodu), bank może dostosowywać swoje produkty i oferty do specyficznych potrzeb różnych segmentów klientów. Dzięki temu, oferty są bardziej atrakcyjne i lepiej dopasowane, co z kolei może przyciągnąć nowych klientów i zwiększyć lojalność obecnych.

Zbudowany nami model pozwala bankowi zoptymalizować straty z niewypłacalności kredytu, z powodu tego, że chociaż mamy niski recall defaultu klientu, detektujemy $\approx 95\%$ klientów wypłacających kredyt.

10 Co jeżeli będziemy traktować CREDIT_SCORE jako zmienną, a nie potecjalny target

Było nam też ciekawo, jak się zmienią wyniki naszego najlepszego modelu, jeżeli będziemy traktować CREDIT_SCORE jako jeszcze jedną zmienną. Na podstawie wykresu na Fig.7 zauważylśmy, że DEFAULT i CREDIT_SCORE to różne rzeczy.

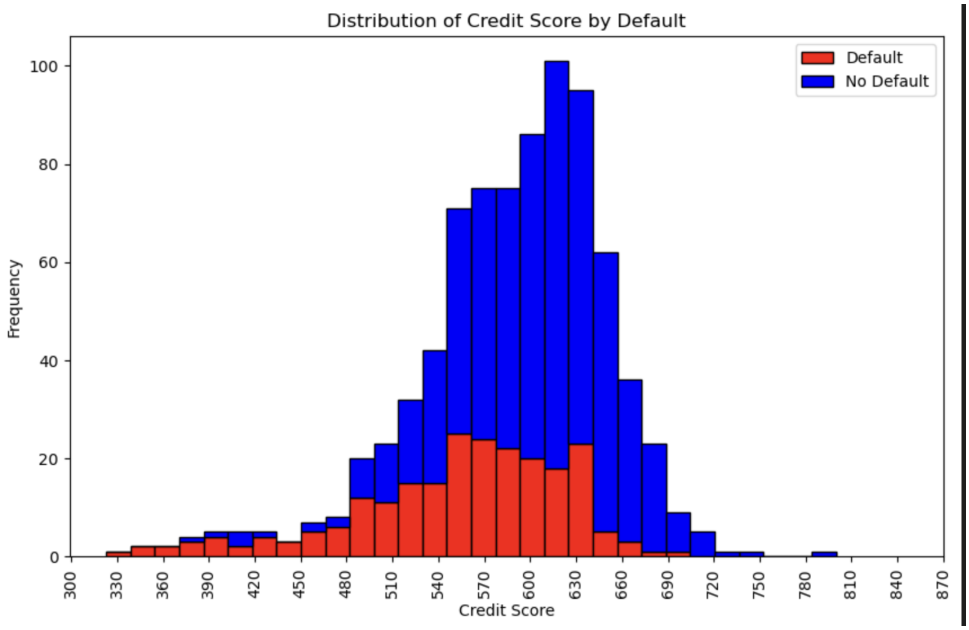

Figure 7: DEFAULT a CREDIT_SCORE

Wyniki modelu Logistic Regression można zobaczyć w tabeli 4:

	Accuracy	Recall 1
Cross-validation	0.74375	0.291
Testing	0.74375	0.244

Table 5: wyniki modelu Logistic Regression jeżeli CREDIT_SCORE to kolejna zmienna, a nie target

11 Przyczyny takich wyników

1. Mała ilość danych. Na początku mieliśmy 1000 obserwacji, 200 musieliśmy oddać zespołowi walidacyjnemu. Więc podczas cross walidacji było wykorzystane tylko 640 obserwacji, bo 160 były w zbiorze testowym. To jest za mało, przypominając, że mamy więcej niż 80 kolumn.
2. Nasze dane to sztucznie wygenerowane dane.

References

1. Credit score dataset: https://www.kaggle.com/datasets/conorsully1/credit-score