Modele klasyfikacyjne uczenia maszynowego wykorzystane do minimalizowania ryzyka przyznawania kart kredytowych – analiza porównawcza modeli

Opracował: Dominik Bałon

1. Cel pracy

Celem niniejszej pracy była budowa i ocena modeli uczenia maszynowego mających na celu prognozowanie decyzji o przyznaniu karty kredytowej na podstawie cech klientów. Zadanie to oparte było na zbiorze danych "Credit Card Approval", który zawiera różnorodne informacje dotyczące osób ubiegających się o kredyt, takie jak wiek, poziom dochodów, status cywilny, liczba dzieci oraz więcej.

Przed przystąpieniem do budowy modeli, zbiór danych wymagał przeprowadzenia szeregu wstępnych działań przygotowawczych, takich jak oczyszczenie danych z niekompletnych wpisów, analiza brakujących wartości oraz usunięcie potencjalnych nieprawidłowości. W ramach analizy eksploracyjnej danych przeprowadzono podstawowe statystyki opisowe, takie jak średnie, mediany, odchylenia standardowe, a także wizualizacje pomagające zrozumieć rozkład poszczególnych zmiennych oraz ich wzajemne zależności. Istotnym krokiem było również zbadanie wpływu poszczególnych cech na zmienną prognozowaną – decyzję o przyznaniu karty kredytowej. Wszystkie te działania miały na celu przygotowanie zbioru danych do budowania na nim modelu, który pozwoli uzyskać wysokie wyniki klasyfikacji.

Po przygotowaniu danych, zbudowane zostały modele klasyfikacyjne, które miały na celu przewidywanie, czy dany klient otrzyma kartę kredytową, na podstawie zgromadzonych cech. Do analizy wybrano popularne algorytmy uczenia maszynowego, w tym drzewo decyzyjne oraz maszynę wektorów nośnych (SVM), a także przeprowadzono ich kalibrację, zmieniając ich hiperparametry w celu optymalizacji wyników.

W ramach pracy zastosowano również techniki zwiększania liczby próbek w klasie mniejszościowej, takie jak oversampling, które miały na celu poprawienie wydajności modelu w przypadku niezbalansowanego zbioru danych. Metoda ta pozwoliła na równoważenie liczby przykładów w obu klasach, co było istotne dla analizowanego zbioru danych, gdzie występowała duża dysproporcja pomiędzy dwoma klasami.

Dodatkowo, aby ocenić model w sposób bardziej rzetelny i uniknąć przeuczenia, zastosowano walidację krzyżową (cross-validation). Dzięki temu możliwe było uzyskanie stabilniejszych wyników i lepszej oceny modelu, zważając na jego generalizację na różnych podzbiorach zbioru treningowego.

Podjęte działania miały na celu nie tylko zbudowanie dokładnych modeli prognozujących decyzje kredytowe, ale również zapoznanie się z metodami przetwarzania danych, analizowania ich struktury oraz oceny jakości działania modeli.

2. Eksploracyjna analiza danych (EDA)

2.1. Opis i wstępna analiza danych

Pierwotnie zbiór danych dotyczących przyznawania kart kredytowych składał się z dwóch odrębnych zbiorów danych ze wspólnym „id”. W jednym były cechy osób, ubiegających się o kartę kredytową, które poniżej zostaną opisane, a w drugim decyzja w postaci 0 i 1, liczby te odpowiadały kolejno za pozytywną decyzję wydania karty kredytowej i odrzucenie wniosku o kartę kredytową. Zbiory te zostały połączone z użyciem odpowiedniej funkcji pandas.

```
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Ind_ID           1548 non-null   int64
 1   GENDER           1541 non-null   object
 2   Car_Owner        1548 non-null   object
 3   Propert_Owner    1548 non-null   object
 4   CHILDREN         1548 non-null   int64
 5   Annual_income    1525 non-null   float64
 6   Type_Income      1548 non-null   object
 7   EDUCATION        1548 non-null   object
 8   Marital_status   1548 non-null   object
 9   Housing_type     1548 non-null   object
 10  Birthday_count   1526 non-null   float64
 11  Employed_days    1548 non-null   int64
 12  Mobile_phone     1548 non-null   int64
 13  Work_Phone       1548 non-null   int64
 14  Phone            1548 non-null   int64
 15  EMAIL_ID         1548 non-null   int64
 16  Type_Occupation  1060 non-null   object
 17  Family_Members   1548 non-null   int64
 18  label            1548 non-null   int64
```

Rys. 1 Objaśnienie typów zmiennych w zbiorze danym.

2.2. Dalsza analiza oraz przygotowywanie danych

2.2.1. Oczyszczenie zbioru z braków

Zbiór zawierał określoną ilość brakujących danych, z racji na dużą liczbę wierszy zdecydowano się na usunięcie wierszy z brakującymi danymi. Po tej operacji zbiór danych zredukował się w sposób przedstawiony na poniższym rysunku.

```
Kształt pierwotnego zbioru danych: (1548, 19)
Kształt zbioru danych po obróbce: (907, 19)
```

Rys. 2 Kształt zbioru danych po usunięciu braków.

2.2.2. Modyfikacja typów zmiennych

Istniejąca forma zbioru danych nie mogłaby być wykorzystana do budowy modeli uczenia maszynowego. Wiele kolumn przechowywało wartości typu object (najczęściej string). Konieczne było zatem zastosowanie odpowiednich technik, aby przygotować zbiór danych do budowy modeli. W tym dokumencie przedstawię jak tylko kilka operacji, które zostały zastosowane, aby przerobić dane na typ numeryczny.

Kolumna „GENDER" zawierała zmienne w postaci [„Male", „Female"], więc zastosowano tutaj najprostszą metodę mapowania [{„Male": 1, „Female": 0}]. Także po przekształceniu uzyskano kolumnę ze zmienną numeryczną w postaci [0, 1]. Podobna sytuacja miała miejsce z kolumną Car_Owner, Propert_Owner oraz Marital_status.

Inna metoda została zastosowana do przerobienia kolumny „Housing_type". Pierwotnie posiadała ona zmienne w postaci string przyjmujące następujące wartości: ['House / apartment', 'With parents', 'Municipal apartment', 'Rented apartment', 'Office apartment', 'Co-op apartment']. Wykorzystano do tego metodę „OrdinalEncoder", która umożliwia przypisanie unikalnych wartości liczbowych dla poszczególnych kategorii w danych, co pozwala na ich dalszą analizę i wykorzystanie budowie modeli. Oceniłem, że zmienne posiadały określoną hierarchię, przedstawioną w powyższej tablicy. Po zastosowaniu tej metody kolumna przyjęła wartości kolejno z przedziału od 0 do 5. Ta sama metoda została wykorzystana dla kolumny „Type_occupation".

Kolumny „EDUCATION" oraz „Type_income" zostały zamienione na zmienne numeryczne przy użyciu metody „one-hot-encoding".

Zastosowanie tych wszystkich opisanych modyfikacji doprowadziło do powstania zbioru danych posiadającego wszystkie kolumny w postaci numerycznej. Aktualny wygląd zbioru danych z typami zmiennych przedstawia rysunek poniżej.

#	Column	Non-Null Count	Dtype
0	Gender	907 non-null	int64
1	Car_Owner	907 non-null	int64
2	Propert_Owner	907 non-null	int64
3	Children	907 non-null	int64
4	Annual_income	907 non-null	float64
5	Marital_status	907 non-null	int64
6	Age	907 non-null	float64
7	Employed_days	907 non-null	int64
8	Work_Phone	907 non-null	int64
9	Family_Members	907 non-null	int64
10	Type_Income_Pensioner	907 non-null	bool
11	Type_Income_State servant	907 non-null	bool
12	Type_Income_Working	907 non-null	bool
13	Education_Higher education	907 non-null	bool
14	Education_Lower secondary	907 non-null	bool
15	Education_Secondary / secondary special	907 non-null	bool
16	Housing_type_encoded	907 non-null	float64
17	Type_Occupation_encoded	907 non-null	float64
18	CreditCard_decision	907 non-null	int64

Rys. 3 Typy zmiennych po modyfikacjach.

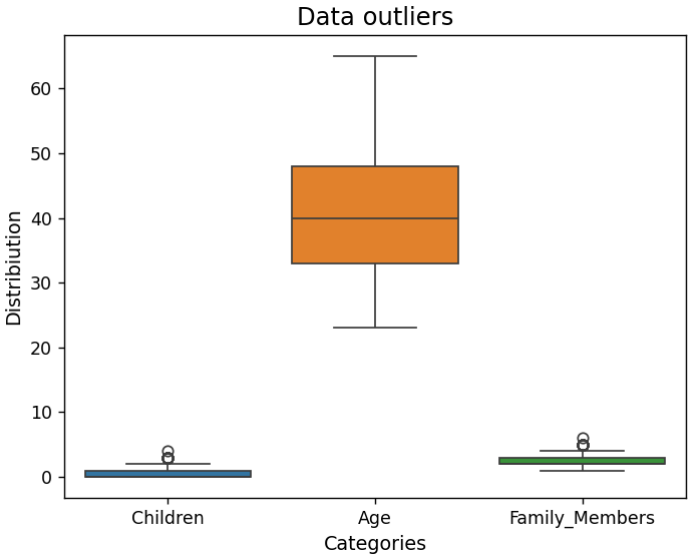
2.2.3. Sprawdzenie istnienia wartości odstających

Kolejnym krokiem w celu przygotowaniu zbioru do budowy modeli, było sprawdzenie, czy istnieją wartości odstające w poszczególnych kolumnach liczbowych.

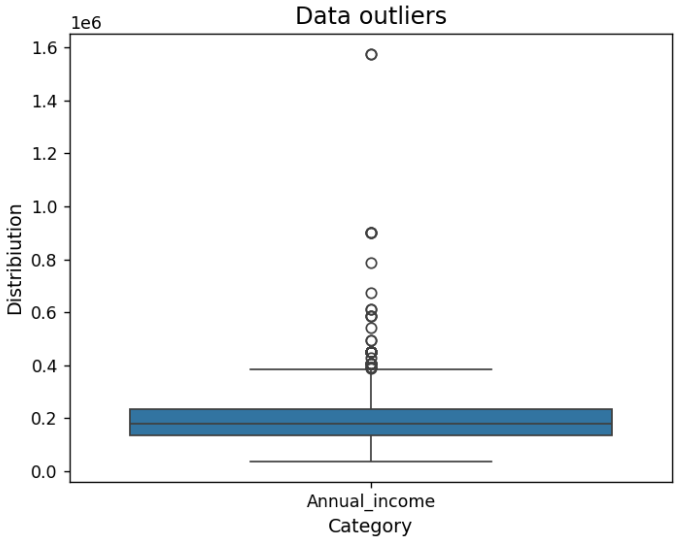
	Gender	Car_Owner	Propert_Owner	Children	Annual_income	Marital_status	Age	Employed_days
count	907.000000	907.000000	907.000000	907.000000	9.070000e+02	907.000000	907.000000	907.000000
mean	0.433297	0.439912	0.646086	0.503859	2.009606e+05	0.837927	40.663727	2695.826902
std	0.495804	0.496650	0.478447	0.746394	1.245861e+05	0.368721	9.472188	2359.148076
min	0.000000	0.000000	0.000000	0.000000	3.600000e+04	0.000000	23.000000	73.000000
25%	0.000000	0.000000	0.000000	0.000000	1.350000e+05	1.000000	33.000000	996.000000
50%	0.000000	0.000000	1.000000	0.000000	1.800000e+05	1.000000	40.000000	1994.000000
75%	1.000000	1.000000	1.000000	1.000000	2.340000e+05	1.000000	48.000000	3648.500000
max	1.000000	1.000000	1.000000	4.000000	1.575000e+06	1.000000	65.000000	14887.000000

Rys. 4 Podstawowe statystyki opisowe.

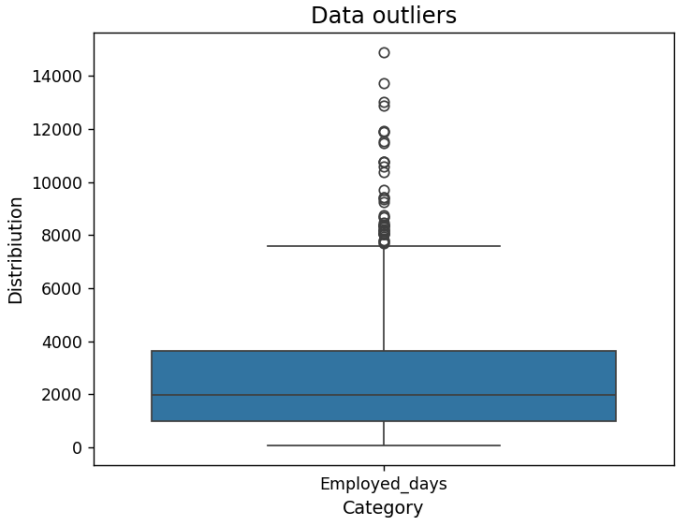
Statystyki opisowe pokazały wstępnie, w których kolumnach możemy spodziewać się wartości odstających. Szczegółowo pokazano to na wykresach pudełkowych.



Rys. 5 Wykres pudełkowy wybranych zmiennych.



Rys. 6 Wykres pudełkowy zmiennej „Annual_income”.

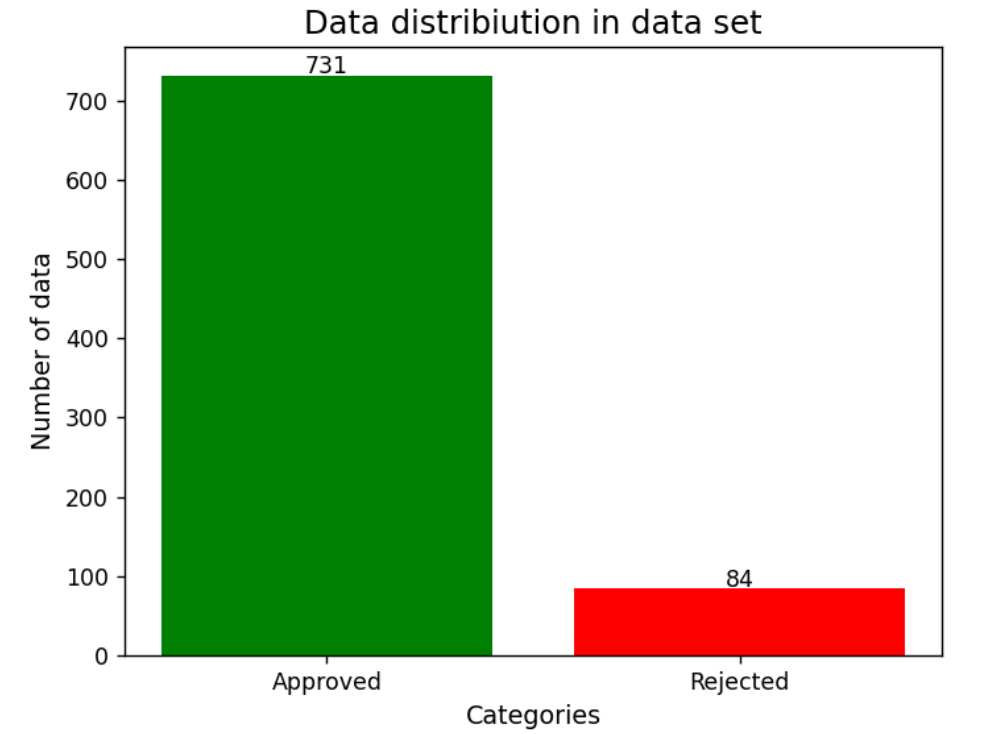


Rys. 7 Wykres pudełkowy zmiennej „Employed_days”.

Analizując przedstawione wykresy pudełkowe można stwierdzić, że istnieją wartości odstające, które mogą mieć wpływ na jakość budowanego modelu. Zjawisko to zauważono głównie dla kolumny „Annual_income" oraz „Employed_days". Wartości odstające zostały usunięte w ilości 92.

2.2.4. Zbalansowanie danych w zbiorze

Zbiór danych, można określić jako znacznie niezbalansowany. Liczba próbek należących do klasy 0 jest znacznie większa niż dla klasy 1. Walkę z tym problemem podjęto w następnych krokach.

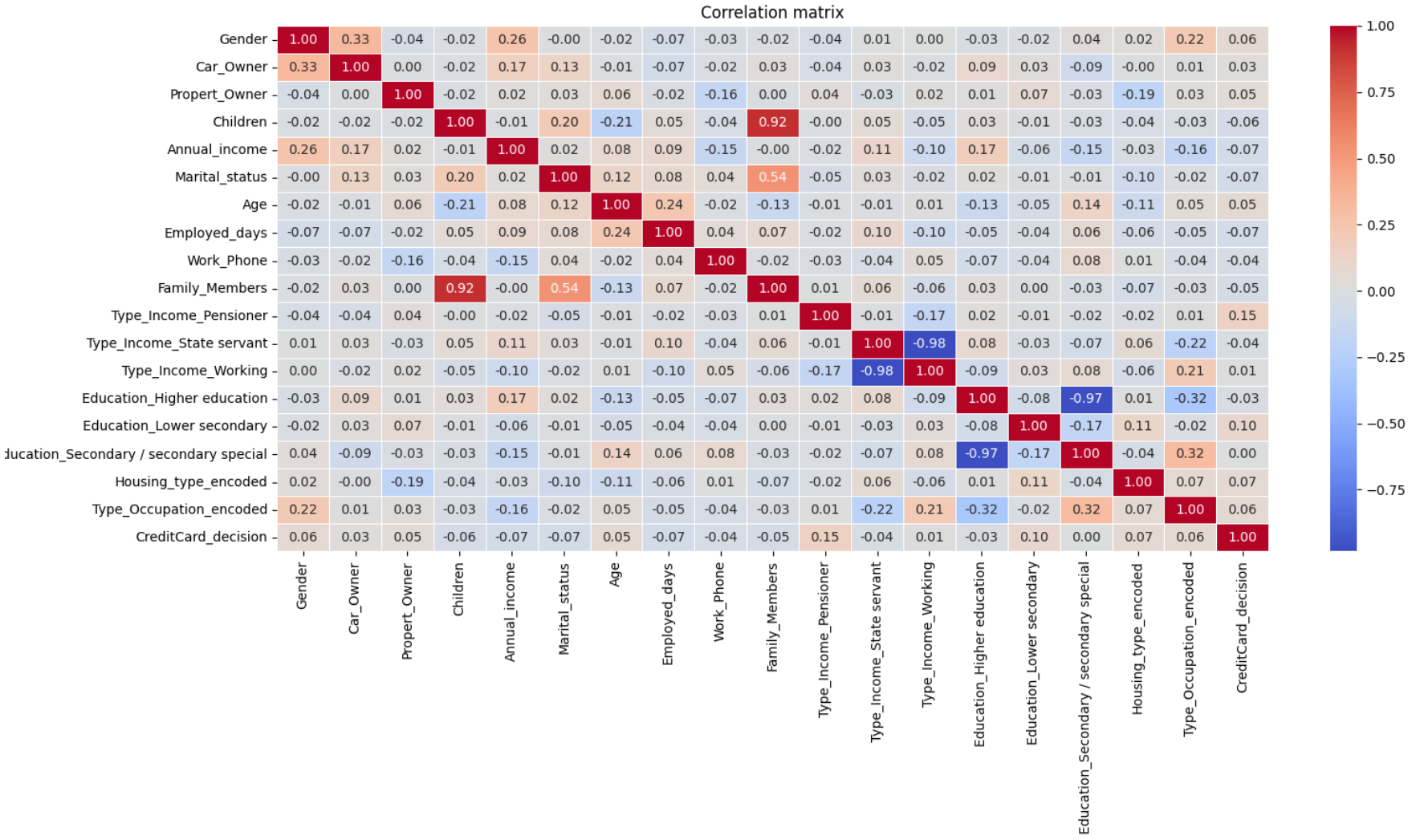


Rys. 8 Zbalansowanie klas w zbiorze.

2.2.5. Macierz korelacji dla istniejących zmiennych

Przedstawiona poniżej macierz korelacji pokazuje, jak poszczególne kolumny oddziałują na inne kolumny. Można odczytać, że kolumny takiej jak „Family members" i „Children" są ze sobą silnie skorelowane, bowiem współczynnik korelacji wynosi blisko 1. Oznacza to, że wraz ze wzrostem liczby ludzi w rodzinie, większa jest też liczba dzieci, co jest logiczne. Z tego powodu, aby ograniczyć zjawisko redundancji usunięto kolumnę „Family Members".

Dodatkowo można zauważyć wysoką korelację pomiędzy zmiennymi uzyskanymi za pomocą „one-hot encoding". Natomiast zdecydowano się na pozostawienie tych zmiennych do budowy modelu.



Rys. 9 Macierz korelacji.

3. Podział zbiorów i zastosowanie metod balansowania zbioru.

Zanim przystąpimy do budowy modelu, należy dokonać niezbędnego podziału na zbiór testowy i treningowy. Ze względu na duże niezbalansowanie danych na zbiorze treningowym zostanie zastosowana metoda oversamplingu.

3.1. Podział na zbiór uczący i testowy

Podziału dokonano przy zastosowaniu najprostszej metody „train_test_split". Do zbioru testowego założono 25 procent danych.

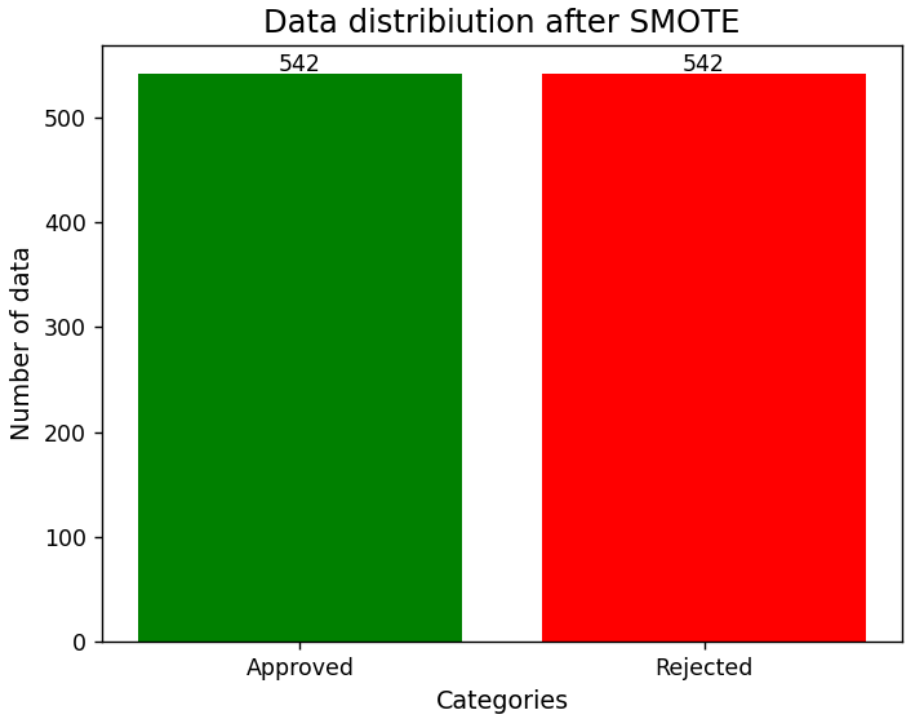
```
X = credit_db_encoded_no_outliers.drop('CreditCard_decision', axis=1)
Y = credit_db_encoded_no_outliers['CreditCard_decision']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=40)
```

Rys. 10 Podział zbioru na testowy i treningowy.

3.2. Zastosowanie metod oversamplingu

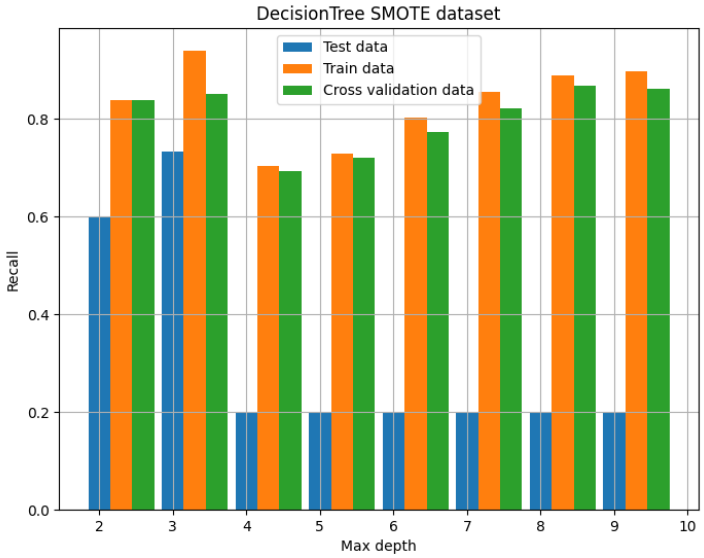
Niezbalansowany zbiór danych wymaga zastosowania metod oversamplingu lub undersamplingu, aby wyrównać proporcję w przynależności do danych klas i wytrenować lepszy model. Wykorzystano do tego dwie metody ADASYN oraz SMOTE. Metoda ADASYN ze względu na swoją charakterystykę dawała gorsze wyniki oceny budowanych modeli. W związku z tym zbiory danych zostały zwiększone o nowe próbki przy użyciu metody SMOTE.



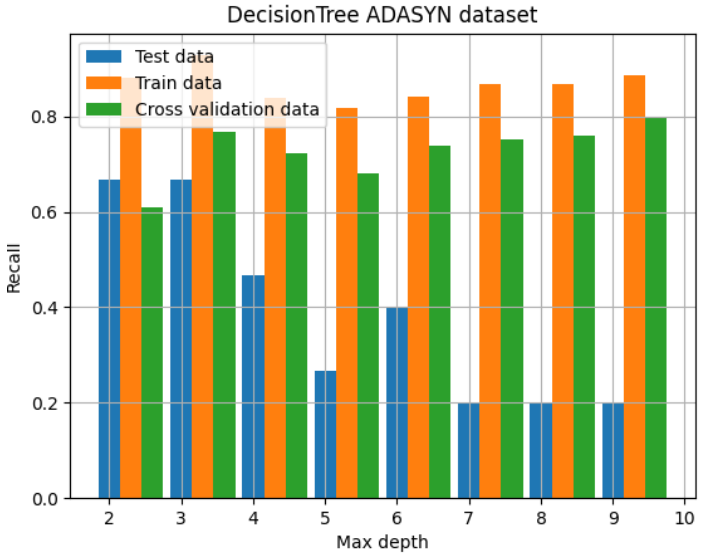
Rys. 11 Zbalansowanie danych po oversamplingu.

3.3. Porównanie wyników drzewa decyzyjnego po ADASYN i SMOTE

Dokonano porównania wyników czułości drzewa decyzyjnego opartego na danych uzyskanych po ADASYN i SMOTE. Z poniższych wykresów można zauważyć, że metoda oversamplingu SMOTE dla naszych danych daje bardziej zadowalające rezultaty.



Rys. 12 Czułość drzewa decyzyjnego na danych SMOTE



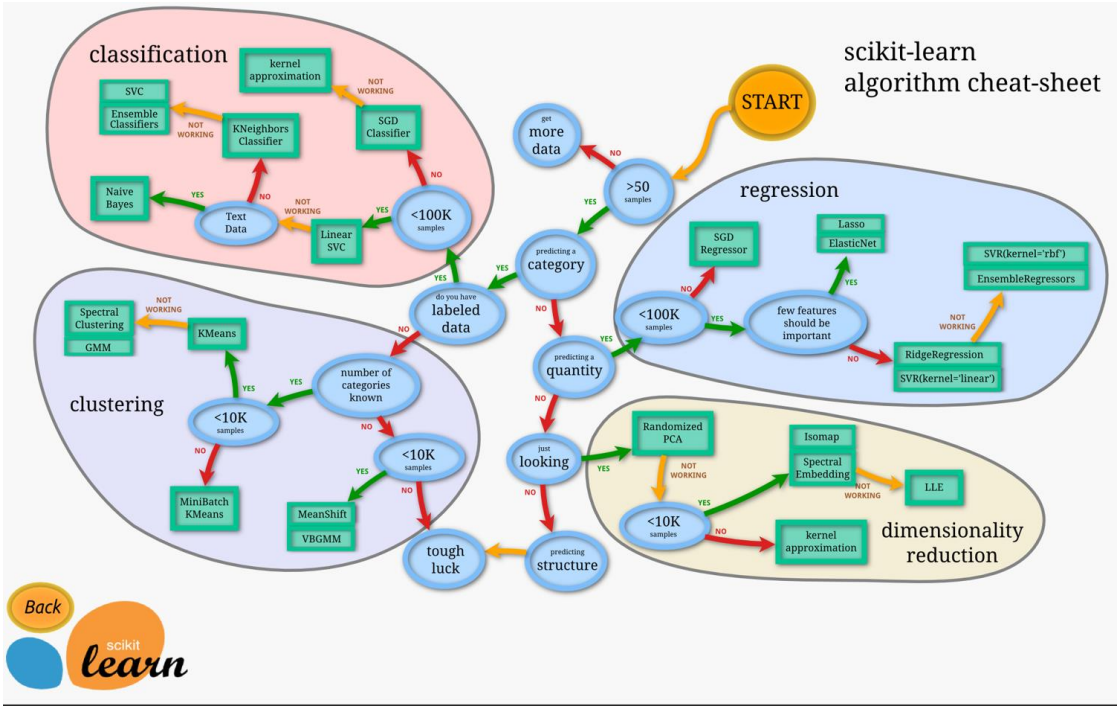
Rys. 12 Czułość drzewa decyzyjnego na danych SMOTE

4. Budowa modeli uczenia maszynowego

Przygotowano 4 modele uczenia maszynowego takie jak: SVM, Decision Tree i Random Forest. Przy ocenie modelu zwrócono szczególną uwagę na czułość, ponieważ klasą pozytywną było odrzucenie wniosku o przydzielenie karty kredytowej. Zależy nam zatem najbardziej na tym, żeby karta kredytowa nie była wydana osobie, której nie powinna być wydana. Pozwoli to uchronić instytucję wydającą kartę przed potencjalnym brakiem spłacenia zadłużenia. Czułość jest miarą modelu pokazującą jaki procent klasy pozytywnej został prawidłowo rozpoznany przez model.

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$$

Rys. 13 Wzór na czułość modelu.

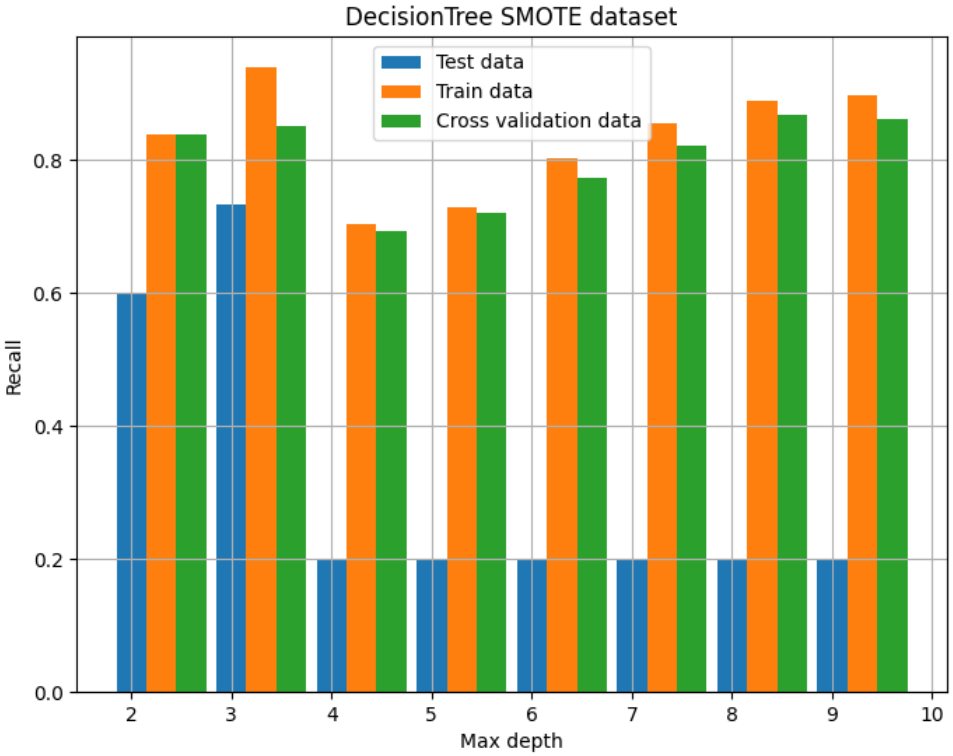


Rys. 14 Algorytm wyboru odpowiednich metod uczenia maszynowego.

4.1. Drzewo decyzyjne

Pierwszym budowanym modelem było drzewo decyzyjne, w dalszym etapie pracy będę wykorzystywał dane uzyskane dzięki oversamplingu metodą SMOTE.

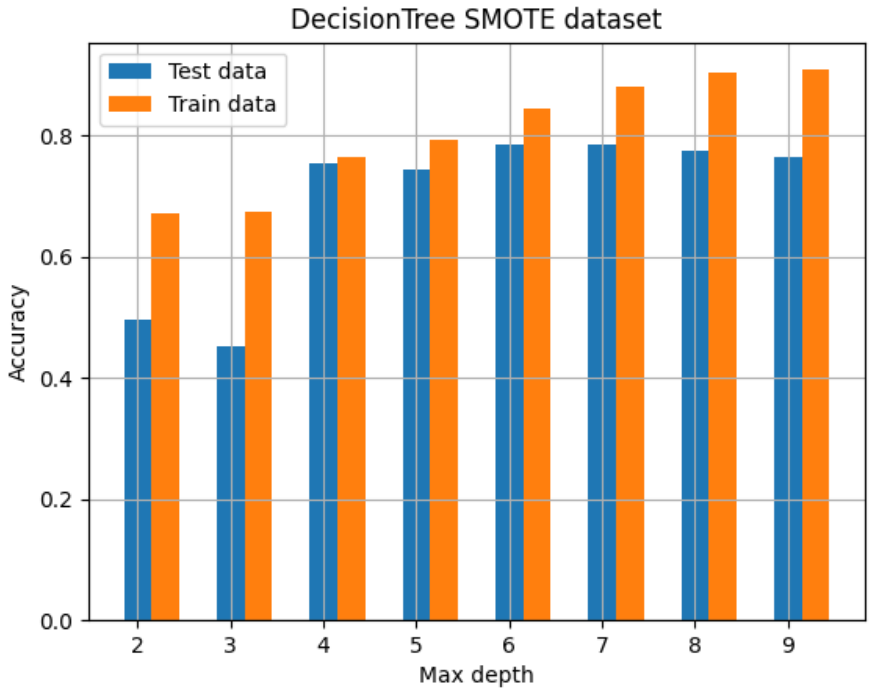
4.1.1. Miara czułości dla poszczególnych wartości hiperparametru max_depth



Rys. 15 Porównanie recall w zależności od hiperparametru max_depth.

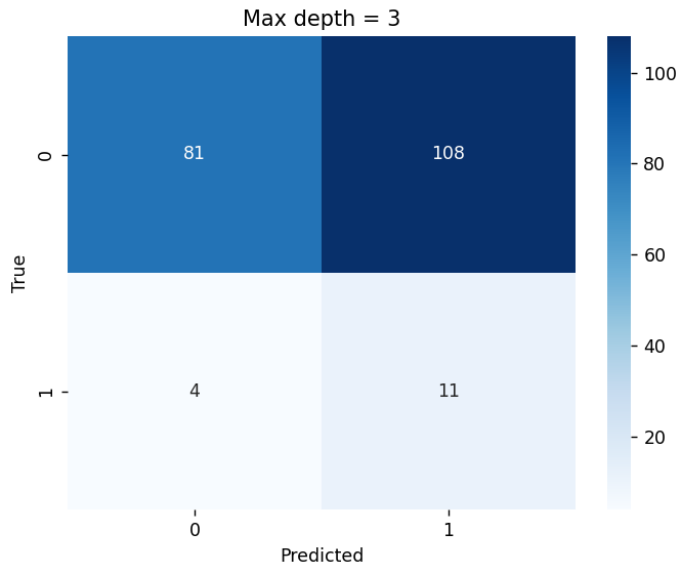
Można zauważyć, że recall najwyższy jest dla głębokości drzewa równego 3. Sytuacja ta ma miejsce zarówno dla zbioru testowego, jak i dla przeprowadzonej walidacji krzyżowej. Należy zauważyć, że recall jest szczególnie ważny ponieważ, w zbiorze testowym znajduje się mniejsza liczba danych należących do klasy 1.

4.1.2. Miara dokładności dla drzewa decyzyjnego



Rys. 16 Porównanie dokładności w zależności od max_depth.

Analizując powyższy wykres widać, że model nie radzi sobie dobrze z rozpoznawaniem klasy negatywnej (przyznanie karty kredytowej). Model dla większej wartości max_depth klasyfikuje dobrze klasę negatywną, natomiast nie rozpoznaje klasy mniejszościowej. Miara dokładności ze względu na swoją specyfikę obliczania będzie miała wysokie wartości, ponieważ do klasy 1 należy mała część danych, więc model jeśli wszystko zaklasyfikuje jako 0, to dokładność będzie wysoka, co będzie mylne dla naszego problemu.



Rys. 17 Macierz błędów dla drzewa decyzyjnego i max_depth = 3.

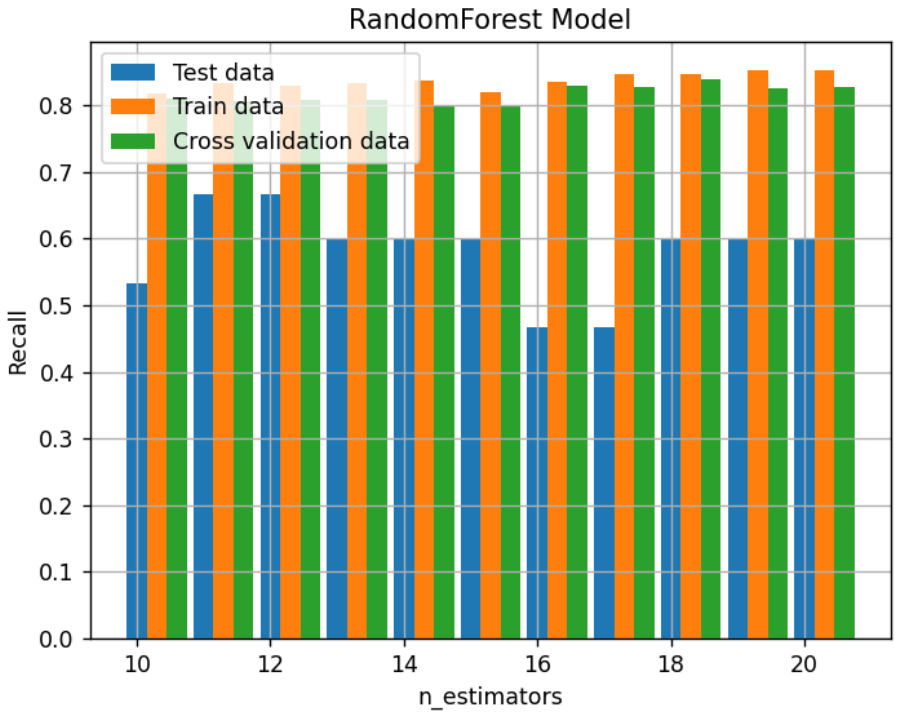
4.1.3. Podsumowanie uzyskanego modelu

Uzyskany model nie może być uważany za najlepszy do przyjęcia. Odnosząc model do realnych warunków, mamy sytuację, gdzie odrzucalibyśmy wnioski o karty kredytowe ludziom, którzy nie powinni tych kart otrzymać w zadowalającym stopniu, natomiast problem pojawia się, gdy chcemy przyznać tę kartę osobie, która na nią zasługuje, wtedy model również w większości przypadków klasyfikowałby taką osobę jako niezdolną do posiadania karty kredytowej. Model jest słaby.

4.2. Las losowy

Metodą, która w większości przypadków daje lepsze wyniki jest pochodna poprzedniej metody, czyli las losowy.

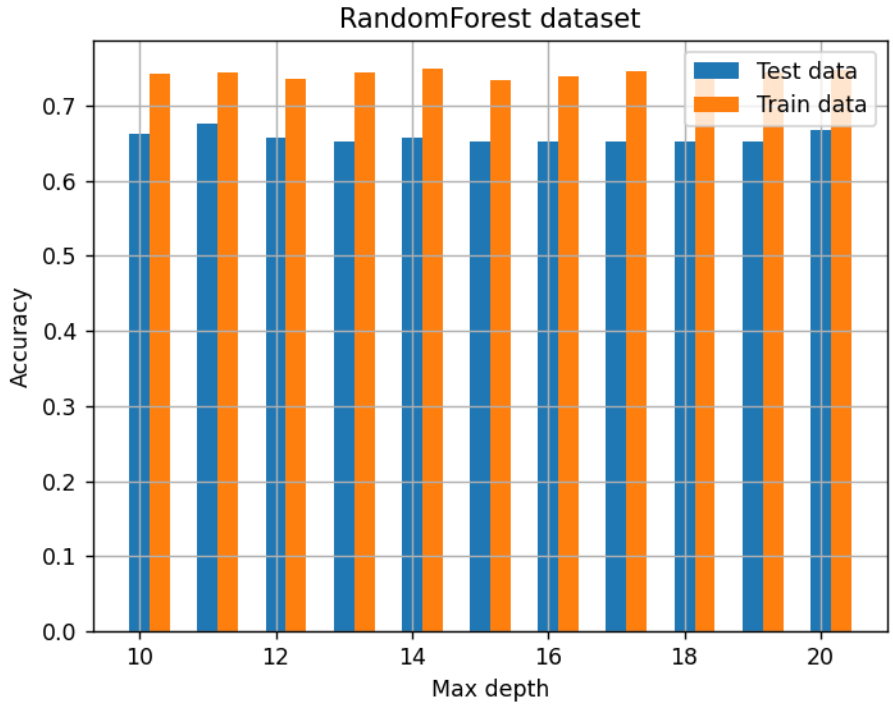
4.2.1. Miara czułości dla lasu losowego w zależności od n_estimators.



Rys. 18 Czułość lasu losowego.

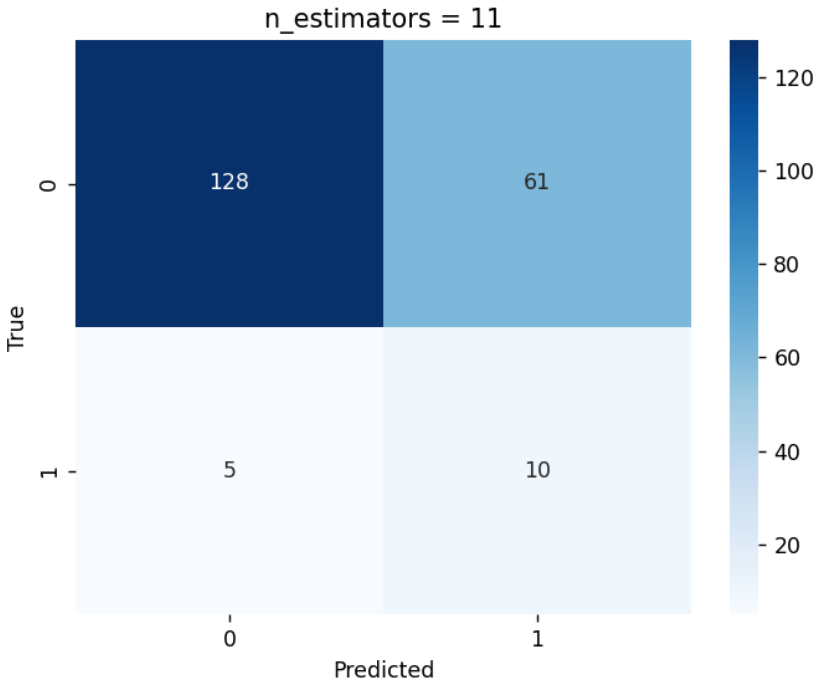
Las losowy osiąga najlepszy wynik miary czułości dla hiperparametru n_estimators = 11. Z otrzymanego wykresu można zauważyć, że potem wraz ze zwiększaniem tego parametru model ma tendencję do przeuczania, osiągając wysokie wyniki dla zbioru treningowego.

4.2.2. Miara dokładności dla lasu losowego



Rys. 18 Dokładność lasu losowego.

Dokładność lasu losowego wypada najlepiej dla n_estimators równego 11. Model radzi relatywnie dobrze, dokonuje prawidłowej klasyfikacji na poziomie 68%, biorąc pod uwagę dużą dysproporcję w klasach zbioru testowego.



Rys. 19 Macierz błędów dla lasu losowego.

4.2.3. Podsumowanie uzyskanego modelu

Uzyskany model w miarę zadowalający sposób rozróżnia osoby, które powinny dostać kartę kredytową, od osób, które tej karty nie powinny dostać. Patrząc na zbiór walidacji krzyżowej, model daje bardzo dobre wyniki, natomiast dla zbioru testowego już nieco gorsze. Jest to spowodowane tym, że walidacja krzyżowa działa na zbiorze po oversamplingu, dlatego też może model testuje się na danych stworzonych poprzez tę metodę.

4.3. SVM model

Kolejnym budowanym modelem był model SVM. Maszyna wektorów nośnych wymaga ustandaryzowania danych, co będzie pierwszym krokiem budowy modelu.

4.3.1. Standaryzacja danych

SVM bazuje na odległościach w przestrzeni cech, dlatego wymagana jest standaryzacja danych w zbiorze. Kolumny takie jak np. „Annual_income” zawierają zmienne znacznie odbiegające od średnich w kolumnach binarnych 0 i 1.

Wykorzystano do tego metodę StandardScaller. Jest to metoda skalowania, która przekształca dane, aby miały średnią równą 0 i odchylenie standardowe równe 1. Osiąga się to poprzez odjęcie średniej wartości każdej cechy i podzielenie przez jej odchylenie standardowe, co sprawia, że dane stają się znormalizowane i porównywalne między różnymi cechami.

4.3.2. Dobór hiperparametrów

Hiperparametry dla SVM zostały dobrane za pomocą metody GridSearchCV.

```
param_grid = {
  'C' : [0.1, 1, 10],
  'gamma': [0.01, 0.1, 1],
  'kernel': ['linear', 'rbf', 'poly']
}
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
grid_search.fit(X_train_scaled,y_train_smote)
```

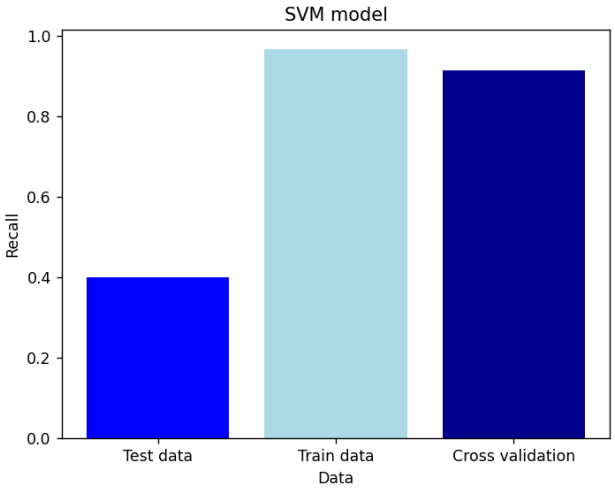
Rys. 20 Dobór hiperparametrów GridSearchCV.

Po zastosowaniu tego zapytania otrzymano wyniki jak na rysunku poniżej.

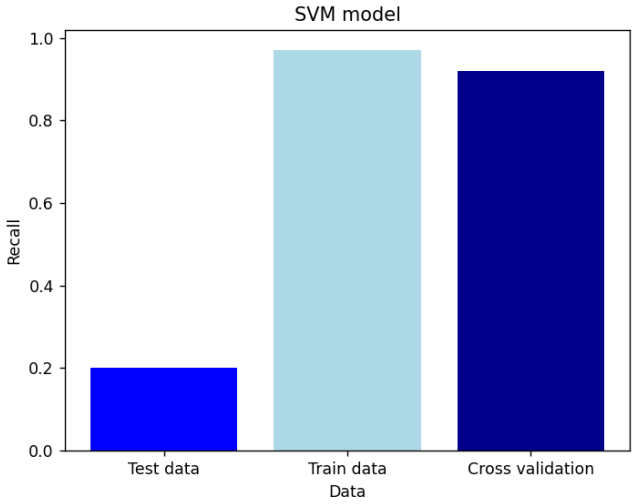
```
Best parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

Rys. 21 Najlepsze hiperparametry.

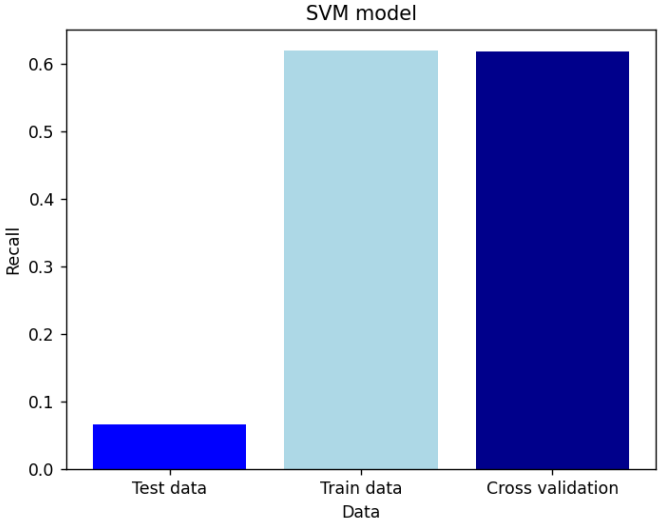
4.3.3. Porównanie czułości dla różnych jąder modelu SVM



Rys. 22 Recall dla jądra „Poly”.

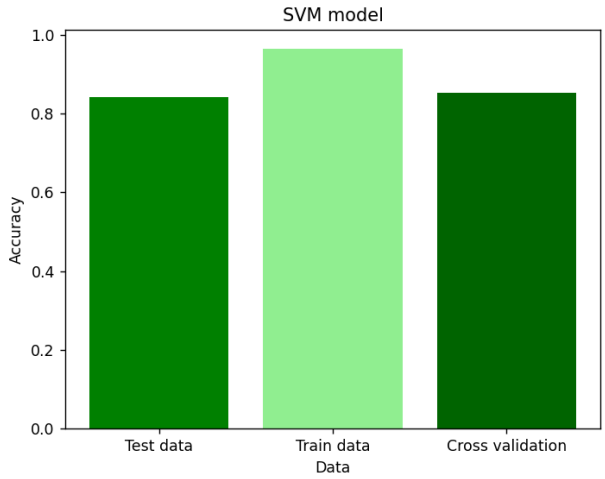


Rys. 23 Recall dla jądra „rbf”.

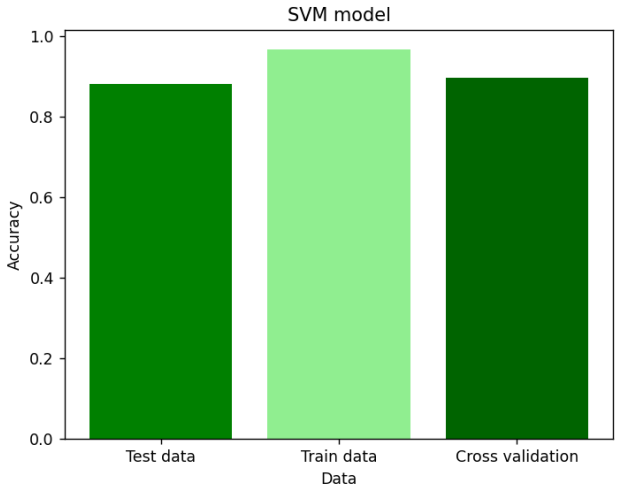


Rys. 24 Recall dla jądra „linear”.

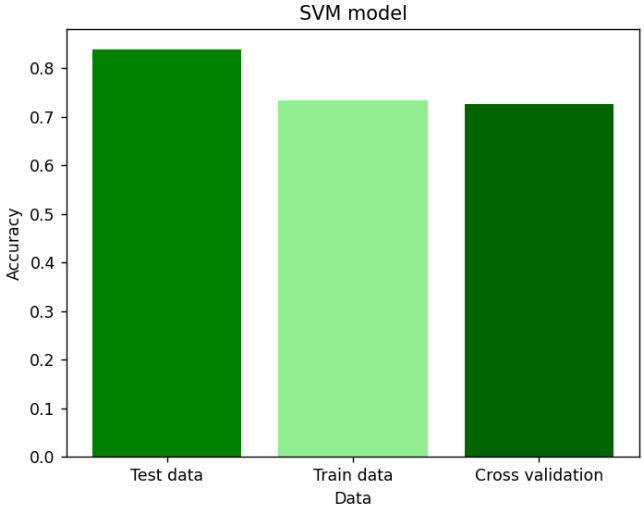
4.3.4. Porównanie dokładności dla różnych jąder modelu SVM



Rys. 25 Dokładność dla jądra „Poly”.



Rys. 27 Dokładność dla jądra „rbf”.



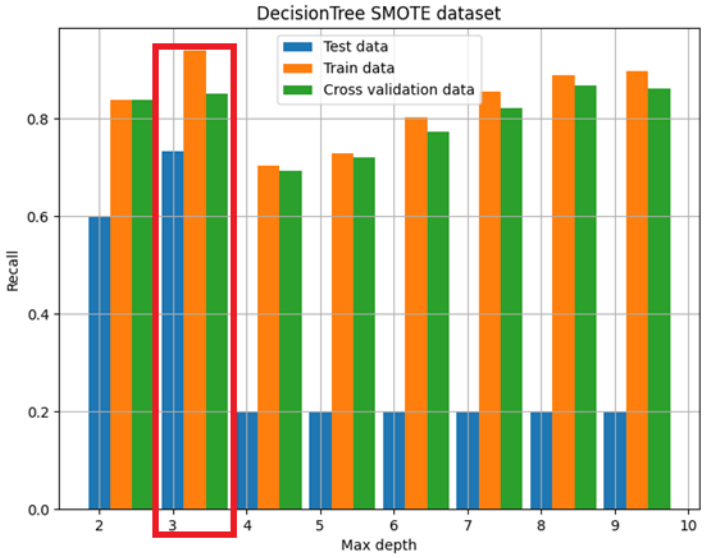
Rys. 26 Dokładność dla jądra „linear”.

4.3.5. Podsumowanie modelu SVM

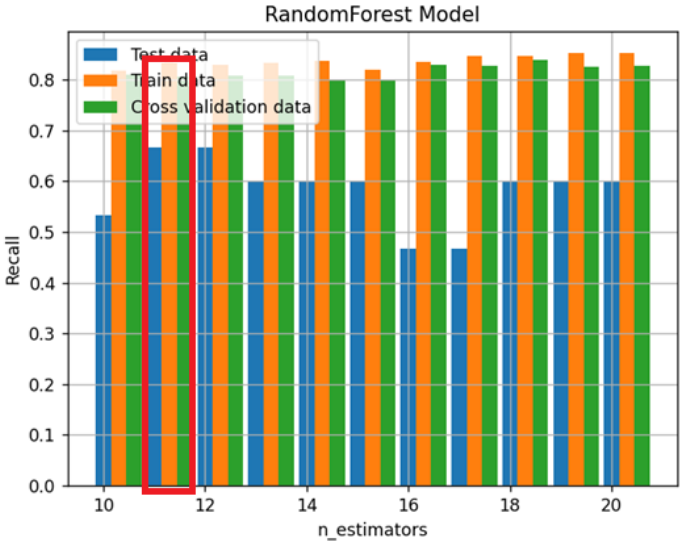
Uzyskany model uzyskał niskie wyniki dla miary czułości dla różnych jąder. Najwyższa wartość jest dla jądra „Poly" oraz „rbf". Zdecydowanie najsłabszy wynik dla czułości i dokładności model osiągnął dla jądra „linear", co może świadczyć o tym, że dane w zbiorze nie są ułożone w sposób liniowy.

4.4. Analiza porównawcza zbudowanych modeli

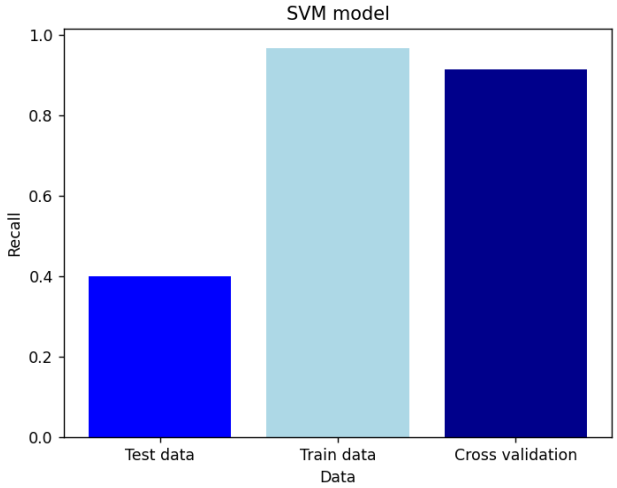
Analiza porównawcza obejmie miarę dokładności jaką jest czułość, ponieważ dokładność dla większości modeli była wysoka, ze względu na nierówny udział klas w zestawie testowym.



Rys. 27 Czułość dla drzewa decyzyjnego z największą wartością.



Rys. 28 Czułość lasu losowego z największą wartością.



Rys. 29 Czułość SVM z największą wartością.

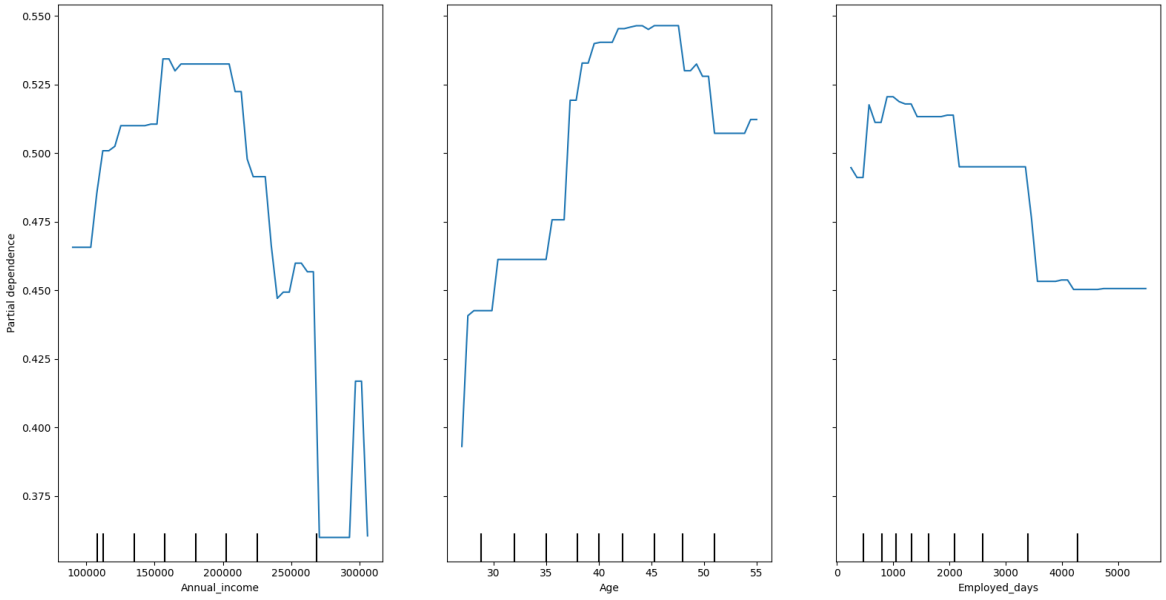
Miara czułości, która w tym przypadku jest kluczowa, odzwierciedla zdolność modeli do wykrywania pozytywnych przypadków, była najwyższa dla modelu drzewa decyzyjnego (0,7), co sugeruje, że model ten najlepiej radzi sobie z wykrywaniem pozytywnych przypadków w porównaniu do innych. Natomiast model ten musi zostać odrzucony ze względu na bardzo słaby wynik miary dokładności (0,42). Random Forest, mimo że osiągnął nieco niższą czułość (0,65), wykazał wyższą dokładność (0,65), co oznacza, że jest bardziej ogólny i skuteczny w poprawnym klasyfikowaniu zarówno pozytywnych, jak i negatywnych przypadków. Model SVM uzyskał najniższą czułość (0,4), co wskazuje na jego problemy z identyfikowaniem pozytywnych przykładów, mimo że osiągnął najwyższą dokładność (0,8). Oznacza to, że SVM mógłby mieć tendencję do ignorowania pozytywnych przypadków na korzyść większej liczby poprawnie klasyfikowanych przypadków ogólnych.

Model	Czułość	Dokładność	Uwagi
Decision Tree	0,70	0,42	Najlepsza czułość, słaba dokładność
Random Forest	0,65	0,65	Dobry kompromis pomiędzy miarami
SVM	0,40	0,80	Słaba czułość, wysoka dokładność

Tab. 1 Charakterystyka modeli ML

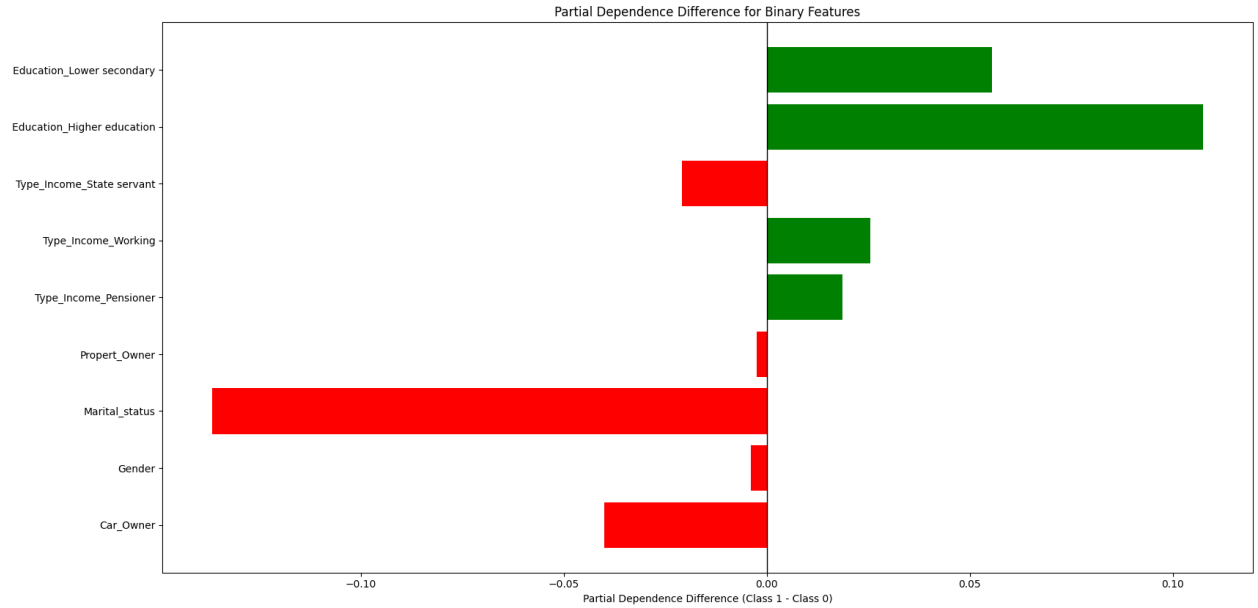
4.5. Analiza interpretowalności modeli

Profile ceteris paribus zostały wykorzystane do analizy wpływu poszczególnych zmiennych na zmienną objaśnianą. Umożliwiło to zbadanie zależności między wartościami zmiennych niezależnych a przewidywaniami modelu, przy założeniu, że pozostałe cechy pozostają stałe.



Rys. 30 Wykresy interpretowalności dla zmiennych liczbowych.

Z powyższych wykresów można odczytać, dla „Annual_income" wraz ze wzrostem tej wartości zmniejsza się szansa na odrzucenie wniosku o kartę kredytową (klasa 1), podobnie sytuacja ma się dla „Employed_days". Natomiast zastanawiający jest wykres środkowy, gdzie młody wiek zwiększa szansę na otrzymanie karty kredytowej.



Rys. 40 Wykresy interpretowalności dla zmiennych binarnych.

Analizując powyższy wykres można zauważyć wiele ciekawych rzeczy, z perspektywy badanego problemu. Powyższy rysunek należy interpretować w następujący sposób, jeśli dane cechy przyjmowały wartość 1, to przy kolumnach, gdzie znajdują się słupki czerwone, wpływały one na model zwiększając szansę na korzyść klasy 0 (wydanie karty kredytowej). Największy wpływ ma status cywilny osoby starającej się o taką kartę, dla osób żonatych, bądź zamężnych szansa na uzyskanie karty kredytowej była większa. Natomiast z drugiej strony kolorem zielonym są oznaczone cechy, które gdy przyjęły wartość 1 wpływały pozytywnie na model tzn. przydział klasy 1 (odrzucenie karty kredytowej). Zastanawiającym jest z perspektywy prawidłowości modelu, że wyższe wykształcenie miało pozytywny wpływ na odrzucenie wniosku o kartę, co w związku z rzeczywistością ma raczej małe pokrycie.

5. Podsumowanie

W pracy oceniono skuteczność trzech modeli uczenia maszynowego: drzewa decyzyjnego, lasu losowego oraz SVM, w prognozowaniu decyzji o przyznaniu karty kredytowej. Najważniejszym kryterium oceny była miara czułości, ponieważ zależało na prawidłowym identyfikowaniu przypadków odrzucenia wniosku o kartę (klasa 1). Analiza wyników pokazała, że model drzewa decyzyjnego osiągnął najwyższą czułość (0,7), ale jego niska dokładność (0,42) dyskwalifikuje go jako praktyczne rozwiązanie.

Najbardziej zbalansowanym modelem okazał się las losowy, który przy czułości 0,65 osiągnął dokładność (0,65), co świadczy o jego skuteczności w rozróżnianiu obu klas. Model SVM wykazał najwyższą dokładność (0,8), lecz jego czułość była znacząco niższa (0,4), co wskazuje na trudności w identyfikacji klasy mniejszościowej.

Dodatkowa analiza interpretowalności wykazała, że cechy takie jak wyższe dochody czy dłuższy czas zatrudnienia obniżają szansę na odrzucenie wniosku, natomiast wyższe wykształcenie, co zaskakujące, zwiększa tę szansę. Wyniki podkreślają potencjał modelu lasu losowego w tego typu zadaniach, ale jednocześnie wskazują na konieczność dalszych prac nad poprawą danych i optymalizacją modeli, ponieważ wszystkie zbudowane modele nie wykazały się dobrą skutecznością.

Problemem, przy budowie modeli, dla posiadanego zbioru danych okazała się bardzo duża dysproporcja pomiędzy próbkami w klasach. Po użyciu walidacji krzyżowej widać, że model dobrze radzi sobie podczas testowania go na zbiorze po oversamplingu, natomiast po testach na zbiorze niewidocznym przy budowaniu modelu, daje on przeciętne wyniki.