

Wpływ wrogiego uczenia maszynowego na modele estymacji wiarygodności kredytowej

Mateusz Kuchta

14 sierpnia 2023

Spis treści

| | |
|---|----|
| Wstęp | 4 |
| 1 Credit Scoring | 5 |
| 1.1 Credit Scoring - definicja i cechy | 5 |
| 1.2 Wyznaczanie oceny punktowej | 9 |
| 1.3 Score kredytowy w erze Big Data | 14 |
| 1.4 Metody Uczenia Maszynowego wykorzystywane w dziedzinie Credit Scoring | 20 |
| 2 Adversarial Machine Learning – Wrogie Uczenie Maszynowe | 25 |
| 2.1 Wprowadzenie do technik Uczenia Maszynowego | 25 |
| 2.2 Charakterystyka Wrogiego Uczenia Maszynowego | 29 |
| 2.3 Typy ataków na algorytmy Uczenia Maszynowego | 33 |
| 2.4 Przykłady ataków na algorytmy Uczenia Maszynowego | 38 |
| 3 Budowa modelu drzewa decyzyjnego do celu Credit Scoring | 45 |
| 3.1 Analiza oraz wstępne przetworzenie wybranego zbioru danych | 45 |
| 3.2 Wybór narzędzi, budowa modelu i analiza wyników | 55 |
| 4 Atak na opracowany model | 65 |
| 4.1 Opis hipotez badawczych oraz strategii ataku | 65 |
| 4.2 Hipoteza 1 | 68 |
| 4.3 Hipoteza 2 | 71 |
| 4.4 Hipoteza 3 | 74 |
| 4.5 Hipoteza 4 | 78 |
| 4.6 Podsumowanie badań AML | 82 |
| Podsumowanie | 84 |
| Literatura | 85 |
| Spis rysunków | 91 |
| Spis tabel | 95 |
| Załączniki | 96 |

Wstęp

W dzisiejszym społeczeństwie, gdzie technologia i analiza danych stanowią fundament rozwoju, zagadnienia związane z oceną ryzyka kredytowego nabierają szczególnego znaczenia. W ramach pracy magisterskiej połączono rozważania nad dwoma kluczowymi dziedzinami dzisiejszego świata analitycznego: Credit Scoring oraz Adversarial Machine Learning.

Rozpoczęto od zrozumienia procesu oceny zdolności kredytowej. Zdefiniowanie i przedstawienie cech tego procesu ukazuje jego kluczową rolę w umożliwianiu instytucjom finansowym dokonania właściwych decyzji kredytowych. Następnie poruszono temat wyznaczania oceny punktowej, odsłaniając kryteria stosowane do oszacowania ryzyka kredytowego. W kolejnym podrozdziale uwypuklono dynamiczne zmiany, jakie niesie ze sobą rosnąca dostępność ogromnych ilości danych oraz ich analiza w procesie weryfikacji kredytowej. Rozdział podsumowano wprowadzeniem do metod Uczenia Maszynowego wykorzystywanych w dziedzinie Credit Scoring, co ukazało aktualny stan zaawansowania technologicznego w stosowaniu interpretowalnych modeli predykcyjnych.

Następnie pochyłono się nad teorią dotyczącą Adversarial Machine Learning, gdzie poruszono temat ryzyka i bezpieczeństwa algorytmów. Zapoznanie się z tym pojęciem pozwoliło na poszerzenie wiedzy dotyczącej bieżącego stanu wiedzy w tej tematyce, a także pomogło uświadomić o skali niebezpieczeństwa związanego z działalnością adversarzy, którzy próbują wykorzystywać podatność modeli na manipulacje i ataki. Charakterystyka oraz rodzaje ataków na algorytmy Uczenia Maszynowego rzucają światło na ten aspekt, ukazując konieczność analizy i implementacji mechanizmów obronnych.

W drugiej części pracy zajęto się czynnościami praktycznymi. Omówiono proces budowy modelu predykcyjnego dla wystąpienia zdarzenia default w ciągu pierwszych dwunastu miesięcy od zaciągnięcia zobowiązania kredytowego, do czego zastosowano algorytm XGBoost. Wykorzystany zestaw danych zawiera informacje o klientach aplikujących zarówno o kredyty gotówkowe, jak i ratalne. Przeanalizowano wybór danych, jakość zbiorów oraz wykorzystane narzędzia programistyczne, a efektywność modelu oceniano przez współczynnik Gini.

Pracę sfinalizowano zbadaniem czterech hipotez dotyczących ataku na wcześniej zbudowany model. W ramach testów wykorzystano różne rodzaje manipulacji danymi, celem oszukania algorytmu i zmuszenia go do generowania błędnych klasyfikacji.

1 Credit Scoring

Ocena wiarygodności kredytowej jest stosowana na całym świecie do przetwarzania wielu rodzajów pożyczek. Jest ona wykorzystywana najszerzej i z największym powodzeniem w przypadku osobistych kart kredytowych oraz kredytów hipotecznych. Ryzyko spłaty tych zobowiązań jest ściśle powiązane z czynnikami weryfikowalnymi, takimi jak dochód, ocena Biura Informacji Kredytowej, czy też demografia, tj. wiek, wykształcenie, status cywilny itp. (Caire, Barton, de Zubiria, Alexiev, & Dyer, 2006). Nieraz trudno jest ocenić, czy dana osoba zasługuje na zaufanie, czy może tym razem bank powinien się wstrzymać, nie ryzykując problemami ze spłatą danego kredytobiorcy, jednocześnie rezygnując z potencjalnego zysku.

Dokładne określenie granicy między dobrym, a złym klientem jest trudne nawet dla najbardziej doświadczonych pracowników finansowych. Zwiększona konkurencja i rosnąca presja na generowanie przychodów skłoniły instytucje udzielające kredytów do poszukiwania skutecznych sposobów pozyskiwania nowych klientów, przy jednoczesnej kontroli zysków i strat. Agresywne działania marketingowe wymusiły konieczność dokładniejszej analizy danych potencjalnych klientów, a potrzeba szybkiego i efektywnego ich przetwarzania doprowadziła do rosnącej automatyzacji procesu składania wniosków kredytowych i ubezpieczeniowych, a co za tym idzie, skrócenia czasu ich rozpatrywania (Siddiqi, 2016).

1.1 Credit Scoring - definicja i cechy

Credit Scoring można zdefiniować jako zespół metod statystycznych, używany w celu wyznaczania prawdopodobieństwa nie wywiązania się wnioskodawcy ze spłaty zaciągniętych zobowiązań w ustalonym terminie, co pomaga ustalić, czy kredyt powinien być przyznany potencjalnemu kredytobiorcy. Scoring kredytowy jest jedną z metod systematycznej oceny, która została uznana za istotnie wpływającą na obniżenie poziomu ryzyka wygenerowania strat finansowych w bankach.

W literaturze można znaleźć wiele definicji scoringu, co wynika z faktu, że jest to pojęcie w znacznym stopniu subiektywne. Ważne jest, aby model scoringowy charakteryzował się wysoką skutecznością w oddzielaniu klientów przynoszących zyski od tych, którzy prawdopodobnie przyniosą straty (Wysiński, 2013). Jest to zatem kluczowe narzędzie w rękach instytucji finansowych, dające możliwość ograniczania potencjalnego ryzyka współ-

pracy z niewiarygodnym klientem, przy jednoczesnym osiągnięciu jak największych korzyści finansowych poprzez zawieranie umów z wartościowymi kredytobiorcami(direct.money.pl, 2022).

Credit Scoring charakteryzuje się kilkoma podstawowymi cechami(mfiles.pl, 2020):

- Dane historyczne jako baza – porównuje się ze sobą charakterystyki grup kredytobiorców rzetelnych z nierzetelnymi i na tej podstawie dokonuje się oceny, czy potencjalny klient będzie terminowo spłacał zaciągnięte zobowiązanie, czy też może istnieje wysokie prawdopodobieństwo, że zachowa się on podobnie jak usługobiorcy mający problemy z uiszczaniem kolejnych rat na czas;
- Okresowość – mechanizmy wyliczania oceny wymagają częstych aktualizacji o nowe dane, w celu zapewnienia jak najwyższej skuteczności otrzymanego wyniku;
- Rzetelne zbadanie zdolności kredytowej kredytobiorców jako środek do celu jakim jest ochrona interesów kredytodawcy;
- Realizowany za pomocą zaakceptowanych i zatwierdzonych metod statystycznych.

Zwykle mówi się o dwóch typach scoringu - aplikacyjnym i behawioralnym. Pierwszy z nich jest stosowany u klientów, o których informacje są dostępne jedynie na podstawie wypełnionych przez nich wniosków kredytowych oraz danych pozyskanych z zewnętrznych źródeł np. z BIK - Biura Informacji Kredytowej.

Scoring behawioralny bierze pod uwagę informacje zgromadzone podczas wcześniejszej współpracy z klientem. Stanowi również narzędzie wspomagające monitorowanie portfela kredytowego oraz ograniczanie wysokości rezerw tworzonych na tzw. kredyty zagrożone. System scoringu behawioralnego jest szczególnie przydatny przy określaniu nowego limitu kredytowego lub modyfikacji limitu przyznanego wcześniej. Stosuje się go również w celu przeciwdziałania przekraczaniu określonych przez bank limitów na rachunkach, czy przedłużaniu warunków umów na dodatkowe produkty (np. kartę kredytową). Zatem podstawową różnicę pomiędzy scoringiem aplikacyjnym, a behawioralnym stanowi grupa docelowa klientów(Matuszyk, 2009).

W analizie ryzyka przedsiębiorstw stosuje się inny typ oceny, tzw. profit scoring (ang. scoring zysku). Taki system punktowania pozwala na określenie maksymalnego zysku, zamiast minimalizacji ryzyka związanego z obsługą danego klienta. Profit scoring, jako

rozszerzenie podstawowego modelu scoringowego, bierze pod uwagę szereg dodatkowych czynników ekonomicznych tj. strategie marketingowe, dobór polityki cenowej, czy też poziom obsługi (bankier.pl, 2007).

Punktowa ocena wiarygodności kredytowej budowana jest na zasadzie przyznawania punktów za określone cechy kredytobiorcy, gdzie im wyższy wynik wnioskodawca osiągnie, tym większa szansa, że spłaci kredyt w terminie. Tabela punktowa tworzona jest na podstawie analizy statystycznej bazy danych klientów z przeszłości, gdzie poszukuje się cech, które w jak najlepszy sposób oddzielają od siebie dobrych i złych biorców kredytowych (bankier.pl, 2012).



Rysunek 1: Przykład cechy wykorzystanej w Credit Scoring (bankier.pl, 2012)

Dla pewnego modelu, wpływ posiadania własnego mieszkania na spłacenie zobowiązania bez opóźnień zwizualizowano na rysunku 1. Widać na nim, że wśród osób mających problemy ze spłatą pożyczki (200 klientów), aż 95 procent (190 klientów) stanowią osoby wynajmujące nieruchomość. Zatem posiadanie własnego mieszkania będzie stawiać w uprzywilejowanej pozycji potencjalnych klientów banku i otrzymają oni wyższą ocenę za tę cechę w ogólnym scoring'u. Taką logiką kierują się zarówno banki, jak i BIK. Sposób wyliczania oceny punktowej często jest tajemnicą w instytucjach finansowych i zwykle nie dowiemy się jaki score otrzymaliśmy oraz co jest powodem takiego wyniku.

W Biurze Informacji Kredytowej, mimo iż nie poznamy pełnej logiki oceniania, pośrednio wiadome są najważniejsze kryteria oszacowań. Wśród najważniejszych z nich należy wymienić(bankier.pl, 2012):

- Liczba nieterminowo spłacanych zobowiązań;
- Wysokość zobowiązań (np. na karcie kredytowej lub debetowej);
- Czas zwłoki ze spłatą zobowiązania;
- Czas jaki upłynął od ostatniego wykroczenia.

Nieodłącznym elementem predykcji zachowań potencjalnych kredytobiorców względem zaciągniętego zobowiązania jest szacowanie zdolności kredytowej. O ile w przypadku kredytów gotówkowych banki często stosują dość liberalne podejście, to gdy pod uwagę brane są duże kwoty pożyczki, tak jak ma to miejsce w przypadku kredytów hipotecznych, etap estymacji zdolności stanowi gęste sito dla wnioskujących, szczególnie w czasach kryzysów gospodarczych, gdzie najczęściej mamy do czynienia z wysokimi stopami procentowymi. Oczywiście analiza zdolności kredytowej opiera się na różnych kryteriach, które dodatkowo różnią się między poszczególnymi bankami, jednak relacja zarobków do wydatków, które są pośrednio zależne od poziomu stóp procentowych, stanowi bazę do dalszej oceny(habza.com.pl, 2022).

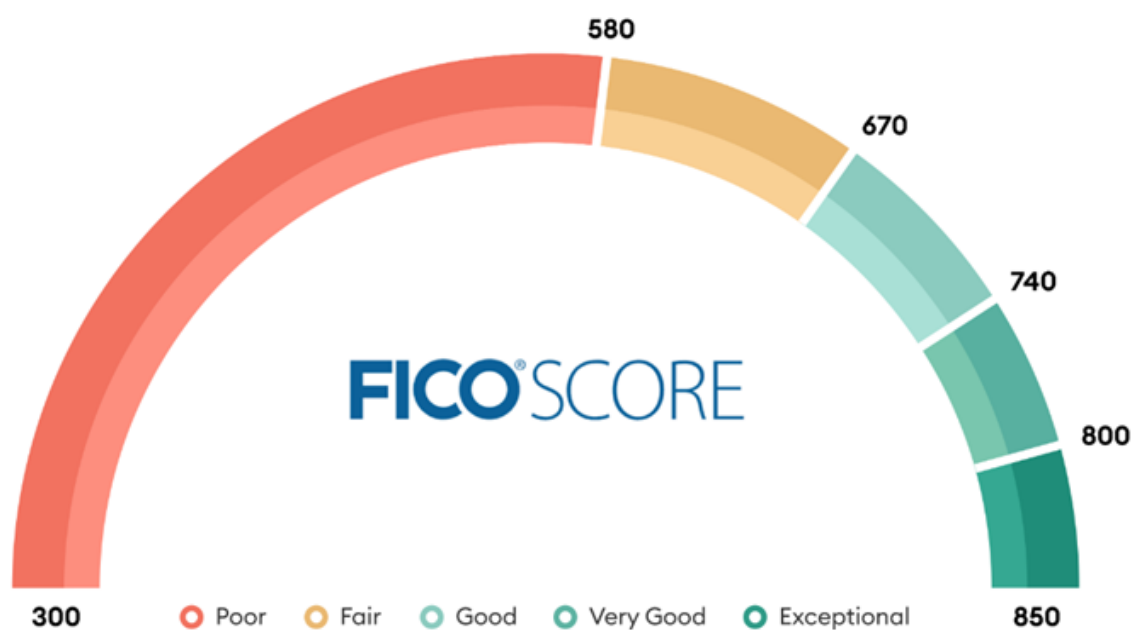
Biuro Informacji Kredytowej nie ukrywa jakie cechy są preferowane przez banki oraz na co należy zwrócić uwagę, aby podnieść kwotę maksymalnego możliwego do uzyskania kredytu(bik.pl, 2022):

- Dochody i forma zatrudnienia - umowa o pracę to zdecydowanie wariant, który wpływa dobrze na zdolność kredytową. W przypadku umowy o dzieło czy samozatrudnienia bank ma mniejszą pewność co do stabilności zarobków. Mniej oczywista jest również kwestia dochodów mikroprzedsiębiorców;
- Obciążenia wydatkami - Koszty stałe, kredyty, pożyczki, karty kredytowe - im mniej z nich wnioskodawca posiada tym potencjalnie wyższa miesięczna rata jaką może obsłużyć;

- Warunki wnioskowanego kredytu:
 - Czas - wydłużenie okresu kredytowania może obniżyć wysokość miesięcznej raty, co ma bezpośredni wpływ na zdolność kredytową;
 - Wspólny kredyt - rodzice, partner, partnerka, rodzeństwo to potencjalnie dobrzy kandydaci do wspólnego kredytu. Takie rozwiązanie nie tylko poprawia zdolność kredytową, ale jednocześnie daje dodatkowe zabezpieczenie dla banku;
 - Równe raty - lepiej wybrać kredyt o równych ratach kapitałowo-odsetkowych, aby zwiększyć swoje szanse na pozytywną decyzję kredytową;
- Historia kredytowa w BIK- z Raportu BIK można dowiedzieć się jakie informacje na temat wnioskodawcy dotychczas przekazywały do BIK banki, SKOK-i i firmy pożyczkowe.

1.2 Wyznaczanie oceny punktowej

W Stanach Zjednoczonych score kredytowy zawiera się w granicach od 300 do 850 punktów, gdzie w zależności od instytucji za odpowiednio dobry wynik traktuje się wartości powyżej 661-670 punktów(experian.com, 2021). Przedziały ocen deklarowane przez FICO (instytucję opisano szerzej w podrozdziale 1.3) przedstawiono na rysunku 2.




Rysunek 2: Diagram oceny wiarygodności kredytowej według firmy FICO(forbes.com, 2021)

W sieci można znaleźć darmowe kalkulatory score'u kredytowego i choć nie należy bezkrytycznie ufać ich kalkulacjom, jako że nie znamy dokładnych mechanizmów oraz zbiorów danych na podstawie których zbudowano dany model, to niektóre z nich potrafią zwrócić wyniki, które z pewną dozą niepewności możemy przyjąć jako prawdopodobne wartości wyliczane w profesjonalnych instytucjach. Jednym z takich narzędzi jest kalkulator na stronie CalcXML, który zwraca wynik w skali FICO. Osoba zainteresowana obliczeniem swojej indywidualnej oceny powinna przygotować kilka podstawowych informacji na swój temat(calcxml.com, 2023):

- Czy Wnioskujący posiadał pożyczkę lub kartę kredytową przez okres dłuższy niż 6 miesięcy;
- Ile lat temu Wnioskujący po raz pierwszy wziął pożyczkę lub posiadał kartę kredytową;
- Które z poniższych zobowiązań Wnioskujący posiada lub posiadał:
 - Kredyt hipoteczny;
 - Karta kredytowa;
 - Pożyczka na samochód/Pożyczka studencka/Inna pożyczka.
- Suma limitów ze wszystkich posiadanych przez Wnioskującego kart kredytowych;
- Czy Wnioskującego kiedykolwiek dotyczyło którekolwiek z poniższych "negatywnych zdarzeń":
 - Bankructwo;
 - Interwencja komornika/przejęcie własności;
 - Problemy podatkowe;
 - Inne "negatywne zdarzenie".
- Kiedy miało miejsce ostatnie "negatywne zdarzenie", które dotyczyło Wnioskującego.

Przykładowa kalkulacja, dla osoby posiadającej kartę kredytową z limitem 2000 dolarów od ponad roku, nie posiadającej dodatkowych zobowiązań, która w ostatnich dwunastu miesiącach nie wysyłała wniosków o kredyt, nie spóźniała się z spłatą zobowiązań oraz



Input And Assumptions

Have you had a credit card or loan for at least 6 months?

Yes

How many years ago did you get your first credit card or loan? (0 to 120)

1

Checkmark each type of credit account or loan that you have on your credit report, whether open or closed.

☐ Mortgage
☒ Credit Card
☐ Auto Loan
☐ Student Loan
☐ Other Loan
☐ Consumer Finance Account

How many times have you applied for credit in the last year?

0 times

When did you last miss a payment on any of your credit accounts?

Never

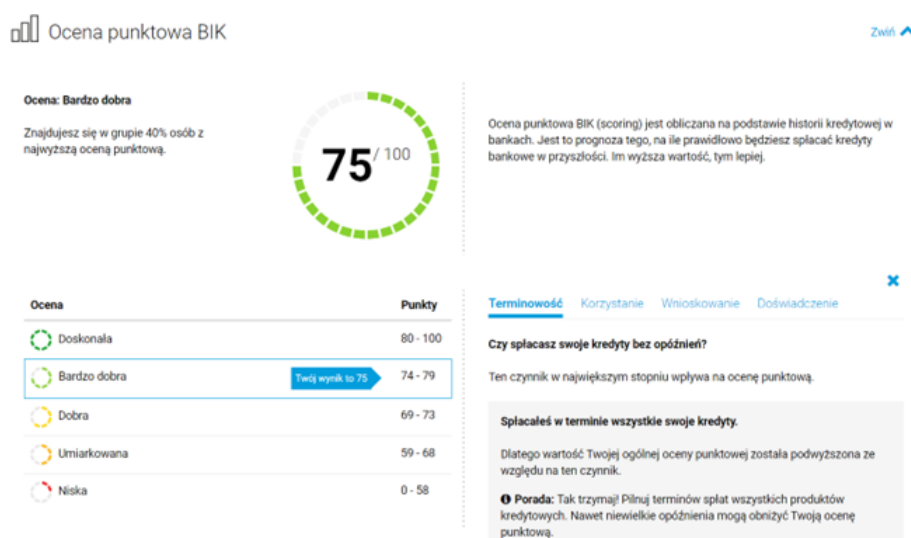
What is your total credit limit? (Add up the credit limits on all your credit card accounts.) (\$)

2,000

Rysunek 3: Fragment kalkulacji ze strony calcxml.com (calcxml.com, 2023)

nie brała udziału w żadnym z "negatywnych zdarzeń", której fragment przedstawiono na rysunku 3, daje wynik od 740 do 790 punktów, co oznacza bardzo dobry score kredytowy.

Od 21 grudnia 2017 roku BIK generuje ocenę punktową w nieco inny sposób. Dotychczas score zawierał się w zakresie od 192 do 631 punktów, co wizualizowane było za pomocą gwiazdek, gdzie im więcej gwiazdek, tym wyższa szansa na otrzymanie kredytu. Aby uczynić reprezentację graficzną bardziej czytelną dla osób fizycznych, ocenę przekształca się do postaci od 1 do 100 (totalmoney.pl, 2020) Zostało to zilustrowane w postaci wykresu kołowego na rysunku 4.



Rysunek 4: Wizualizacja oceny punktovej w BIK (źródło opracowanie własne)

Przed zmianą z grudnia 2017, Biuro Informacji Kredytowej do wyliczania oceny punktowej wykorzystywało tzw. Model II generacji o nazwie BIKSco CreditRisk. Dla tego algorytmu zakres możliwych do uzyskania rezultatów zawierał się pomiędzy 192, a 631 punktów i nie był w żaden sposób przekształcany. W najnowszych raportach wykorzystywany jest model III generacji o nazwie BIKSco CreditRisk 3, a punktacja zawiera się od 98 do 711 punktów. Jednakże otrzymany wynik ulega przekształceniu i tak jak jest to widoczne na rysunku 4, przedstawiony jest w zakresie od 1 do 100, co może mylnie wskazywać, że jest to procent maksymalnej, możliwej do uzyskania punktacji (scoringexpert.pl, 2018).

Bazując na cytowanym źródle, wynik wyliczonej punktacji podlega procesowi normalizacji według równania 1:

Równanie 1. Wzór na przekształcenie oceny wyliczonej z modelu BIKSco CreditRisk 3 do przedziału znormalizowanego 1-100 (experian.com, 2021)

$$S_{new} = \frac{S - min}{max - min} * (new_{max} - new_{min}) + new_{min}$$

gdzie:

- S_{new} - ocena punktowa z raportu BIK;
- S - oryginalna ocena punktowa z modelu BIKSco CreditRisk 3;
- min - minimalna wartość punktów z modelu BIKSco CreditRisk 3 (wynosi 98);
- max - maksymalna wartość punktów z modelu BIKSco CreditRisk 3 (wynosi 711);
- new_{max} - maksymalna wartość punktów z nowego zakresu (wynosi 100);
- new_{min} - minimalna wartość punktów z nowego zakresu (wynosi 1).

Na podstawie wartości widocznej dla osoby fizycznej (punktacji z przedziału 1-100), można uzyskać wynik wyliczony z algorytmu na podstawie równania 2:

Równanie 2. Wzór na wyliczenie wartości otrzymanej z modelu BIKSco CreditRisk 3 na podstawie wartości uzyskanej z Biura Informacji Kredytowej (experian.com, 2021)

$$S = \frac{613 * S_{new} + 9089}{99}$$

Biuro Informacji Kredytowej proponuje swoją interpretację znormalizowanej oceny punktowej, przedstawioną na rysunku 4, według której potencjalny kredytobiorca może ocenić swoje aktualne szanse na otrzymanie pożyczki.

Na podstawie cytowanej tabeli 1, można zinterpretować również ocenę nieznormalizowaną.

| Ocena punktowa BIK | Słowna ocena | Komentarz |
|-----------------------------|--------------|---|
| 550+ (74+) | Bardzo dobra | Twój „scoring BIK” przewyższa średni „scoring BIK” Polaków. W ocenie banków Twoja wiarygodność kredytowa powinna być bardzo wysoka |
| 500-549 (66-73) | Dobra | Twój „scoring BIK” oscyluje blisko średniego „scoringu BIK” Polaków. Banki zapewne ocenią Cię jako rzetelnego kredytobiorcę |
| 400-499 (52-65) | Przeciętna | Twój „scoring BIK” jest poniżej średniego „scoringu BIK” Polaków. Tylko część banków oceni Twoją wiarygodność kredytową jako wystarczającą do uzyskania kredytu |
| poniżej 400 (poniżej 52) | Słaba | Twój „scoring BIK” jest znacznie poniżej średniego „scoringu BIK” Polaków. Taki scoring wskazuje, że jesteś ryzykownym kredytobiorcą dla banków. Możesz więc mieć problem z uzyskaniem kredytu, jeśli bank oprze się na „scoringu BIK” przy ocenie Twojego ryzyka kredytowego |

Tabela 1: Interpretacja oceny punktowej BIK(scoringexpert.pl, 2017)

Za jedną z najbardziej znanych metod Credit Scoring’u jest uważana przytaczana na łamach tej pracy amerykańska metoda FICO (Fair, Isaac and Company). Zdefiniowana w 1989 roku, opiera się na 5 czynnikach(pl.economy pedia.com, 2021):

- Historia płatności (35 procent punktów) – na bazie dotychczasowych zobowiązań ocenia się, czy dana osoba wywiązuje się z nich na czas;
- Wykorzystanie kredytu (30 procent) – jeśli potencjalny klient instytucji finansowej dotychczas wykorzystywał niewielki procent dostępnych limitów kredytowych (np. limit na karcie kredytowej), ma on większe szanse na wyższą ocenę;
- Długość historii kredytowej (15 procent) – jeśli wnioskodawca jest doświadczonym

pożyczkobiorcą i przez długi czas poprawnie wywiązuje się z zobowiązań otrzymuje wyższą notę w tabeli punktowej;

- Nowe kredyty (10 procent) – złożenie wielu wniosków kredytowych przez osobę poszukującą kredytu, może wzbudzać wątpliwości instytucji przed udzieleniem pożyczki;
- Rodzaje wykorzystanego kredytu (10 procent) – dla banków mile widziane jest doświadczenie wnioskującego w zarządzaniu różnymi rodzajami kredytów (karta kredytowa, kredyt hipoteczny, pożyczka gotówkowa itp.).

Rozwój Credit Scoringu jest jednym z powodów, dla których rynek kredytów konsumenckich w Stanach Zjednoczonych w latach 90. XX w. eksplodował. Kredytodawcy czuli się bardziej pewni w udzielaniu pożyczek szerszym grupom ludzi, ponieważ mieli dokładniejsze narzędzie do pomiaru ryzyka. Scoring kredytowy pozwolił im również na szybsze podejmowanie decyzji, umożliwiając rozpatrzenie większej liczby wniosków, czego rezultatem był bezprecedensowy wzrost ilości dostępnych kredytów konsumenckich (Weston, 2012). Banki bardzo skrupulatnie podchodzą do operowania swoimi pieniędzmi, dokładnie „prześwietlając” swoich klientów pod kątem wypłacalności. Polskie instytucje finansowe często posiłkują się opinią BIK, jednakże równie chętnie stosują także własne algorytmy oceny, których szczegółów zwykle szerzej nie udostępniają.

1.3 Score kredytowy w erze Big Data

Historia rozwoju modeli scoringowych sięga tak daleko, jak historia pożyczania i spłacania. Widać to w szczególności w potrzebie ustalenia odpowiedniej stopy procentowej, uwzględniającej ryzyko braku możliwości odzyskania pożyczonych pieniędzy. Wraz z nadejściem współczesnej ery statystyki w XX. wieku rozpoczęto opracowywanie technik oceny prawdopodobieństwa niewykonania przez kredytobiorcę zobowiązania do zwrotu środków. Pod uwagę brano podobieństwo przypisanych cech do tych, którymi charakteryzują się osoby niewywiązujące się ze zobowiązań w przeszłości. Tego typu metody statystyczne są obecnie stosowane powszechnie przez praktycznie wszystkie banki, a również znaczną część pozostałych instytucji finansowych (prawniczydotblog.wordpress.com, 2019).

Dziedzina Credit Scoringu nie należy do odkryć bieżącej dekady, a jej początki można upatrywać w połowie XX wieku, wraz z utworzeniem w 1956 roku wspomianej już wcześniej firmy FICO. Przedsiębiorstwo założyli inżynier Bill Fair oraz matematyk Earl Judson Isaac (Wikipedia, 2022). Firma zajmowała się budową systemów scoringowych, jednakże przez długi czas stosowano zapis papierowy (StatSoft, 2010). Początkowo zajmowano się głównie procesami weryfikacji wniosków kredytowych w bankach, stosując proste tabele punktowe i głównie opierając się na tzw. metodzie eksperckiej (Thonabauer & Nosslinger, 2004). Sposób musiał być na tyle łatwy i intuicyjny, aby dawać możliwość obiektywnej oceny zdolności potencjalnego kredytobiorcy do wywiązania się z zaciągniętego zobowiązania kredytowego również mniej doświadczonym i niewykwalifikowanym pracownikom banku (Thomas, Edelman, & Crook, 2002). Jednym z pierwszych i najważniejszych osiągnięć Fair Isaac & Company było utworzenie pierwszego, wykorzystywanego komercyjnie, systemu scoringowego (Poon, 2007).

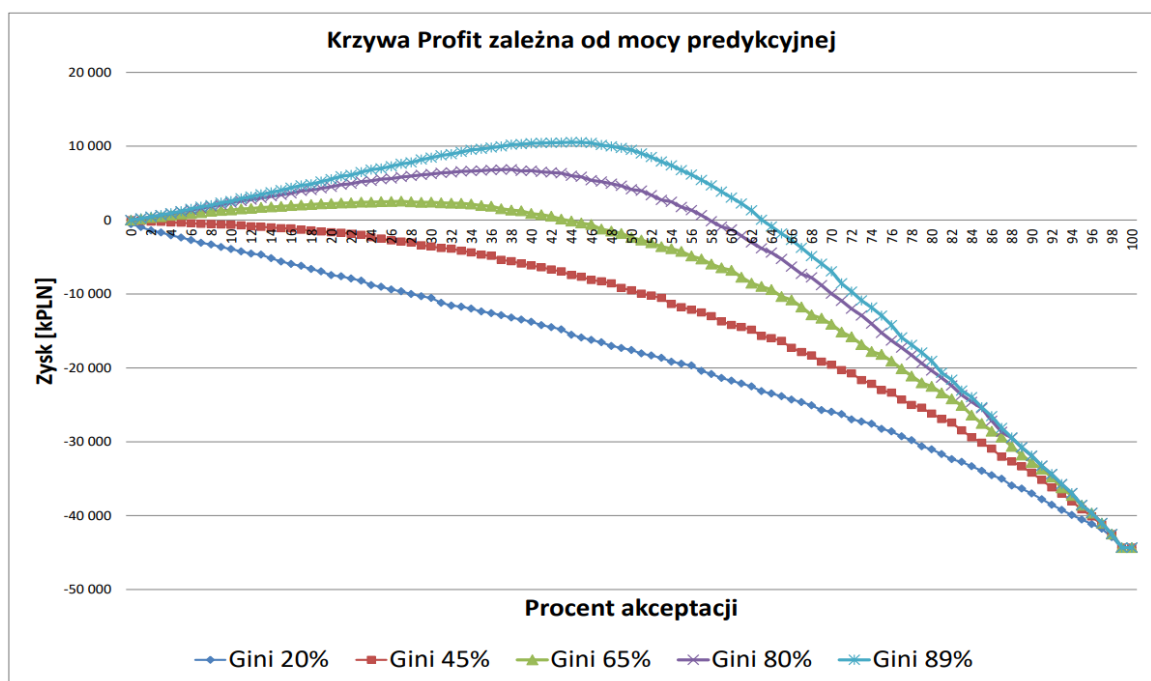
Jednakże zanim FICO rozpoczęło swoją działalność, ludzkość zebrała już pierwsze doświadczenia ze złymi kredytobiorcami, a co za tym idzie zaczęto zastanawiać się nad ich właściwym zidentyfikowaniem zanim zostanie im udzielona pożyczka. Rok 1826 przyjmuje się za początek wymiany informacji między wierzycielami, co miało na celu podniesienie jakości "filtracji" właściwych klientów, natomiast w 1899 roku w Atlancie rozpoczęto zbieranie danych na większą skalę. Po powstaniu Fair Isaac & Company rozwój w dziedzinie udzielania kredytów znacząco przyspieszył, co skutkowało powstawaniem konkurencyjnych dla FICO firm (wśród nich warto wymienić chociażby Experian czy Equifax), ale też było bodźcem do podwyższenia jakości usług (ecomparemo.com, 2020). Zwieńczenie i zarazem wykładnik wzrostu znaczenia score'u kredytowego stanowiło wejście Fair, Isaac and Company na giełdę w Nowym Jorku w lipcu 1987 roku (fico.com, 2023).

Wraz z rozwojem technik informatycznych rozpoczęto wdrażanie bardziej zautomatyzowanych procesów scoringowych. Najpowszechniej do tego celu wykorzystywano model regresji logistycznej, jednakże dziś stosuje się szeroką gamę różnych metod predykcyjnych tj. sieci neuronowe, lasy losowe czy drzewa decyzyjne (Przanowski, 2014). Analitycy i inżynierowie danych w dzisiejszych czasach wykorzystują potencjał płynący z zastosowania skomputeryzowanych metod predykcji zdarzeń.

Jednym z podstawowych podejść w modelowaniu credit scoring jest wyliczenie prawdo-

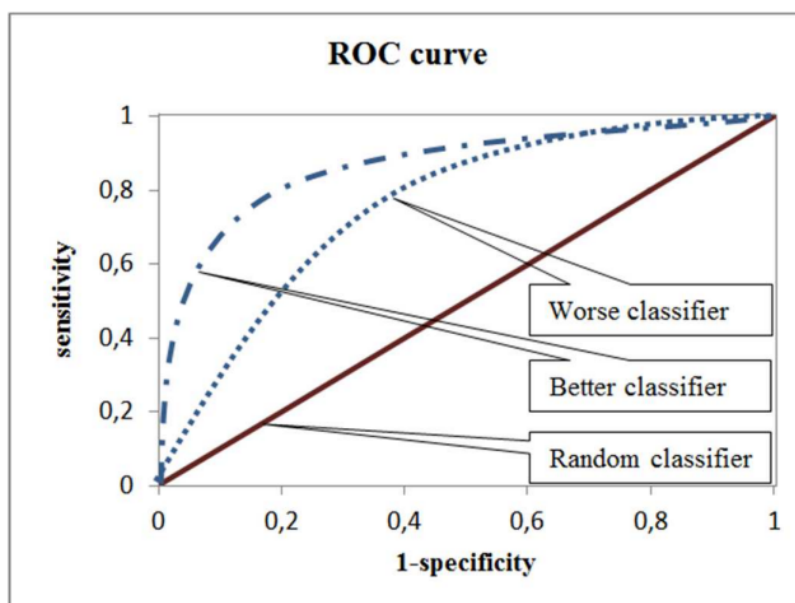
podobieństwa wystąpienia zdarzenia znanego pod nazwą "default". Finalnie, bazując na zebranych danych, zarówno tych opisujących klientów z przeszłości, jak również dotyczących wnioskodawcy, model powinien obliczyć procentową szansę na to, że wnioskodawca "wpadnie w default", gdzie np. zmienna default12 wskazuje, iż dłużnik w ciągu 12 miesięcy od zaciągnięcia długu spóźniał się ze spłatą raty o co najmniej 90 dni (Przanowski, 2015). Pozornie logika nakazywałaby odrzucać wnioski, dla których prawdopodobieństwo problemów ze spłatą wynosi więcej niż 50 %, jednakże nie jest to tak oczywiste, jak może się na pierwszy rzut oka wydawać. Celem zobrazowania tego zagadnienia warto jest omówić i zrozumieć działanie tzw. Krzywej Profit (rysunek 5).

Na Krzywej Profit wizualizuje się wartość zysku w zależności od procentu zaakcepto-



Rysunek 5: Krzywa Profit zależna od mocy predykcyjnej (Przanowski, 2023)

wanych wniosków dla różnych modeli predykcyjnych, gdzie każdy z nich opisuje się współczynnikiem Giniego. Model buduje się "trenując go" na zbiorze treningowym, złożonym z danych historycznych, gdzie algorytm wychwytuje zależności cechujące klientów spłacających zobowiązania bez opóźnień, a także uczy się odróżniać niewiarygodnych wnioskodawców. Następnie na zbiorze testowym porównuje się wnioski prognozowane przez model z sytuacją jaka w rzeczywistości miała miejsce i na tej podstawie buduje się tzw. krzywą ROC - przykładowa wizualizacja na rysunku 6.



Rysunek 6: Krzywa ROC i jej możliwe warianty (Gajowniczek et al., 2014)

Do skonstruowania krzywej ROC niezbędne jest wyliczenie czterech parametrów, składających się na tzw. macierz pomyłek (Fawcett, 2005):

- False Positive (FP) - model wskazał, że dana osoba wpadnie w default, mimo iż w rzeczywistości sumiennie spłacała kredyt;
- False Negative (FN) - model wskazał, że dana osoba nie wpadnie w default, mimo iż w rzeczywistości miała kłopoty ze spłatą;
- True Positive (TP) - model wskazał, że dana osoba wpadnie w default, co okazało się zgodne z rzeczywistością;
- True Negative (TN) - model wskazał, że dana osoba nie wpadnie w default, co okazało się zgodne z rzeczywistością.

Na podstawie powyższych wartości oblicza się wielkości takie jak czułość, swoistość itp., które pozwalają na zwizualizowanie jakości modelu na krzywej ROC. Jednakże do tego celu należy wyliczyć wiele punktów, a dokonuje się tego poprzez badanie powyżej opisanych właściwości w zależności od przyjętego punktu odcięcia (Fawcett, 2005).

W procesie predykcyjnym wyznaczane są prawdopodobieństwa wejścia w default, a decyzja o tym czy dana osoba zostanie zaklasyfikowana jako wystarczająco wiarygodna zależy od przyjętego poziomu akceptacji. Jeśli dla pewnego wnioskodawcy prawdopodobieństwo default zostało wyznaczone na 35%, a próg odcięcia założono na poziomie 40%,

kredyt zostanie udzielony. Jednakże w przypadku zdefiniowania punktu podziału równego 30%, wniosek zostanie odrzucony. W celu stworzenia wykresu jakości modelu należy wyliczyć wartości macierzy pomyłek dla wielu progów odcięcia (Fawcett, 2005). Po wykonaniu tych operacji możemy wyznaczyć wartość współczynnika Giniego dla rozpatrywanego algorytmu, który wynosi dwukrotność pola pod krzywą ROC, a nad krzywą klasyfikatora losowego (na rysunku 6 oznaczona jako "Random classifier").

Wracając do Krzywej Profit (rysunek 5), łatwo zauważyć, że im skuteczniejszy model (im wyższy współczynnik Giniego), tym wykres jest bardziej wybrzuszony, co oznacza możliwość osiągnięcia wyższych zysków. Jako że straty powodowane przez jednego dłużnika potrafią przewyższyć profity uzyskiwane przez bank we współpracy z wieloma wiarygodnymi klientami, bardzo istotny jest dobór odpowiedniego progu akceptacji. Przytoczony wykres, dla najlepszego z modeli, sugeruje wybór progu odcięcia pomiędzy 40%, a 46% (Przanowski, 2023).

Klasyczne karty punktowe cechują się prostotą w interpretacji wyników, a wykorzy-

| Attribute | Variable | Partial Score |
|--------------|-----------------|---------------|
| <20 | Age | 10 |
| 20>= and <34 | | 20 |
| 35>= | | 30 |
| Bad | Payment history | 10 |
| Not good | | 25 |
| Good | | 40 |

Tabela 2: Klasyczna karta oceny punktowej (Przanowski, 2023)

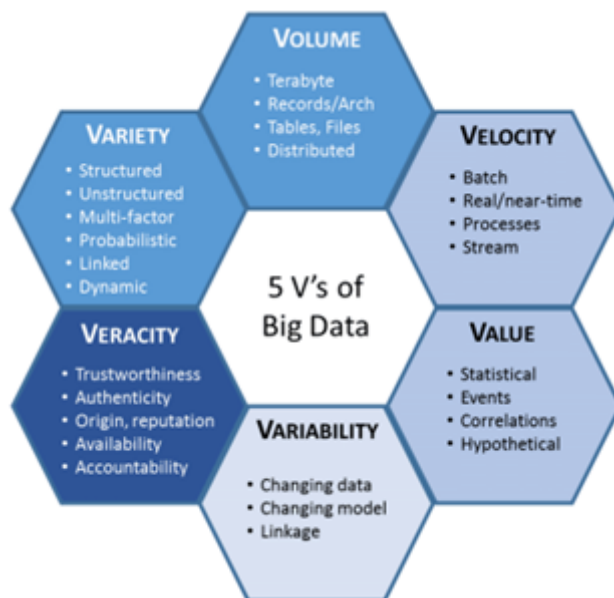
stanie takich jak tabela 2 nie wymaga wiedzy analitycznej wymaganej w stosowanych współcześnie metodach komputerowych. Trudno zatem oprzeć się wrażeniu, iż klasyczny Credit Scoring usuwa się w cień na korzyść narzędzi z dziedziny Big Data, choć w rzeczywistości staje się jedną z poddziedzin szerokiej tematyki „dużych danych” (Przanowski, 2014). Lecz czym właściwie jest Big Data?

Pojęcie to powinno iść w parze z rozbudowanymi systemami informatycznymi, które dają możliwość przetwarzania dużych danych. Często spotykanym jest tzw. 5V Big Data (zwizualizowane na rysunku 7 (researchgate.net, 2017)), dające ogólne spojrzenie na podstawowe cechy tejże dziedziny (techtarg.com, 2021):

- Volume (ang. wolumen, wielkość) – jeśli mamy do czynienia z subiektywnie dużą

ilością danych (co dokładnie oznacza „duża ilość” nie zostało precyzyjnie zdefiniowane), możemy nazwać ich przetwarzanie jako Big Data;

- Velocity (ang. szybkość) – ze względu na ich ilość, dane muszą być odpowiednio szybko procesowane, najczęściej w czasie rzeczywistym;
- Variety (ang. różnorodność) – technologia musi radzić sobie z operowaniem na danych zróżnicowanego, nie koniecznie uporządkowanego typu;
- Veracity (ang. wiarygodność) – dane najczęściej nie są wysokiej jakości, a ich braki czy też błędne informacje w nich zawarte stawiają wymóg odpowiedniej odporności na tego rodzaju zaburzenia;
- Value (ang. wartość) – kluczowa przy przetwarzaniu danych jest możliwość uzyskania na ich bazie istotnych informacji, które mogą wnieść pewną wartość dla przedsiębiorstwa, dokonującego lub zlecającemu wykonanie takich analiz.



Rysunek 7: Schemat 5V Big Data(researchgate.net, 2017)

Początkowo Credit Scoring specjalizował się głównie we wspomaganiu procesów decyzyjnych w bankach, a narzędzia Big Data stosowane były w globalnych firmach świadczących usługi w świecie wirtualnym tj. Google, Amazon czy Facebook. Z kolei w Polsce zarządzaniem dużymi danymi na poważnie zainteresowały się jako pierwsze Onet czy portal Nasza Klasa(Przanowski, 2014). Mimo zainteresowania różnymi branżami, Big Data i Credit Scoring poruszają podobne problemy ze strony merytorycznej, gdzie głównym

i najpoważniejszym problemem zawsze był kluczowy element ich funkcjonowania – dane.

Modele Credit Scoring służą do prognozowania zjawisk na podstawie dotychczas zaobserwowanej i zebranej historii danych. Proces spłacania kredytów najczęściej trwa wiele lat, zatem potrzeba dużo czasu, aby zebrać dostatecznie reprezentatywną pulę informacji rzeczywistych, którą następnie można wykorzystać do sprawdzenia użyteczności i poprawności skonstruowanego modelu (Przanowski, 2014).

W przypadku danych bankowych, sytuacja jest jeszcze trudniejsza z uwagi na wrażliwość informacji. Skutkuje to koniecznością występowania do instytucji finansowych z oficjalnymi podaniami, a otrzymane dane często są zafałszowane i zanonimizowane, co zwykle uniemożliwia ich zinterpretowanie. Znacząco utrudnia to tworzenie odpowiednio wiarygodnych modeli scoringowych, na co analitycy odpowiadają tworzeniem własnych, symulowanych danych (Przanowski, 2014).

Dziedzina Big Data nie jest bez wad i mimo jej niekwestionowanej przydatności, bez trudu można wymienić niepowodzenia i wyzwania jakie napotyka, gdzie najistotniejsze z nich to (Przanowski, 2023):

- Dane często nie są zbierane, a jeśli ktoś już je magazynuje, nie zapewnia odpowiedniej ich przydatności i interpretowalności;
- Problem jakości danych;
- Liczne braki danych;
- Brak publicznych danych, dostępnych i przykładowych;
- Brak inwestycji w przygotowanie i wykształcenie inżyniera danych.

1.4 Metody Uczenia Maszynowego wykorzystywane w dziedzinie Credit Scoring

Dotychczas w bankowości nie stosowano powszechnie niektórych technik Machine Learningu (ML) do zarządzania ryzykiem, a geneza takiego postępowania była zrozumiała – modele są trudne w interpretacji, a ponadto generują popyt na wysoce wyspecjalizowanych pracowników. Z drugiej zaś strony, rynek konkurencyjny zmienia się - transformacja cyfrowa, czy też nowy model bankowości otwartej wywierają wpływ na praktyki zarządzania ryzykiem. W tym kontekście stosowanie technik Uczenia Maszynowego zapewnia

istotną przewagę, skracając czas podejmowania decyzji w procesach kredytowych oraz podnosząc ich skuteczność (crif.pl, 2018).

Techniki stosowane do opracowania kart scoringowych to np. dyskryminacja statystyczna i metody klasyfikacji. Należą do nich modele regresji liniowej (łatwo interpretowalne, oparte na metodzie minimalizacji sumy kwadratów reszt), analiza dyskryminacyjna (odmiana regresji stosowana do klasyfikacji), modele logitowe i probitowe (maksymalizacja prawdopodobieństwa zaobserwowania wartości), oraz tzw. modele eksperckie (np. proces analizy hierarchicznej – AHP)(Group, 2021).

Natomiast główne metody Uczenia Maszynowego, jakie stosuje się przy Credit Scoring’u należą do grupy technik klasyfikacji nadzorowanej(Bajek, 2011). W dalszej części pracy, celem dokładniejszego opisanie, skupiono się na dwóch, najpopularniejszych metodach stosowanych do budowy modeli scoringowych, jakimi są regresja logistyczna i drzewa decyzyjne, pomijając równie ciekawe, aczkolwiek rzadziej używane podejścia tj. sieci neuronowe, lasy losowe czy też SVM – Support Vector Machines (ang. Metoda Wektorów Nośnych)(Statsoft, 2010).

Regresja logistyczna (LR) jest metodą statystyczną umożliwiającą ocenę wpływu wielu cech - tzw. zmiennych objaśniających - na szanse zajścia zdarzenia, np. zachorowania na pewną chorobę, czy też spłaty kredytu w planowanym terminie(Deryło, 2021). Model LR jest szczególnym przypadkiem uogólnionego modelu liniowego. Znajduje zastosowanie, gdy zmienna zależna jest dychotomiczna, to znaczy przyjmuje tylko dwie wartości takie jak np. sukces lub porażka, zwykle reprezentowane jako cyfry 1 i 0(statystyka.az.pl, 2021).

Starając się jak najkrócej scharakteryzować to podejście, należy podkreślić, że głównym celem modeli regresji logistycznej jest znalezienie najlepszych współczynników(tzw. wag), które minimalizują błąd pomiędzy przewidywanym prawdopodobieństwem, a obserwowanym wynikiem. Realizowane jest to za pomocą algorytmu optymalizacji, takiego jak opadanie gradientu, celem dostosowania współczynników, aż model będzie odpowiednio pasował do danych, na których jest uczony, co następnie walidowane jest na zbiorze testowym(newsblog.pl, 2022). Wartości zmiennej dychotomicznej możemy przekształcić w postać prawdopodobieństwa wystąpienia danego zdarzenia, które przyjmuje wartości pomiędzy 0 lub 1. Gdy zastosuje się transformację logit możliwe jest zlinearyzowanie modelu LR i przedstawienie go w postaci regresji liniowej(naukowiec.org, 2014).

Regresja logistyczna jest najczęściej wykorzystywaną techniką modelowania scoringowego. Jej największą zaletą jest stabilność w czasie, co czyni ją względnie odporną na zaburzone dane i może sprawiać wrażenie, że metoda jest odporna na ataki. Jej zaleta może być również interpretowana jako wada, ponieważ model może powodować nieoptymalność dopasowania do koniunktury rynkowej(Karolak, 2014).

Jako przykład implementacji modelu regresji logistycznej można przedstawić badanie wpływu wysokości dochodów na fakt palenia papierosów przez badaną próbę osób. Przyjmijmy, że zmienna niezależna, jaką jest w tym przypadku ilość dochodu, jest zmienną ilościową, a palenie papierosów określamy binarnie, gdzie 1 oznacza, że osoba pali papierosy, a 0 zdarzenie przeciwne. Jedna z takich analiz wskazała, że poziom dochodów jest istotny statystycznie, a osoby zarabiające więcej mają wyższą skłonność do palenia wyrobu tytoniowego(naukowiec.org, 2014).

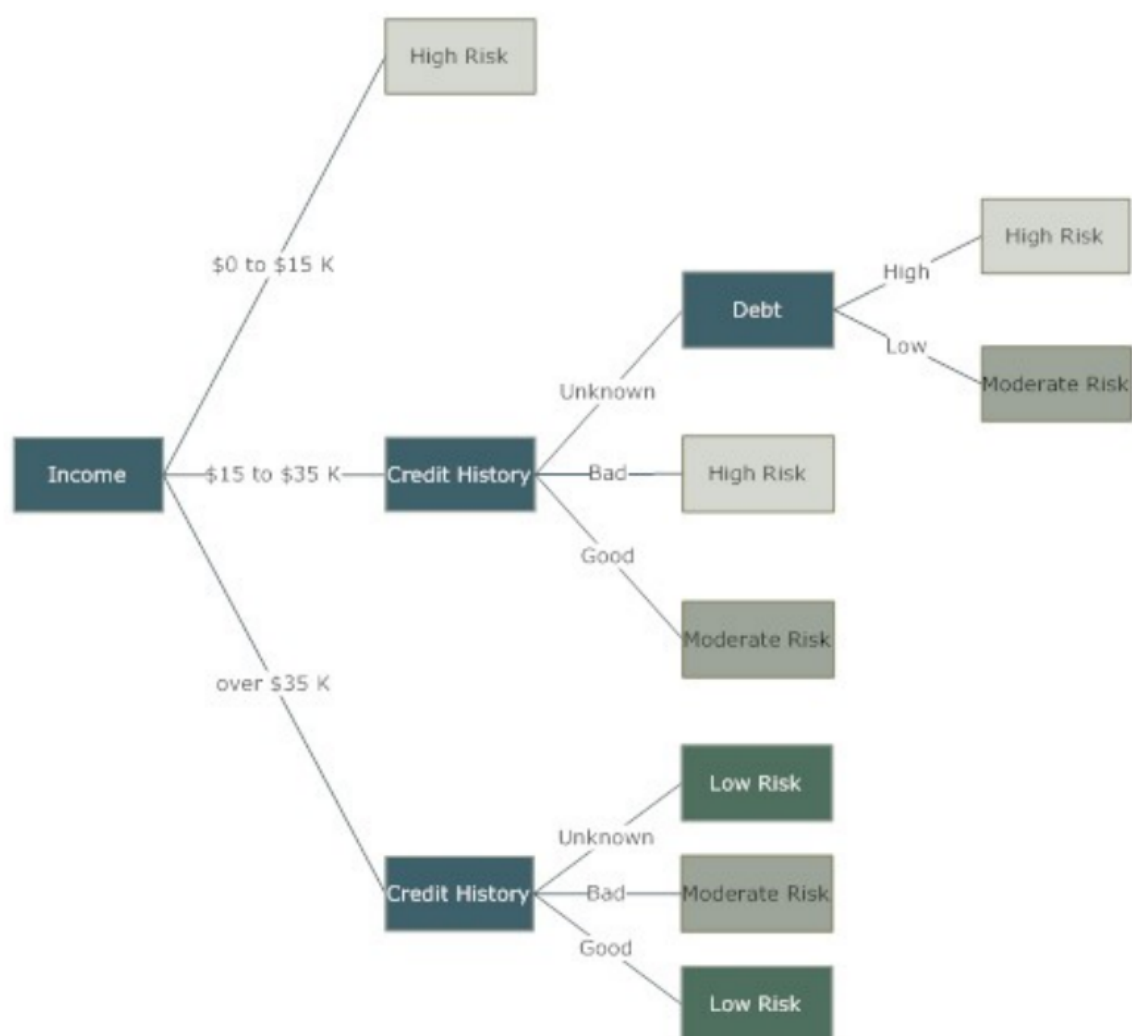
Drzewa decyzyjne (DT) są modelami o strukturze drzewiastej, podejmującymi decyzje na podstawie wartości poszczególnych cech. Można ich używać zarówno do klasyfikacji binarnej(służy do przewidywania, do której z dwóch klas/kategorii należy wystąpienie danych), jak i wieloklasowej (odnosi się do zadań klasyfikacyjnych, które mają więcej niż dwie etykiety klas). Pośród zalet drzew decyzyjnych należy wymienić prostotę w budowie oraz łatwość w interpretacji, dzięki czemu można dokładnie prześledzić cały proces podejmowania decyzji i jasno określić konkretne kryteria, na bazie których zrealizowano dany podział. Ich nauka jest szybka, a poza tym radzą sobie z zarówno liczbowymi jak i kategoriowymi typami danych(newsblog.pl, 2022). Są również bardzo skuteczne w przetwarzaniu znacznych ilości danych i nie wymagają ponadprzeciętnych mocy obliczeniowych(Bujak, 2008).

Nie wolno również zapomnieć o zagrożeniu jakie ze sobą niosą. Przy ich konstrukcji należy uważać na parametry, ponieważ nadmierna głębokość drzewa, czy też zbyt wiele utworzonych gałęzi może prowadzić do przeuczenia – nadmiernego dopasowania do danych treningowych – co spowoduje nieefektywne oszacowania wartości prawdopodobieństwa zajścia zdarzenia w środowisku produkcyjnym(newsblog.pl, 2022).

Drzewa decyzyjne mają ustalony porządek(Bujak, 2008):

- korzeń odpowiada wszystkim możliwym decyzjom;
- każdy wewnętrzny węzeł odpowiada pewnej decyzji, którą możemy podjąć;
- liściom odpowiadają cele.

Algorytm przedstawiono również na przykładzie prostej decyzji kredytowej na rysunku 8:



Rysunek 8: Drzewo decyzyjne zastosowane do kategoryzacji potencjalnych kredytobiorców pod kątem ryzyka kredytowego (Bujak, 2008)

Rozwój Uczenia Maszynowego na początku XXI wieku, może dawać nadzieję na automatyzację w wielu branżach. Jesteśmy coraz bliżej powszechnego wykorzystania pojazdów autonomicznych, a od wielu lat modele ML są implementowane w dynamicznych

start-up'ach, czy też w badaniach na uczelniach ekonomiczno-technicznych, stale udowadniając jak wszechstronne mogą być zastosowania systemów uczących się. Ze względu na ich mnogość i zróżnicowanie, poprzez odpowiedni dobór metod, analitycy podnoszą skuteczność stosowanych rozwiązań, dając satysfakcję z wysokiego poziomu poprawnych predykcji uzyskanych z modeli. Stajemy się coraz bardziej zależni od wyników otrzymywanych z analiz ML, czego przykładem są modele Credit Scoring. Dziś to od maszyny zależy, czy wnioskodawca otrzyma kredyt i zrealizuje marzenie o własnym mieszkaniu, czy nowym telewizorze. Jednakże wraz ze wzrostem roli maszyn w naszym codziennym życiu, pojawia się pytanie – czy są one bezpieczne? Temat odporności systemów Uczenia Maszynowego został poruszony w kolejnym rozdziale.

Credit Scoring jest jednym z kluczowych narzędzi wykorzystywanych przez instytucje finansowe. W czasach gdy powszechny konsumpcjonizm i rozpędzone gospodarki świata generują potrzebę kreacji pieniądza, kredyt jest jedną z pierwszych rzeczy przychodzących na myśl. Zarówno wielkie korporacje, jak i osoby prywatne od czasu do czasu potrzebują znaczącego zastrzyku gotówki np. na kupno mieszkania, wymarzony samochód czy dobrze prosperującą inwestycję. Banki dziesiątki lat temu zauważyły potencjał leżący w tej dziedzinie analizy, a rozwój technologii może jeszcze bardziej zwiększyć niezawodność systemów predykcyjnych. Należy jednak pamiętać, że nawet najnowocześniejsze modele uczenia maszynowego potrzebują odpowiedniej puli wiarygodnych danych, aby najpierw skutecznie nauczyć się na błędach sprzed lat, a następnie zapobiec złym kredytobiorcom w przyszłości.

2 Adversarial Machine Learning – Wrogie Uczenie Maszynowe

Modele Uczenia Maszynowego otworzyły zupełnie nowe możliwości w dziedzinie automatyzacji, a wizja wszechobecnej Sztucznej Inteligencji skutecznie rozpala wyobrażenia ludzi o świecie, w którym na porządku dziennym będziemy wykorzystywać roboty, posiadające własną świadomość. Jednakże należy pamiętać, że z wielką mocą wiąże się wielka odpowiedzialność, a wykorzystanie nowych możliwości w złym celu, może nieść ze sobą groźne skutki. W kolejnym podrozdziale pochyłono się nad problemem Wrogiego Uczenia Maszynowego, czyli potencjalnych ataków na modele ML.

2.1 Wprowadzenie do technik Uczenia Maszynowego

Karty punktowe, mimo że łatwe w interpretacji zarówno przez wnioskujących, jak i sprzedawców, nie są najbardziej optymalnym narzędziem do oceny wiarygodności kredytowej. Przez ostatnie kilkadziesiąt lat pojawiło się wiele nowych możliwości analizy, co jest związane z nieustającym rozwojem informatyzacji, a niektóre z nich zostały dopasowane do dziedziny Credit Scoring’u, dając lepszą efektywność predykcji oraz podnosząc zyski instytucji finansowych.

Uczenie Maszynowe, samouczenie się maszyn albo systemy uczące się, w języku angielskim tłumaczone jako Machine Learning (ML) jest dziedziną wchodzącą w skład nauk, zajmujących się Sztuczną Inteligencją (Artificial Intelligence – AI), Głównym jej celem jest tworzenie automatycznego systemu, który potrafi doskonalić się na bazie doświadczenia i nabywać na tej podstawie nową wiedzę. W uproszczeniu proces polega na znalezieniu wzorca w dostarczonych danych. Modele Uczenia Maszynowego powszechnie wykorzystywane są w wielu dziedzinach, w których zachodzi potrzeba predykcji pewnego zjawiska(gov.pl, 2021).

Zadania ML ograniczone są do wąskiego, specyficznego zakresu, w którym ma działać dany system. W przeciwieństwie do sztucznej inteligencji, proces uczenia maszynowego nie jest w stanie stworzyć czegoś nowego, a jedynie uzyskiwać najbardziej optymalne rozwiązania w zadanym problemie. Najpopularniejszymi aplikacjami wykorzystującymi możliwości Uczenia Maszynowego są wyszukiwarki online, algorytmy podpowiadające najciekawsze dla użytkowników materiały w mediach społecznościowych, rozpoznawanie obrazów czy filtrowanie spamu ze skrzynek e-mail(elektronikab2b.pl, 2021).

Machine Learning stało się popularne relatywnie niedawno, ale jego historia jest znacznie dłuższa. Najważniejszy czynnik w analizie danych, czyli właśnie dane, zaczęto zbierać już w czasach starożytnych, a były to informacje o ilości zgromadzonej żywności, czy też szczegóły dot. spisów powszechnych. Kiedy z biegiem lat danych przybywało, ich analiza stawała się trudniejsza, a przełomowym momentem okazała się siedemnastowieczna epidemia dżumy, dziesiątkująca mieszkańców Anglii, gdy wówczas zaczęto publikować pierwsze zbiory danych dotyczące zdrowia publicznego. Jednakże przetwarzanie dużych zbiorów było procesem żmudnym, a jeszcze w drugiej połowie XIX wieku zebranie i przeanalizowanie danych z przeprowadzonego w Stanach Zjednoczonych spisu powszechnego zajmowało nawet dziesięć lat(fotc.com, 2022).

W latach 50. XX w. Alan Turing, znany z udziału w rozszyfrowaniu Enigmy, niemieckiej maszyny szyfrującej, stwierdził, że „jeżeli maszyna będzie w stanie przekonać człowieka, że wcale nie jest maszyną, to będzie to świadczyć o osiągnięciu przez nią sztucznej inteligencji”(fotc.com, 2022). Z kolei Artur Samuel w latach 1952-1962 rozwijał program do szkolenia zawodników szachowych, jak również on, na konferencji w 1959 roku, po raz pierwszy użył pojęcia Uczenie Maszynowe jako „[...] dające komputerom możliwość „uczenia się” bez bycia konkretnie zaprogramowanym do danego zadania.”(Mamczur, 2019).

W roku 1957 Frank Rosenblatt opracował pierwszą komputerową sieć neuronową, która wczytując obrazy, generowała etykiety i kategoryzowała ilustracje(fotc.com, 2022). Kolejnym znanym systemem był Dendral z 1965 roku, który powstał na Uniwersytecie Stanforda z inicjatywy dwóch naukowców - Edwarda Feigenbauma oraz Joshuy Lederberga. Celem programu była automatyzacja analiz i identyfikacji molekuł związków organicznych nieznanymi ówczesnym chemikom(britannica.com, 2019).

Przez kolejne dekady było dość cicho w zakresie Uczenia Maszynowego, a do lat dwudziestych stosowano je głównie w prostych grach. Następnie ML było coraz częściej wykorzystywane przede wszystkim przez przeglądarki internetowe tj. Google czy Yahoo oraz wspomagało systemy anty-spamowe. Wraz z początkiem drugiej dekady XXI wieku o Uczeniu Maszynowym ponownie zrobiło się głośno, głównie dzięki rozwojowi sieci neuronowych(Mamczur, 2019). Badania prowadzone wiele lat temu pozwoliły na to, by wiedza o ML nie była tak egzotyczna, a przeciętny student kierunku związanego z branżą informatyczną potrafił podać co najmniej kilka jej zastosowań w życiu codziennym.

Jednakże aby modele oparte na algorytmach Machine Learning'u mogły być zastosowane w biznesie muszą być odpowiednio dostosowane do zagadnienia, a co za tym idzie – nauczane na bazie wprowadzonych danych. Uczenie maszynowe nie jest jednolitą technologią, a sposób jej działania zależy w dużej mierze od tego, z jakich algorytmów korzysta i jakimi danymi zostanie zasilona.

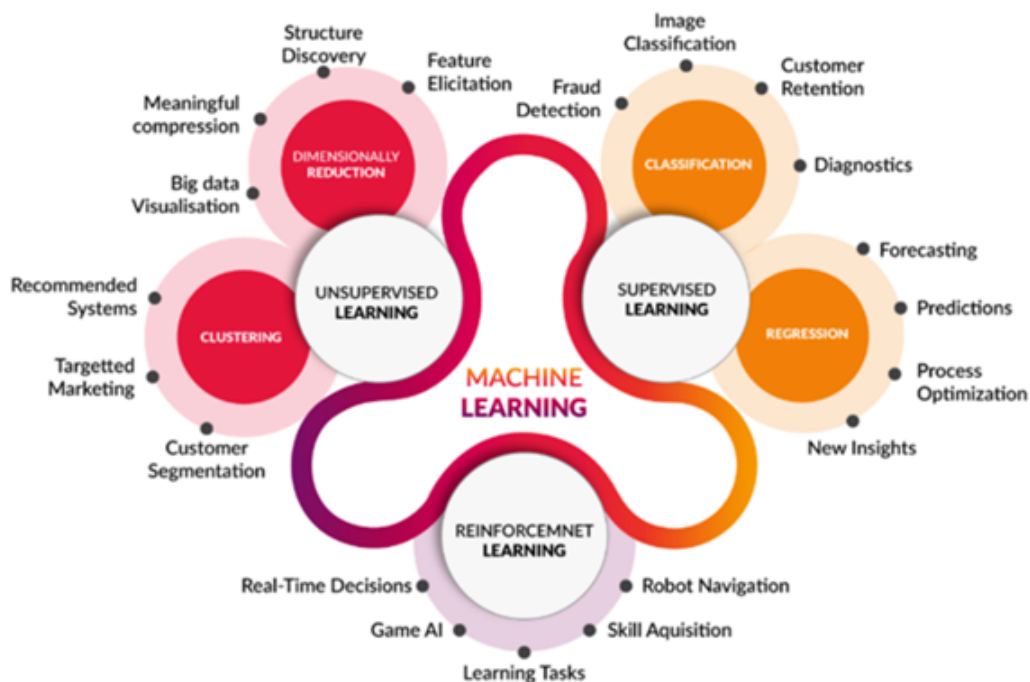
Eksperti SAS wskazują 4 podstawowe techniki uczenia maszynowego(sas.com, 2018):

- Supervised Learning (ang. Uczenie Nadzorowane) - maszyny uczą się na podstawie dostarczonych przykładów, a dane wejściowe są wykorzystywane do wyszukiwania zależności, służących do rozwiązania określonego problemu. Gdy uda się ustalić pewien wzorzec, jest on wykorzystywany w podobnych przypadkach. Do przykładowych zastosowań tej metody należą:
 - zarządzanie ryzykiem;
 - rozpoznawanie mowy, tekstu i obrazu;
 - segmentacja klientów.
- Semi - Supervised Learning (ang. Uczenie Częściowo Nadzorowane) - maszyna otrzymuje zarówno dane wejściowe oznaczone (zawierające odpowiadające im dane wyjściowe, konkretne przykłady), jak i nieoznaczone (wymagające przyporządkowania do danych wyjściowych, znalezienia odpowiedzi). Ten rodzaj uczenia wykorzystuje się w sytuacjach, gdy dana instytucja dysponuje zbyt dużą ilością danych lub gdy informacje cechują się wysokim zróżnicowaniem, które uniemożliwia przyporządkowanie odpowiedzi do każdej z nich. W takiej sytuacji system sam proponuje odpowiedzi i jest w stanie stworzyć ogólne wzorce. Metoda znajduje zastosowanie w:
 - rozpoznawaniu mowy;
 - rozpoznawaniu obrazów;
 - klasyfikacji stron internetowych.
- Unsupervised Learning (ang. Uczenie Nienadzorowane) - maszyna nie posiada „klucza odpowiedzi” i musi sama analizować dane, szukać wzorców i odnajdować relacje. Ten typ ML najbardziej przypomina sposób działania ludzkiego mózgu, który wyciąga wnioski na podstawie spontanicznej obserwacji i intuicji. Wraz ze zwiększaniem

się rozmiaru zbiorów danych prezentowane wnioski są coraz bardziej precyzyjne. Poniżej przykłady wykorzystania:

- analiza koszyka zakupowego;
 - wykrywanie anomalii;
 - rozpoznawanie podobnych obiektów.
- Reinforcement Learning (ang. Uczenie Wzmocnione) - maszyna otrzymuje gotowy zestaw dozwolonych działań, reguł i stwierdzeń oraz wykorzystuje je w taki sposób, aby osiągnąć pożądany efekt. Można to porównać do nauki gry np. w darta. Zasady określające, ile punktów musi zdobyć zawodnik oraz fakt zakończenia wartością podwójną pozostają niezmiennie. Natomiast najbardziej optymalna kombinacja punktów otrzymanych z maksymalnie trzech rzuconych lotek zależy od indywidualnej decyzji gracza. Przykłady zastosowań to:
 - nawigacja GPS (wybór trasy bazując na danych o natężeniu ruchu i pogodzie);
 - przemysł gamingowy (dopasowanie scenariuszy rozgrywki do działań gracza);
 - robotyka (dostosowanie natężenia pracy robotów do popytu).

Opisane techniki zilustrowano również na rysunku 9:



Rysunek 9: Podstawowe techniki Uczenia Maszynowego(Mamczur, 2019)

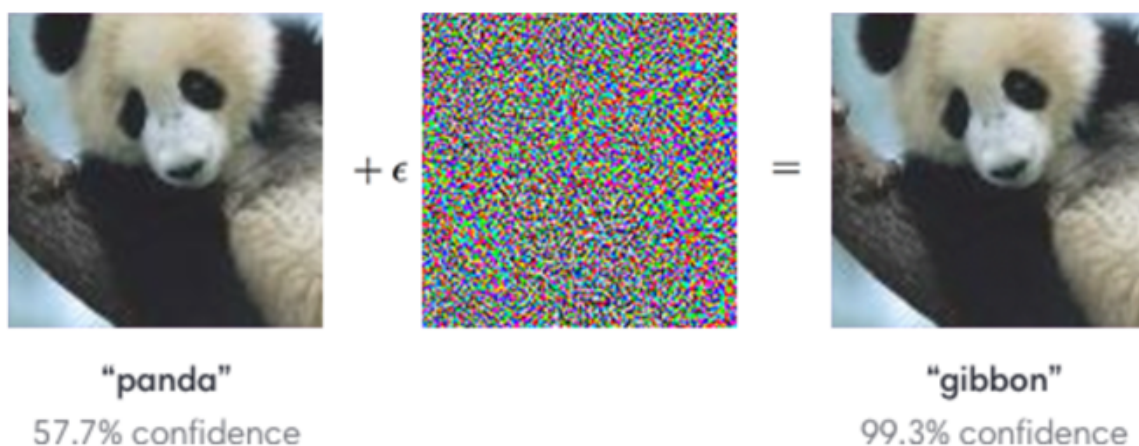
2.2 Charakterystyka Wrogiego Uczenia Maszynowego

Adversarial Machine Learning (AML), tłumaczone na język polski jako Kontradykcyjne Uczenie Maszynowe, czy też Wrogie Uczenie Maszynowe to dziedzina badań, która koncentruje się na opracowywaniu modeli ML odpornych na ataki kontradykcyjne, które są stosowane do oszukiwania lub manipulowania modelami uczenia maszynowego przez imputację złośliwych lub wprowadzających w błąd danych podczas faz uczenia lub wnioskowania. Ataki te mogą mieć poważne konsekwencje, od kradzieży poufnych informacji po nieprawidłowe działanie krytycznych systemów, takich jak samojezdne samochody lub urządzenia medyczne.

W celu dokładniejszego opisu zjawiska AML, w kolejnych akapitach odwoływano się nie do ataków na konkretne systemy Uczenia Maszynowego, lecz w ujęciu ogólnym – jako ataki na Sztuczną Inteligencję (AI). W tej tematyce wkraczamy również w dziedzinę bezpieczeństwa w cyberprzestrzeni. W sektorze cyberbezpieczeństwa AML próbuje oszukać modele poprzez tworzenie unikalnych, wprowadzających w błąd danych wejściowych, aby zmylić model, powodując jego wadliwe działanie(zephyrnet.com, 2022).

Przeciwnicy (w języku angielskim nazywani są jako „attackers”, tłumaczone na język polski również jako „napastnicy”, czy też „adwersarze”) mogą wprowadzać dane, które mają za zadanie zmanipulować rezultaty wyjściowe, wykorzystując luki w modelu. Nie jesteśmy w stanie zidentyfikować tych danych wejściowych ludzkim okiem, jednak powoduje to nieprawidłowe działanie modelu. W systemach AI występują różne formy podatności, takie jak tekst, pliki audio, obrazy. Dużo łatwiej przeprowadzać ataki cyfrowe, takie jak manipulowanie tylko jednym pikselem w danych wejściowych, co może prowadzić do błędnej klasyfikacji(zephyrnet.com, 2022).

Przykład manipulacji obrazu przedstawiono na rysunku 10. W tym przypadku atakowano system rozpoznawania zwierząt, który został nauczony na bazie pewnej puli zdjęć. Przed atakiem, model rozpoznał, że na fotografii znajduje się panda, określając to z pewnością bliską 58%. Po dodaniu szumu, zmanipulowano system do tego stopnia, iż z niemal 100% przekonaniem sklasyfikował zdjęcie pandy jako gibbona(openai.com, 2017). Łatwo zauważyć, że jednym zaburzeniem napastnik zmienił status modelu z przydatnego na bezużyteczny.



Rysunek 10: Przykład ataku na system rozpoznawania obrazów (openai.com, 2017)

O ile zmanipulowanie systemu rozpoznawania obrazów zwierząt może wydawać się niegroźne i niedostatecznie ukazywać niebezpieczeństwo płynące z tego typu ataków, o tyle wpływ napastników np. na działanie samochodów autonomicznych wskazuje na tragiczne skutki jakie mogą zostać spowodowane. Jednym z mniej skomplikowanych algorytmów ML zastosowanych w tych środkach transportu jest system rozpoznawania znaków drogowych, jako że ich liczba jest skończona i względnie nieduża, a ich kształt, kolor i rozmiar jest ściśle znormalizowany. Dla przykładu rozważmy typową sytuację drogową.

Na rysunku 11 zamieszczono widok z przedniej kamery samochodu autonomicznego, tuż przed skrzyżowaniem. Auto bez problemu rozpoznaje znak STOP, a następnie wykonuje odpowiednie czynności, aby zatrzymać się przed skrzyżowaniem. Jednakże bardzo łatwo jest wprowadzić system w błąd, co może wydarzyć się na skutek zabrudzenia znaku,



Rysunek 11: Widok z kamery przedniej samochodu autonomicznego. Właściwie rozpoznany znak STOP (Surma, 2020)



Rysunek 12: Widok z kamery przedniej samochodu autonomicznego. Niewłaściwie rozpoznany znak STOP (Surma, 2020)

czy pomalowania go farbą, czego przykład przedstawiono na rysunku 12 (Surma, 2020). Wskutek tego typu zaburzenia danych wejściowych, system oparty na Uczniu Maszynowym może nie tylko nie rozpoznać tego znaku jako nakaz zatrzymania się, a wręcz może przypisać do otrzymanego obrazu zupełnie inny znak, mający w danym momencie krytyczne znaczenie dla bezpieczeństwa kierowcy. Na rysunku 13 przedstawiono przykładową interpretację przez model ML, gdzie zabrudzony znak STOP został przyjęty jako znak pierwszeństwa (Surma, 2020).

Zakładając, że aby dotrzeć do celu kierowca na danym skrzyżowaniu musi skrócić w lewo, samochód bez zatrzymywania się przejedzie przez to skrzyżowanie. Skutki takiej decyzji z wysokim prawdopodobieństwem będą tragiczne, co ukazuje jak duże znaczenie



Rysunek 13: Widok z kamery przedniej samochodu autonomicznego. Zakładając, że aby dotrzeć do celu kierowca na danym skrzyżowaniu musi skrócić w lewo, samochód bez zatrzymywania się przejedzie przez to skrzyżowanie. Skutki takiej decyzji z wysokim prawdopodobieństwem będą tragiczne, co ukazuje jak duże znaczenie

ma jakość danych dostarczanych do modelu i jak niewielkie zaburzenie może wpłynąć na jego niezawodność, a co za tym idzie, uzasadnienie dalszego wykorzystania zaprojektowanego narzędzia w biznesie(Surma, 2020).

Jednym z głównych czynników blokujących wykorzystanie Sztucznej Inteligencji w coraz to istotniejszych aspektach życia jest jej wrażliwość na wrogą ingerencję. Opisany powyżej przypadek obrazuje problemy z jakimi spotykają się współcześni modelarze systemów uczących się. Wyzwaniem najbliższych lat jest uczynienie tego typu narzędzi bezpiecznymi dla codziennego użytku. Aktualnie wciąż trwa wypracowywanie najodpowiedniejszych technik wzmacniania niezawodności ML, jak również definiowane są tzw. dobre praktyki, będące tymczasową odpowiedzią na niemoc badaczy w niektórych obszarach.

Badacze z instytutu naukowo-badawczego w Albuquerque jako cel obrali sobie stworzenie w pełni wiarygodnego i skutecznego sposobu na obronę przed wrogą ingerencją w modele Uczenia Maszynowego. Aby tego dokonać, przeprowadzili szereg badań nad wieloma zbudowanymi przez nich sieciami neuronowymi ukierunkowanymi na typowe dla tej tematyki rozpoznawanie obrazów. W pierwszym kroku wytrenowali modele próbując osiągnąć możliwie najwyższą skuteczność. Kolejnym etapem było obniżanie ich jakości poprzez "zatrucie" danych stosowanych do uczenia tychże modeli. Finalnie porównywano stopień degradacji skuteczności klasyfikatorów(Short, Pay, & Gandhi, 2019).

W celu podniesienia odporności modeli, stosuje się trening kontradyktoryjny, polegający na wprowadzaniu do modelu mylących danych, mających za zadanie wprowadzenie algorytmu w błąd, jednakże tego typu informacje, wprowadzone na etapie uczenia systemu, cechują się potencjałem do zmniejszania jego wrażliwości na ataki. Zatem można przyjąć, że poza technikami defensywnymi, równie szerokim zagadnieniem jest tworzenie strategii ataku. W literaturze znajdowane są odniesienia do kilku głównych metod kreacji wrogich danych. Niektóre z nich to(Short et al., 2019):

- Fast Gradient Sign Method - celem trenowania modelu jest minimalizacja tzw. funkcji błędu, natomiast poprzez wprowadzenie FGSM zwiększana jest wartość tejże funkcji;
- Basic Iterative Method - metoda będąca wprost rozwinięciem techniki FGSM, polegająca na wielokrotnym, aczkolwiek jasno wcześniej zdefiniowanym, jej powtórzeniu;

- Metoda Carliniego Wagnera (C&W) - technika generowania wrogich danych, skupiająca się na maksymalizacji podobieństwa między oryginalnymi informacjami wejściowymi, a zmanipulowanymi, zachowując przy tym efekt błędnej klasyfikacji przez model.

Naukowcy z Albuquerque wskazują wysoki potencjał zauważony przez nich w metodzie C&W. Mimo wielu prób użycia odpowiedniej taktyki defensywnej, badacze nie byli w stanie zastosować skutecznej obrony przed wrogimi danymi wygenerowanymi w ten sposób (Short et al., 2019), co podkreśla jak duże pole do dalszych badań istnieje w dziedzinie AML.

2.3 Typy ataków na algorytmy Uczenia Maszynowego

Na daną chwilę większość algorytmów proponowanych przez badaczy, naukowców i specjalistów z branży R&D skupia się głównie na wysokiej wydajności i niskiej liczbie błędnych klasyfikacji. Jednakże nawet kiedy wskazane cele zostają osiągnięte, modele te często nie powinny być implementowane w środowiskach produkcyjnych, zwłaszcza w domenach krytycznych, czy aspektach życia, które mogą mieć wpływ na życie znacznej części społeczeństwa, nie uwzględniając innych kryteriów i wymagań dotyczących sztucznej inteligencji. Są nimi: bezpieczeństwo algorytmów, ich interpretowalność i uczciwość. Co więcej, rezultaty osiągane są na danych, które są odpowiednio przygotowane w warunkach laboratoryjnych i możliwe jedynie w sytuacji gdy implementacja również zachodzi w takich warunkach (Pawlicki, 2020).

Zastosowanie Sztucznej Inteligencji na wielką skalę stało się rzeczywistością, za czym idzie świadomość, że bezpieczeństwo samych algorytmów Uczenia Maszynowego wymaga natychmiastowej uwagi. Adwersarze potrafią starannie dobrać próbki danych wejściowych, aby zmieniało to wyniki klasyfikacji lub regresji w oczekiwany przez nich sposób. Świadomość zagrożeń związanych z ich użyciem, a także ich podatność na AML jest jeszcze całkiem niewielka (Pawlicki, 2020), jednakże już powstały definicje określające poszczególne typy ataków.

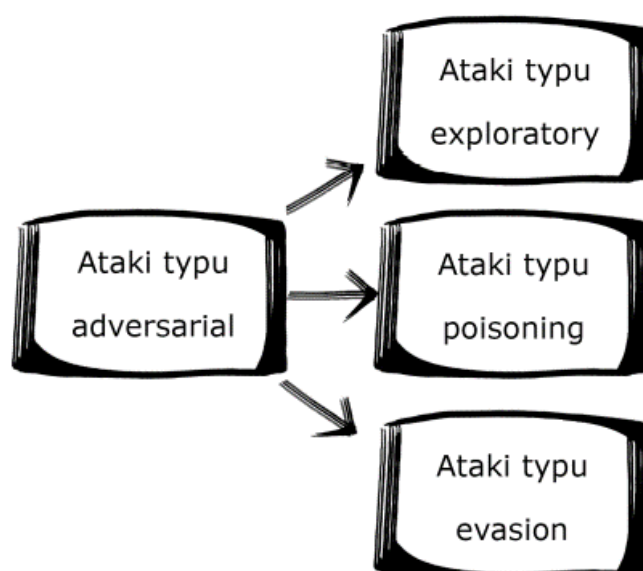
Jednym z najbardziej znanych podziałów jest klasyfikacja ze względu na dostęp do modelu (zephyrnet.com, 2022):

- Atak białoskrzynkowy - odnosi się do sytuacji, w której atakujący ma pełny dostęp

do modelu docelowego. Obejmuje to architekturę i parametry, które pozwalają im tworzyć próbki danych na modelu docelowym. Osoby atakujące będą miały ten dostęp tylko wtedy, gdy testują model jako programista. Mają oni detaliczną wiedzę na temat architektury sieci oraz znają tajniki modelu i tworzą strategię ataku;

- Atak czarnoskrzynkowy - odnosi się do sytuacji, w której atakujący nie ma dostępu do modelu docelowego i może jedynie zbadać dane wyjściowe.

Podział ataków na czarnoskrzynkowe i białoskrzynkowe oparty jest o umiejscowienie atakującego. Inna klasyfikacja bazuje na strategii ingerencji w model, a jej schemat przedstawiono na rysunku 14. Atak zatruwający (typu poisoning) koncentruje się na danych ze



Rysunek 14: Nowe ataki na Uczenie Maszynowe (Pawlicki, 2020)

zbioru uczącego. Tutaj atakujący zmienia istniejące lub wprowadza nieprawidłowo oznakowane dane. Wskutek takiego działania, model przeszkolony na „zatrutym” zbiorze będzie tworzył błędne predykcje na prawidłowo oznakowanych danych(towardsdatascience.com, 2021). W literaturze znaleźć można kilka artykułów na temat ataków tego typu. W jednym z nich, autorzy opisują użycie tzw. wrogiego szumu etykiet (ang. adversarial label noise). W tym artykule przedstawiona jest metoda wykorzystująca sposób działania algorytmu SVM (Support Vector Machines – ang. Metoda Wektorów Nośnych), którego działanie polega na mapowaniu danych na wielowymiarową przestrzeń właściwości w sposób umożliwiający kategoryzację punktów danych(ibm.com, 2021).

Ogólnym założeniem ataku jest wprowadzenie do zbioru treningowego próbki, która

znacząco zmieni wynik klasyfikacji, obniżając skuteczność modelu. Taką próbkę można stworzyć poprzez rozwiązanie problemu optymalizacyjnego, polegającego na wyszukiwaniu lokalnych maksimów powierzchni funkcji błędu, do czego wykorzystano algorytm gradient ascent. Atak wykorzystuje odwracanie etykiet konkretnych próbek w klasyfikacji binarnej zbioru uczącego, przy założeniu, że dane w zbiorze walidacyjnym nie są w żaden sposób zmieniane(Biggio, Nelson, & Laskov, 2012).

Ataki unikowe (typu evasion), w odróżnieniu od ataków zatruwających, nie skupiają się na danych używanych do uczenia modelu, lecz na odpowiedniej manipulacji danymi wejściowymi, dla których model wydaje prognozowany rezultat. Polegają one na modyfikowaniu danych, aby wydawały się uzasadnione, lecz by prowadziły do błędnej prognozy. Przykładem wykorzystania tego typu ataków są modele oceny wiarygodności kredytowej. Ubiegając się o kredyt, osoba atakująca może zamaskować swój prawdziwy kraj pochodzenia za pomocą usługi VPN, ukrywając w ten sposób np. iż jest obywatelem kraju uznawanego przez model jako bardziej ryzykowny, co mogłoby zmniejszyć jego szanse na pozytywną ocenę jego wniosku(towardsdatascience.com, 2021).

Innym kierunkiem wykorzystania ataków unikowych są modele służące do odfiltrowywania wiadomości e-mail będących spamem. Ich podejście może polegać na eksperymentowaniu z mailami, które model już wytrenował w zakresie sprawdzania i rozpoznawania jako spam. Jeśli model został wyszkolony do filtrowania wiadomości e-mail zawierających konkretne słowa, atakujący może tworzyć nowe e-maile zawierające powiązane z tym słowa, które przejdą przez algorytm, co spowoduje, że wiadomość e-mail, która w typowym procesie zostałaby sklasyfikowana jako spam, spamem nie jest, co w oczywisty sposób pogarsza skuteczność modelu(zephyrnet.com, 2022).

Atak poszukiwawczy (typu exploratory) może wystąpić po wytrenowaniu algorytmu, a jego zadaniem jest odkrywanie informacji o wewnętrznym działaniu modelu, w celu identyfikacji słabych punktów. W tym podejściu ingerencja jest ukierunkowana na poszukiwanie informacji o(Shi, Sagduyu, & Grushin, 2017):

- granicy decyzyjnej używanej przez algorytm (np. hiperpłaszczyzny maszyny wektorów nośnych (SVM) algorytm);
- ogólnym zestawie reguł, którymi kieruje się algorytm;

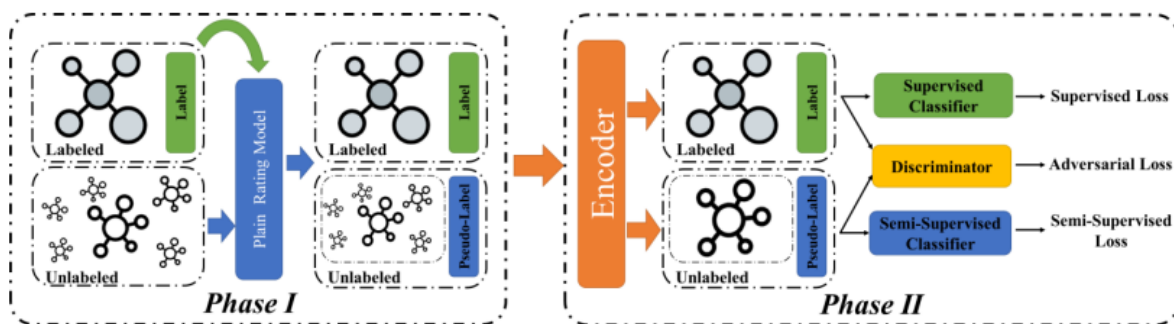
- zestawie logicznych lub probabilistycznych właściwości algorytmu;
- danych, które zostały wykorzystane (lub nie wykorzystane) do uczenia algorytmu.

W ostatnich latach powstało kilka prac, badających ataki na Głęboką Sieć Neuronową (DNN). W artykule z 2017 roku (Shi et al., 2017) naukowcy zbudowali tzw. funkcjonalny ekwiwalent klasyfikatorów modelu DNN, opierając się na algorytmach SVM oraz naiwnego klasyfikatora Bayes’a. Z kolei w badaniu z 2019 roku (Shi, Sagduyu, Davaslioglu, & Li, 2019) opisano jak przy pomocy Głębokiej Sieci Neuronowej można wydobyć klasyfikator wdrożony w warunkach produkcyjnych. Podsumowując - celem ataku typu exploratory jest budowa lokalnego klasyfikatora (Pawlicki, 2020).

Jak zatem zabezpieczyć się przed wrogimi ingerencjami w model? Jedną z możliwości stanowi tzw. trening kontradyktoryjny, będący jednym z podstawowych podejść do poprawy wydajności i bezpieczeństwa ML. Polega on na kontrolowanym atakowaniu modelu na wiele sposobów, znajdując w ten sposób „czułe punkty” zastosowanego rozwiązania. Na skutek takich badań, osoby budujące model mogą zweryfikować jego odporność na wrogie ataki, a następnie podjąć odpowiednie kroki celem poprawy jego bezpieczeństwa (zephyrnet.com, 2022).

Chińscy badacze postanowili wykorzystać zjawisko Adversarial Machine Learning do wyznaczania korporacyjnych rating’ów kredytowych. Dotychczas wykorzystanie standardowych metod predykcyjnych nie przynosiło satysfakcjonujących rezultatów, a ma na to wpływ specyfika rynku. Zatrudnienie i opłacenie zespołu specjalistów, których zadaniem będzie oszacowanie stopnia wiarygodności firmy nie jest małym wydatkiem i mogą sobie na nie pozwolić jedynie duże korporacje. Implikuje to względnie niedużą próbę obserwacji i możliwość ich uzasadnionego wykorzystania jedynie dla podobnych instytucji, jak te posiadające swój rating. Większość średnich i praktycznie wszystkie małe przedsiębiorstwa nie są w stanie otrzymać własnego rate’u, a co za tym idzie, potencjalny model nie ma na czym nauczyć się, a potem przewidywać wiarygodność tych firm (Feng & Xue, 2021).

Naukowcy zaproponowali kilkietapowy proces uczenia modelu, zobrazowany na rysunku 15, rozpoczynający się od podzielenia danych na oznaczone (ang. labeled data) - firmy ze znanym rating’iem, a także nieoznaczone (ang. unlabeled data) - firmy bez rating’u. Na danych oznaczonych nauczono model predykcyjny gradientowo wzmocnionego drzewa decyzyjnego, który następnie wykorzystano do wyznaczenia najbardziej prawdopo-



Rysunek 15: Proces uczenia według ASSL4CCR (Feng B., 2021)

dobnych rating'ów dla obserwacji nieoznaczonych. W ten sposób otrzymano pełen zestaw niewybrakowanych danych, jednakże kluczowym elementem tego etapu było zakwalifikowanie ocen wyliczonych przez model jako 'pseudooceń' - rate'y z ograniczonym do nich zaufaniem(Feng & Xue, 2021).

Proces ten można porównać do nauki ucznia w szkole. W trakcie lekcji, otrzymuje on zestaw zadań i poprawnych do nich odpowiedzi, które dostarcza nauczyciel, a zatem można w pełni zaufać, że są one zgodne z rzeczywistością. Jednakże podczas sprawdzianu, gdy pojawią się zadania nieco inne niż podczas zajęć, podejrzenie odpowiedzi u kolegi daje informację o tym jak do danego pytania ustosunkowała się inna osoba, ale nie musi to oznaczać, że rozwiązanie "ściągnięte" od sąsiada jest zgodne z rzeczywistością, więc trzeba je traktować z ograniczonym zaufaniem(Feng & Xue, 2021).

| Model | Recall | Accuracy | F1-score |
|-----------------|----------------|----------------|----------------|
| LR | 0.76250 | 0.80970 | 0.81946 |
| SVM | 0.83750 | 0.89247 | 0.88961 |
| MLP | 0.91406 | 0.93568 | 0.93245 |
| Xgboost | 0.92343 | 0.94225 | 0.94133 |
| CCR-CNN | 0.92812 | 0.95253 | 0.94518 |
| CCR-GNN | 0.93437 | 0.95012 | 0.95177 |
| ASSL4CCR | 0.95321 | 0.96115 | 0.96252 |

Tabela 3: Klasyczna karta oceny punktowej(Feng B., 2021)

W drugim etapie na przygotowanych danych zastosowano podejście z kategorii AML, polegające na ograniczeniu zaufania do danych wypredykowanych z obserwacji nieoznaczonych, dzięki czemu mogą one brać udział w analizie. Na podstawie zbiorów danych oznaczonych oraz nieoznaczonych obliczono klasyfikator nadzorowany i klasyfikator częściowo-nadzorowany, a dodatkowo wyznaczono kontradyktoryjną stratę między zbiorami co po-

zwoiliło na zastosowanie obydwu zbiorów jako całości i wyznaczenie modelu opisanego jako ASSL4CCR - Kontradiktoryjne Uczenie Częściowo Nadzorowane dla Korporacyjnego Rating'u Kredytowego(Feng & Xue, 2021).

Chińscy naukowcy, jako zwieńczenie projektu, zrealizowali porównanie podstawowych parametrów pozwalających na obiektywną ocenę jakości poszczególnych modeli tj. recall, dokładność oraz F1-score. Z analizy wyników przedstawionych w tabeli 3 wynika, że model ASSL4CCR osiąga najlepsze rezultaty, co dowodzi przydatności świadomego wykorzystania AML w praktyce(Feng & Xue, 2021).

2.4 Przykłady ataków na algorytmy Uczenia Maszynowego

Adversarial Machine Learning jest względnie nową poddziedziną Analizy Danych i na tę chwilę nie łatwo jest znaleźć wzięte z życia przykłady ingerencji w rzeczywiste, używane produkcyjnie modele ML. Dzięki wysokiej świadomości analityków nie trzeba uczyć się na nieświadomie popełnionych błędach następnie wykorzystanych przez adversarzy, a problem został zidentyfikowany zanim pojawiły się jego potencjalnie fatalne skutki. W związku z powyższym, w ostatnich latach, naukowcy prowadzą intensywne badania, samodzielnie generując różne scenariusze ingerencji w model, jednakże wciąż pozostaje w tej dziedzinie wiele przestrzeni dla nowych odkryć.

W ostatnich latach prowadzone jest coraz więcej badań na modelach wykorzystujących Uczenie Maszynowe pod kątem poprawienia ich odporności na zamierzone, bądź też przypadkowe zaburzenia i ataki. Ze względu na względnie wysoką podatność oraz szeroką gamę praktycznych zastosowań najbardziej interesującym, a zarazem najczęściej podejmowanym obiektem rozważań są sieci neuronowe i głębokie sieci neuronowe(Papernot, McDaniel, Goodfellow, Jha, & Celik, 2017).

Większość ciekawych wniosków dotyczy prac w tematyce rozpoznawania obrazów, a konkretnie uodparniania algorytmów na potencjalne ataki. Na łamach cytowanej pracy, naukowcy z Uniwersytetu stanowego Pensylwanii oraz Uniwersytetu w Wisconsin, przy współpracy z reprezentantem firmy OpenAI, która najbardziej znana jest ze stworzenia ChatuGPT(openai.com, 2023), zajęli się praktycznymi atakami na ogólnodostępne modele głębokich sieci neuronowych przy pomocy tzw. ataków czarnokrzynkowych (ten jak i również pozostałe typy ataków opisano w podrozdziale 2.3). Badacze zrealizowali serię



Rysunek 16: Przykłady obrazów wykorzystanych w ataku na model głębokiej sieci neuronowej firmy MetaMind(Papernot et al., 2017)

testów na oprogramowaniu firmy MetaMind, obserwując odpowiedzi modelu na wprowadzane przez nich obrazy. Następnie wygenerowali tzw. Wrogie Przykłady (ang. Adversarial Examples), nazywane dalej Wrogimi Próbkami, które cechują się niezauważalnymi lub trudno zauważalnymi dla człowieka różnicami, w przypadku plików graficznych na poziomie manipulacji pojedynczych pikseli. Na rysunku 16 wskazano przykładowe obrazy wykorzystane w eksperymencie. W górnym wierszu przedstawiono poprawnie sklasyfikowane przykłady, natomiast niżej zaprezentowano przygotowane i błędnie ocenione przez model Wrogie Próbki(Papernot et al., 2017).

Wyniki badań wskazywały, iż naukowcy są w stanie zaburzyć klasyfikacje generowane przez głęboką sieć neuronową w 84,24% przypadków. Jeszcze więcej do myślenia dają kolejne kroki opisane przez naukowców. Stosując to samo podejście, przetestowali skuteczność klasyfikatora zaimplementowane w dostępnych online rozwiązaniach firm Amazon oraz Google. Zarejestrowano błędne klasyfikacje modeli w odpowiednio 96,19% oraz 88,94% przypadków(Papernot et al., 2017).

Obecnie lwia część społeczeństwa posiada skrzynkę mailową, a coraz częściej jedna osoba posiada kilka takich komunikatorów. Łatwość tworzenia nowych kont i docierania do innych użytkowników na masową skalę, wciąż stanowi bardzo popularny sposób nie tylko do porozumiewania się w ważnych sprawach tj. komunikacja służbowa czy też potwierdzenia różnych transakcji internetowych, lecz także daje możliwość przesyłania różnych ofert i reklam. Wśród takich wiadomości łatwo jest umieszczać niebezpieczne linki, kie-

rujące nieświadomych odbiorców do domen stanowiących zagrożenie dla ich prywatności. Według raportu Message Labs Intelligence pośród całej globalnej komunikacji mailowej, wiadomości typu spam stanowią aż 88%(Kuchipudi, Nannapaneni, & Liao, 2020). W celu uniknięcia podejrzanych wiadomości oraz zalewaniu skrzynki przez nachalną propagandę reklamową stosuje się tzw. filtry antyspamowe oparte na algorytmach uczenia maszynowego.

Algorytmy uczone są wykrywania ”wrogich”słów w zawartości e-maili. W zależności od modelu różne słowa w zróżnicowanym stopniu obniżają lub podnoszą poziom zaufania modelu do danej wiadomości, a po przeprowadzeniu oceny podejmowana jest decyzja o umieszczeniu jej w skrzynce odbiorczej lub folderze spam. Na polu filtracji niechcianych wiadomości toczona jest ciągła walka między deweloperami, a adversarzami mającymi na celu oszukać model, aby dokonał błędnej klasyfikacji(Cheng, Xu, Li, & Ding, 2022).

Jednym ze sposobów wykorzystywanych przez napastników jest zaszcycie w wiadomości dostatecznie wielu silnie zaufanych słów, które przeważą ocenę klasyfikatora z negatywnej na pozytywną. Jednakże umieszczenie wielu słów z bardzo wąskiej i zróżnicowanej grupy może skutecznie wypaczyć przekaz wiadomości do tego stopnia, że będzie ona łatwo wykrywalna nawet dla ludzkiego oka, a potencjalna ofiara nie da się nabrać na atak. Pomysłów adversarzy na uniknięcie takiej sytuacji jest wiele i odwołują się one głównie ich kreatywności, jak np. wpisanie kilkudziesięciu zaufanych słów w zawartość maila, stosując jednocześnie dla nich kolor czcionki identyczny z barwą tła, co skutecznie ukrywa podejrzaną zawartość zarówno przed użytkownikiem, jak i algorytmem, czy też umieszczenie literówek w słowach znacznie obniżających prognozowaną przez algorytm ocenę takiej zawartości, dzięki czemu uzyskują one wagę neutralną i nie są wychwytywane przez filtry jako niebezpieczne(Cheng et al., 2022).

Zespół naukowców z Uniwersytetu Michigan postanowił przeprowadzić badania mające na celu sprawdzić skuteczność ataków typu AML na algorytmy anty-spamowe. Wykorzystali do tego celu trzy względnie proste techniki(Kuchipudi et al., 2020):

- synonym replacement (ang. podmiana słowa przy pomocy synonimu);
- ham word injection (ang. wprowadzenie zaufanych słów);
- spam word spacing (ang. wprowadzenie przerw/spacji pomiędzy literami w słowach

związanych ze spamem).

Modelem, którego skuteczność do obrony badano, był algorytm naiwnego klasyfikatora Bayesa. Badacze argumentują wybór właśnie tej techniki implementacji filtra antyspamowego jego popularnością w tego typu zastosowaniach, co empirycznie wykazało jego wysoką skuteczność w wychwytywaniu wrogich, bądź też niechcianych wiadomości (Kuchipudi et al., 2020).

Naukowcy na łamach pracy wskazują również potencjalne trudności, jakie mogą wystąpić przy próbie tego typu ataków. Główną przeszkodą może być "czarnoskrzynkowość", czyli ograniczony lub całkowicie uniemożliwiony dostęp do informacji na temat wybranego i nauczonego modelu filtracji antyspamowej, jak również nieznaną do danych wejściowych użytych do jego budowy. Jednakże warto zauważyć, że badania wykazują, iż do przeprowadzenia udanego ataku na algorytmy uczenia maszynowego filtrów spam, często wystarczy znajomość zaledwie 1% danych treningowych, co znacznie ułatwia zadanie adversarzy (Kuchipudi et al., 2020).

| Modified Message | Cosine Similarity | Prediction |
|--|-------------------|------------|
| Ringtone Club: acquire the UK single graph on your Mobile_River each hebdomad and take any top_side caliber ringtone! This content is free_people of charge. | 0.583 | spam |
| Ringtone Club: become the UK bingle graph on your nomadic each workweek and select any upper_side caliber ringtone! This subject_matter is liberate of charge. | 0.583 | spam |
| Ringtone Club: go the UK one graph on your peregrine each calendar_week and pick_out any upside character ringtone! This substance is release of charge. | 0.583 | ham |

Tabela 4: Wrogie próbki zastosowane na atakowanym modelu filtra antyspamowego (Kuchipudi et al., 2020)

W pierwszym podejściu, zastosowano synonimy słów, wychwytywanych przez filtr jako niebezpieczne, nie zmieniając przy tym sensu samej wiadomości. Wykorzystano do tego celu technikę NLP - Natural Language Processing(ang. przetwarzanie języka natural-

nego) - będącą poddziedziną Sztucznej Inteligencji i odpowiedzialną za rozumienie języka ludzkiego przez maszyny i roboty(Castagno, 2020). Na łamach pracy, jako przykład przetworzenia wiadomości niebezpiecznej zmanipulowanej w sposób pozwalający przejść przez filtrację podano zdanie: "Ringtone Club: Get the UK singles chart on your mobile each week and choose any top quality ringtone! This message is free of charge." Różnie zmodyfikowane wiadomości wraz z rezultatem ich klasyfikacji przez model antyspamowy przedstawiono w tabeli 4(Kuchipudi et al., 2020). Łatwo zauważyć, że wiadomość, która została przepuszczona przez klasyfikator jako zaufana, zapewne nie zostałaby potraktowana poważnie przez użytkownika, co wskazuje na wdrożenie niezbędnych poprawek w zestawach synonimów, wykluczając te mające wpływ na kontekst informacji.

Drugie podejście opiera się na manipulacji częstotliwością pojawiania się słów zaufanych. Wśród publicznie dostępnych zbiorów można bez trudu znaleźć zestawy słów znanych jako słowa związane ze zjawiskiem spamu i z wysokim prawdopodobieństwem generujące uruchomienie filtra. W związku z powyższym, jako zaufane należy traktować wyrazy, które nie znajdują się w tym zbiorze, dzięki czemu wprowadzenie ich do zawartości wiadomości podnosi prawdopodobieństwa przejścia przez filtrację(Kuchipudi et al., 2020).

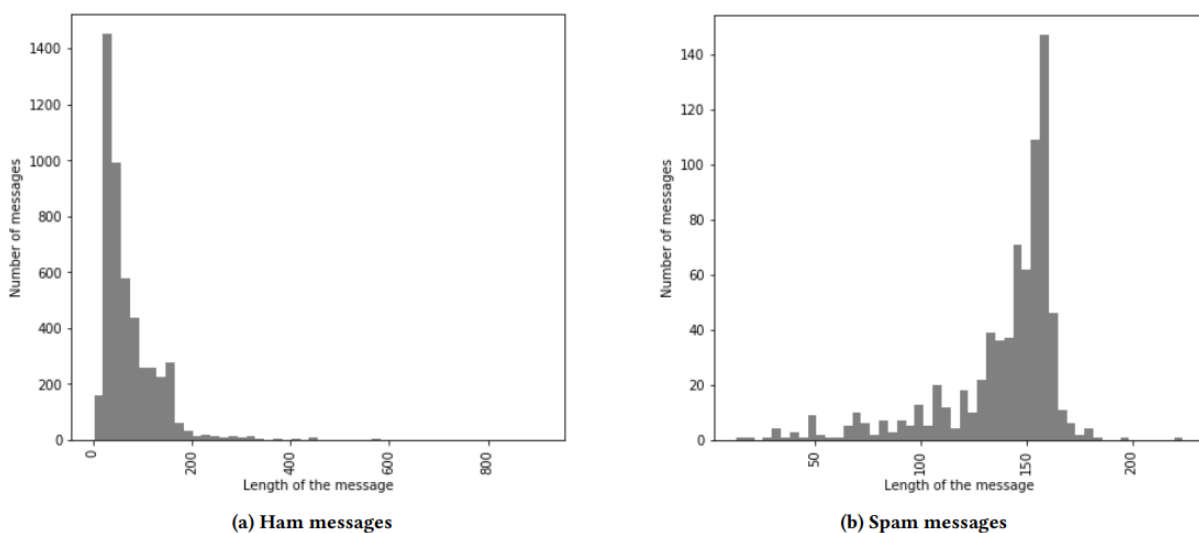
Przykładowa wiadomość klasyfikowana początkowo jako spam: "Congratulations ur awarded 500 of CD vouchers or 125gift guaranteed and Free entry 2 100 wkly draw txt MUSIC to 87066 TnCs www.Ldew.com1win150ppmx3age16", po wprowadzeniu kilkukrotnie słów zaufanych tj. good, love, deal, jest w stanie przejść przez filtr jako zaufana: "Congratulations good ur awarded good 500 of CD vouchers or 125 good gift guaranteed love and Free entry 2 good 100 wkly draw txt MUSIC to 87066 TnCs www.Ldew.com1win150ppmx3age16 good good good good good deal". Ponownie można zastanawiać się nad sensem samej wiadomości, jednakże nie było to obiektem rozważań cytowanej pracy (Kuchipudi et al., 2020).

Podczas badań tego podejścia zauważono pewną właściwość filtrów antyspamowych. Z testów można wnioskować, że model jest wyczulony na używanie skrótów w mailach tj. użycie "U" zamiast "you" (ang. Ty) czy też "R" w miejsce "are" (ang. jesteś/jest/jesteśmy/jesteście/są). Wynika z tego, że stosowanie bardziej zadbanego i oficjalnego języka tekstu daje większą szansę na ominięcie filtracji i skuteczne umieszczenie potencjalnie wrogiej

wiadomości w atakowanej skrzynce odbiorczej(Kuchipudi et al., 2020).

Trzecia technika polegała na wprowadzeniu odstępów między znakami w słowach prawdopodobnie klasyfikowanych jako niebezpieczne. Wysoki współczynnik podobieństwa, przy pomocy którego badano podobieństwo znaczeniowe wiadomości oryginalnej i zmanipulowanej, wskazuje iż w tym przypadku treść jest najmniej zaburzona względem odniesienia i możliwie najlepiej oddaje sens zamierzony przez autora. W przytoczonym przez badaczy przykładzie edytowanie słów "sexy" i "flirt" do form "s e x y" oraz "f l i r t" skutkuje oszukaniem modelu i uznaniem wiadomości niebezpiecznej za zaufaną (Kuchipudi et al., 2020).

W podsumowaniu pracy można znaleźć konkluzję, iż przy wykorzystaniu trzech opisanych powyżej sposobów potrafiąco w około 60 % przypadków oszukać filtr antyspamowy. W trakcie badań wykorzystano zbiór danych z witryny Kaggle, zawierający 5572 wiadomości, z czego 747 zostały oznaczone jako spam. Wielkość zestawu jest wystarczająca do budowy takiego mechanizmu, jednakże zdecydowanie pewniejszy model można byłoby uzyskać przy zebraniu nieco większej próbki do jego nauki, co przyczyniłoby się do zebrania bardziej wiarygodnych wniosków(Kuchipudi et al., 2020).



Rysunek 17: Dwa histogramy obrazujące rozkład liczby znaków wiadomości w zależności od liczby znaków dla treści spam oraz tekstów zaufanych(Kuchipudi et al., 2020)

Ciekawe spostrzeżenie, mogące mieć duży wpływ na skuteczność wykrywania niebezpiecznych maili, płynie z analizy długości(liczby znaków) w wiadomościach. Zauważono, że treści zaufane zawierają zwykle poniżej 150 znaków, gdzie w przypadku spamu obser-

wuje się koncentrację wokół tejże liczby. Zobrazowano to na rysunku 17, jednakże należy zwrócić uwagę na oś poziomą, która w skutek różnej skali dla obydwu wykresów może nieco zakłamywać skalę zróżnicowania obserwowanych wartości(Kuchipudi et al., 2020).

Na łamach pierwszych dwóch rozdziałów opisano tematykę Credit Scoringu, jak również Wrogiego Uczenia Maszynowego. Rozdziały 3 i 4 stanowią będą praktyczne połączenie tych dwóch zagadnień.

3 Budowa modelu drzewa decyzyjnego do celu Credit Scoring

3.1 Analiza oraz wstępne przetworzenie wybranego zbioru danych

Wybór zbioru danych jest jednym z najtrudniejszych, a jednocześnie najważniejszych etapów budowy modelu predykcyjnego. To na tym zestawie informacji algorytm decyzyjny uczy się kluczowych właściwości, które dają mu możliwość odpowiedniej klasyfikacji. W przypadku dziedziny Credit Scoring, uzyskanie dostępu do obszernych zbiorów rzeczywistych danych jest często niemożliwe. Właścicielami tego typu zestawów informacji najczęściej są banki, które niechętnie dzielą się takimi starannie utworzonymi zbiorami, bądź też argumentują swoją odmowę wrażliwością tychże danych oraz restrykcyjnymi zasadami narzuconymi przez Nadzorcę (w przypadku Polski mowa o Komisji Nadzoru Finansowego).

W przypadku zestawów dostępnych na publicznych witrynach internetowych jak np. Kaggle.com, można na nich zbudować model predykcyjny, jednakże należy zadać sobie pytanie odnośnie jego rzetelności i skuteczności. Często zebrane tam zbiory nie są dostatecznie duże jako całość, cechują się niską zawartością "default'ów", bądź też są to dane symulacyjne i nierzeczywiste. Dzięki życzliwości Pana dr Karola Przanowskiego, pełniącego rolę nauczyciela akademickiego w Zakładzie Metod Statystycznych i Analiz Biznesowych na Szkole Głównej Handlowej w Warszawie, który udzielił zgody na wykorzystanie dużego i rzetelnego zbioru na potrzeby tej pracy. Rozważany zestaw danych wykorzystywany jest do celów dydaktycznych np. do budowy karty scoringowej w ramach przedmiotu "Credit Scoring - automatyzacja procesu biznesowego".

Zbiór zawiera 219 zmiennych, zarówno ciągłych, jak i kategorycznych. Umieszczono w nim 68499 obserwacji, gdzie każda z nich dotyczy innej aplikacji o produkt gotówkowy lub ratalny. Tak duży zbiór uniemożliwia pełne opisanie każdej ze znajdujących się w nim zmiennych, zatem szerzej przeanalizowano jedynie potencjalnie najważniejsze z nich lub opisano grupy tychże zmiennych. Pełną dokumentację wszystkich zmiennych zamieszczono w pliku Excel VariablesDescription.xlsx jako załącznik do pracy, natomiast wspomniany opis, wraz z przykładowymi wartościami przedstawiono w tabeli 5:

| Nazwa zmiennej | Opis | Przykładowe wartości |
|-------------------------|--|---|
| cid | Identyfikator klienta | 0000024576 |
| aid | Identyfikator rachunku/wniosku kredytowego | css1970022600002 |
| product | Typ produktu, o który aplikuje klient | 'css', 'ins' |
| period | Sześciocyfrowa liczba wskazująca na moment aplikacji o dany produkt, gdzie pierwsze cztery cyfry oznaczają rok, a pozostałe dwie informują o miesiącu | 197304 |
| act_... | Zmienne bieżące, zebrane z niezależnego źródła (tj. BIK) opisujące sytuację klienta w momencie aplikowania o produkt | wiek, suma zaciągniętych zobowiązań, liczba spłaconych kredytów |
| act_age | Wiek klienta w latach | 54 |
| act_cus_active | Zmienna binarna informująca o tym, że klient miesiąc temu miał aktywną pożyczkę (był w trakcie jej spłacania) | '1' lub brak danych |
| act_ccss_cc | Zdolność kredytowa klienta określana stosunkiem sumy raty i wydatków do przychodów | 0.45 |
| app_... | Zmienne aplikacyjne, deklarowane przez klienta w trakcie składania aplikacji | płeć, miejsce zamieszkania, stan cywilny |
| app_income | Zarobki klienta | 1512 |
| app_spendings | Wydatki klienta | 500 |
| app_char_cars | Informacja czy klient posiada auto | 'Owner', 'No' |
| agr_... | Zmienne behawioralne, wyliczane jako agregaty z 3, 6, 9, lub 12 ostatnich miesięcy, bez uwzględniania braków danych | minimalna, maksymalna oraz średnia liczba dni spóźnienia w spłacie kredytu ratalnego w ciągu ostatnich 9 miesięcy |
| ags_... | Zmienne behawioralne, wyliczane jako agregaty z 3, 6, 9, lub 12 ostatnich miesięcy, z uwzględnieniem braków danych | minimalna, maksymalna oraz średnia liczba dni spóźnienia w spłacie kredytu gotówkowego w ciągu ostatnich 6 miesięcy |
| default3, default6, ... | Binarna zmienna celu informująca o zdarzeniu wejścia klienta w opóźnienie w spłacie równe co najmniej 90 dni w ciągu x miesięcy od zaciągnięcia zobowiązania | '1' lub '0' |

Tabela 5: Opis zmiennych oraz grup zmiennych w zbiorze danych (źródło opracowanie własne)

Realizowanie kodu rozpoczęto od wczytania zbioru danych, którym jest plik SAS-owy, o rozszerzeniu sas7bdat. Zastosowano funkcję read_sas z biblioteki pandas. Odpowiadającą tej operacji fragment kodu jest widoczny na rysunku 18:

```
In [98]: 1 source_data = pd.read_sas('abt_app.sas7bdat', encoding = 'LATIN2')
        2 data = source_data.copy()
        3 print("Zaimportowany zbiór danych wraz z jego rozmiarem:", data.shape)
        4 data.head(5)
```

Zaimportowany zbiór danych wraz z jego rozmiarem: (68499, 219)

Out[98]:

| | cid | aid | product | period | act_age | act_cc | act_loaninc | app_income |
|---|------------|------------------|---------|--------|---------|----------|-------------|------------|
| 0 | 0000000001 | css1970020500001 | css | 197002 | 59.0 | 0.492428 | 2.442599 | 2047.0 |
| 1 | 0000000060 | css1970020700004 | css | 197002 | 52.0 | 0.447902 | 1.023541 | 4885.0 |
| 2 | 0000000013 | css1970021100003 | css | 197002 | 38.0 | 0.586847 | 8.431703 | 593.0 |
| 3 | 0000000089 | css1970022400005 | css | 197002 | 37.0 | 0.165007 | 2.370792 | 2109.0 |
| 4 | 0000000005 | css1970022600002 | css | 197002 | 46.0 | 0.529964 | 1.805054 | 2770.0 |

Rysunek 18: Fragment kodu odpowiadający za import zbioru danych (źródło opracowanie własne)

W zbiorze danych znajdują się informacje o wielu klientach, którzy wielokrotnie aplikują o dwa różne produkty - kredyt gotówkowy oraz kredyt ratalny. Różnią się one nie tylko pożyczanymi kwotami, ale przede wszystkim, mają innych klientów docelowych. Kredyty ratalne związane są np. z kupnem lodówki, czy laptopa, a banki najczęściej nie zarabiają na tych produktach, oferując zerowe rzeczywiste stopy procentowe. Taka operacja ma na celu pozyskanie klienta celem zebrania informacji takich jak zmienne behawioralne. W przypadku, gdy klient sumiennie spłacał zobowiązanie w odpowiednim czasie, taka informacja jest odnotowywana, a kredytobiorca staje się celem dla ofert kredytów gotówkowych. W przeciwnym przypadku, klient staje się niewiarygodny, a taka informacja również zawiera się w bazie danych pożyczkobiorców instytucji finansowych. Banki zarabiają głównie na "gotówce", natomiast kredyty ratalne w najlepszym przypadku nie przynoszą strat.

```
1 data_css = data.loc[data['product'] == 'css'].copy()
2 print("Rozmiar zbioru danych dla produktu 'css':", data_css.shape)
```

Rozmiar zbioru danych dla produktu 'css': (34188, 219)

Rysunek 19: Fragment kodu odpowiadający za ograniczenie zaimportowanego zbioru danych do produktu gotówkowego (źródło opracowanie własne)

W związku z powyższym, do dalszej analizy należało wybrać jeden z tych produktów.

Opierając się na potrzebie realizowania wyższych zysków, wybrano kredyt gotówkowy, co zrealizowano przy pomocy kodu z rysunku 19, czego efektem było zmniejszenie liczby obserwacji do 34188.

W opracowywanym zbiorze zbadano rozkład zmiennej binarnej 'default12', reprezentującej wystąpienie zdarzenia spóźnienie się ze spłacaniem zobowiązania sumarycznie przez co najmniej 90 dni w trakcie pierwszych 12 miesięcy od wypłacenia kredytu, która została wybrana jako modelowana zmienna celu i zdefiniowana jako parametr globalny 'target_variable' używając kodu z rysunku 20.

```
1 target_variable = 'default12'
```

Rysunek 20: Fragment kodu odpowiadający za definicję zmiennej celu jako parametr globalny (źródło opracowanie własne)

Podczas analizy rozkładu wartości zmiennej default12 wychwycono 4411 obserwacji z brakami danych w zmiennej celu. Liczbowy rozkład wartości zbadano przy pomocy kodu z rysunku 21.

```
1 print("Rozkład wartości zmiennej celu w zbiorze danych dla produktu 'css':")
2 print(data_css[target_variable].value_counts())
3 print("Liczba braków danych zmiennej celu w zbiorze danych dla produktu 'css':",
4       data_css[target_variable].isnull().sum())
```

Rozkład wartości zmiennej celu w zbiorze danych dla produktu 'css':
1.0 19986
0.0 9791
Name: default12, dtype: int64
Liczba braków danych zmiennej celu w zbiorze danych dla produktu 'css': 4411

Rysunek 21: Fragment kodu odpowiadający za analizę liczbową rozkładu wartości zmiennej celu (źródło opracowanie własne)

Ze względu na dostatecznie dużą liczbę obserwacji bez braków danych, podjęto decyzję o nie stosowaniu technik imputacji danych i usunięcie wybrakowanych tychże danych z dalszych rozważań, co zrealizowano kodem z rysunku 22. Dodatkowo posortowano zbiór rosnąco według zmiennej 'aid'. W zestawie danych pozostało 29777 obserwacji.

```
1 data_css.dropna(subset = [target_variable], inplace = True)
2 data_css.sort_values(by = ['aid'], inplace = True)
3 print(" Rozmiar zbioru danych dla produktu 'css'\n",
4       "po wyeliminowaniu obserwacji z brakami danych w tej zmiennej:",
5       data_css.shape)
```

Rozmiar zbioru danych dla produktu 'css'
po wyeliminowaniu obserwacji z brakami danych w tej zmiennej: (29777, 219)

Rysunek 22: Fragment kodu odpowiadający za usunięcie braków danych zlokalizowanych w zmiennej celu, oraz posortowanie zbioru (źródło opracowanie własne)

Jako zmienne objaśniające rozpatrywano te posiadające przedrostki 'app', 'act', 'agr' lub 'ags', co zrealizowano przy pomocy kodu zaprezentowanego na rysunku 23.

```
1 variables = [variable for variable in list(data_css)
2               if any(variable.casefold().startswith(prefix) for prefix in ['app', 'act', 'agr', 'ags'])]
3 print('Liczba zmiennych spełniających warunek określonego przedrostka =', len(variables))
```

Liczba zmiennych spełniających warunek określonego przedrostka = 201

Rysunek 23: Fragment kodu odpowiadający za wyfiltrowanie wybranych zmiennych objaśniających (źródło opracowanie własne)

W ramach wstępnej analizy należało zapoznać się również z typem zmiennych, ponieważ analiza danych jakościowych znacząco różni się od podejścia w przypadku danych ilościowych. Stosując kod z rysunku 24 dokonano analizy liczby tychże zmiennych, z podziałem ze względu na ich typ.

```
1 variables_char_list = list(data_css[variables].select_dtypes(include = 'object'))
2 variables_num_list = list(data_css[variables].select_dtypes(include = 'number'))
3 print('Liczba zmiennych jakościowych =', len(variables_char_list))
4 print('Liczba zmiennych ilościowych =', len(variables_num_list))
```

Liczba zmiennych jakościowych = 7
Liczba zmiennych ilościowych = 194

Rysunek 24: Fragment kodu odpowiadający za wstępną analizę typów wybranych zmiennych objaśniających (źródło opracowanie własne)

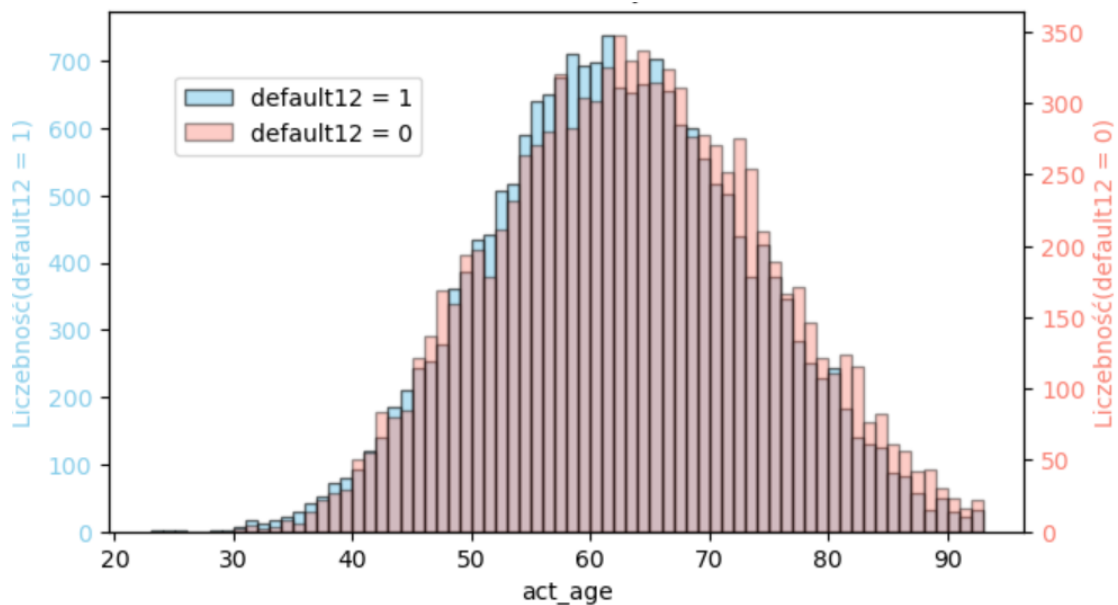
W podrozdziale 3.2, jako jeden z etapów budowy modelu, opisano realizację modelu wstępnego, celem wybrania potencjalnie najciekawszych zmiennych, na których można zaprezentować przykładową analizę eksploracyjną. Wyboru zmiennych dokonano na podstawie statystyki Feature Importance (ang. ważność cechy), której wynik zaprezentowano na rysunku 37. Sugerując się wynikami tego wskaźnika, wybrano trzy najistotniejsze zmienne we wstępnym modelu i przeprowadzono ich analizę.

Analizę eksploracyjną rozpoczęto od podziału przetworzonego zbioru danych na dwa podzbiory - 'data0', zawierający obserwacje z zmienną celu równą '0' oraz 'data1', gdzie 'default12' jest równe '1'. Przy pomocy funkcji 'describe()' uzyskano podstawowe statystyki. Jako pierwszą przeanalizowano zmienną 'act_age', wskazującą wiek klienta w momencie składania wniosku, która została zakwalifikowana przez model wstępny jako najważniejsza. Wyniki analizy przedstawiono w tabeli 6, na podstawie której można zauważyć, że podzbiór klientów stwarzających wyższe ryzyko cechuje się niższym wiekiem w chwili aplikacji, co widać porównując wartości poszczególnych centyli oraz średnich.

| Statystyka | default12 = 0 | default12 = 1 |
|------------------------|---------------|---------------|
| Liczba obserwacji | 9791 | 19986 |
| Liczba braków danych | 0 | 0 |
| Średnia | 63.028904 | 61.882468 |
| Odchylenie standardowe | 11.340882 | 11.004611 |
| Wartość minimalna | 30 | 23 |
| Centyl 25% | 55 | 54 |
| Centyl 50% | 63 | 62 |
| Centyl 75% | 71 | 69 |
| Wartość maksymalna | 93 | 93 |

Tabela 6: Podstawowe statystyki zmiennej 'act_age' (źródło opracowanie własne)

Celem wizualizacji rozkładu liczebności zmiennej 'act_age' w zależności od wartości zmiennej celu, wygenerowano histogram przedstawiony na rysunku 25. Mimo, iż model wstępny wskazał istotność wieku w predykcji 'default12', wykres nie wskazuje wyraźnych różnic pomiędzy podgrupami. Przy dokładnej analizie można zauważyć delikatne przesunięcie się w lewo rozkładu dla 'default12' = 1, co znajduje swoje potwierdzenie w średniej z tabeli 6. W kodzie źródłowym można znaleźć również wizualizacje przy pomocy wykresów wiolinowych oraz pudełkowych, jednakże różnice są na nich równie niewielkie, stąd decyzja o nie umieszczaniu ich w pracy.



Rysunek 25: Rozkład zmiennej 'act_age' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne)

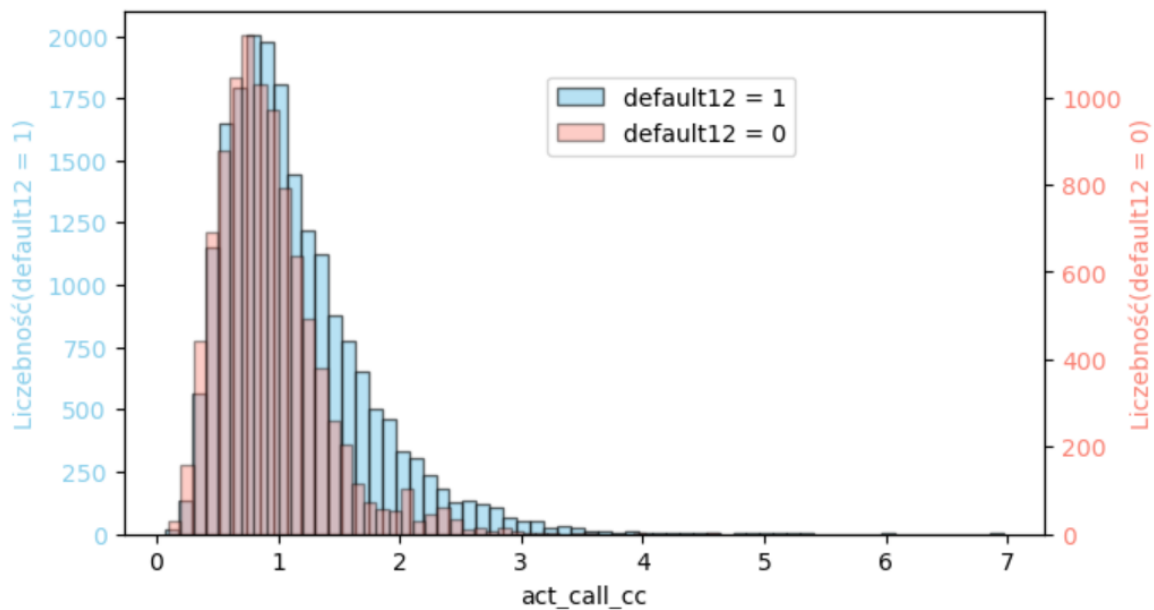
Jako drugą przeanalizowano zmienną act_call_cc, oznaczającą sumę potencjalnych rat kredytu i wydatków w stosunku do dochodów klienta. Z tabeli 7 wynika, iż istnieje różnica

między dobrymi, a złymi klientami, możliwa do zauważenia na podstawie względnego obciążenia finansowego klientów wynikającego z otrzymania nowego produktu kredytowego. Prawie każda ze statystyk (oprócz wartości minimalnej) cechuje się wyższą wartością dla zmiennej 'default12' równej 1.

| Statystyka | default12 = 0 | default12 = 1 |
|------------------------|---------------|---------------|
| Liczba obserwacji | 9791 | 19986 |
| Liczba braków danych | 0 | 0 |
| Średnia | 0.925345 | 1.141037 |
| Odchylenie standardowe | 0.436998 | 0.590710 |
| Wartość minimalna | 0.099325 | 0.075747 |
| Centyl 25% | 0.628913 | 0.721041 |
| Centyl 50% | 0.851278 | 1.010427 |
| Centyl 75% | 1.126885 | 1.423545 |
| Wartość maksymalna | 4.622150 | 6.964356 |

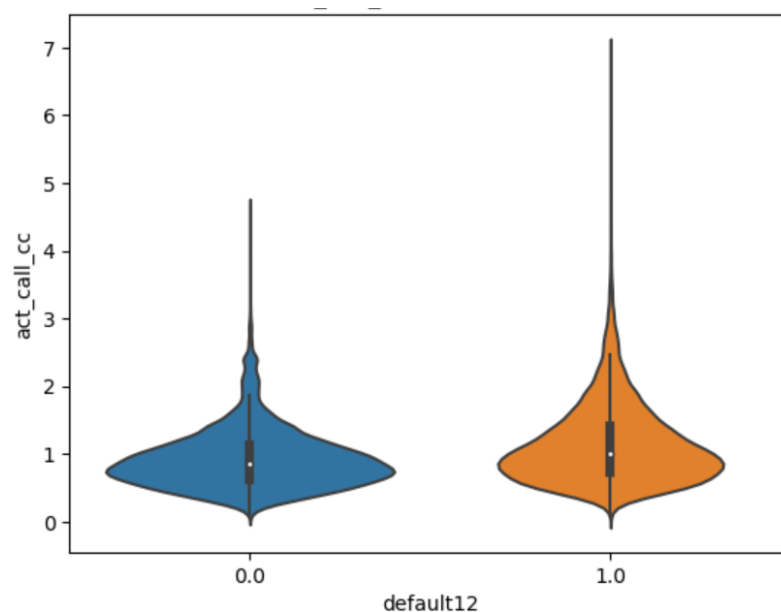
Tabela 7: Podstawowe statystyki zmiennej 'act_call_cc' (źródło opracowanie własne)

Histogram widoczny na rysunku 26 wykazuje liczniejsze wystąpienia klientów z wyższą wartością zmiennej 'act_call_cc' wnioskodawców z podgrupy 'default12' = 1.

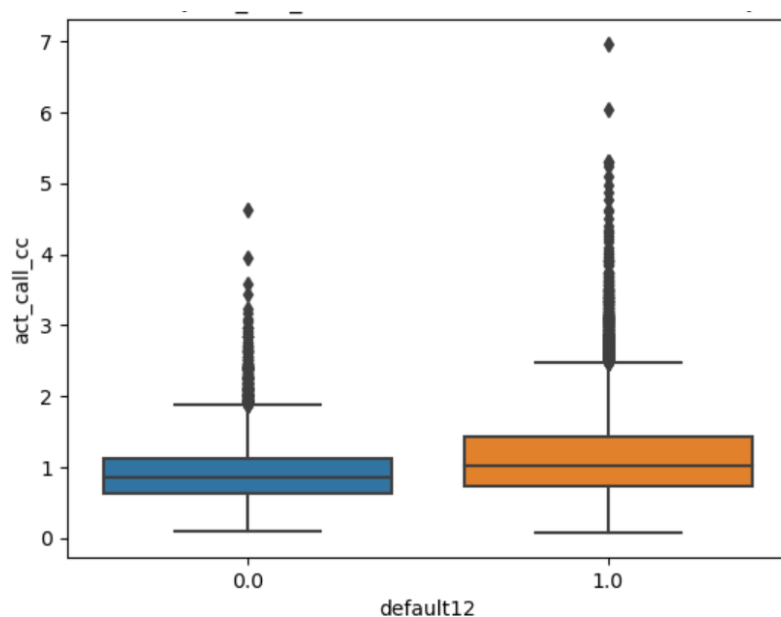


Rysunek 26: Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne)

Ciekawe wizualizacje występującego zjawiska udało się osiągnąć na wykresach - wioli-nowym, na rysunku 27 oraz pudełkowym, na rysunku 28.



Rysunek 27: Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' na wykresie wiolinowym (źródło opracowanie własne)



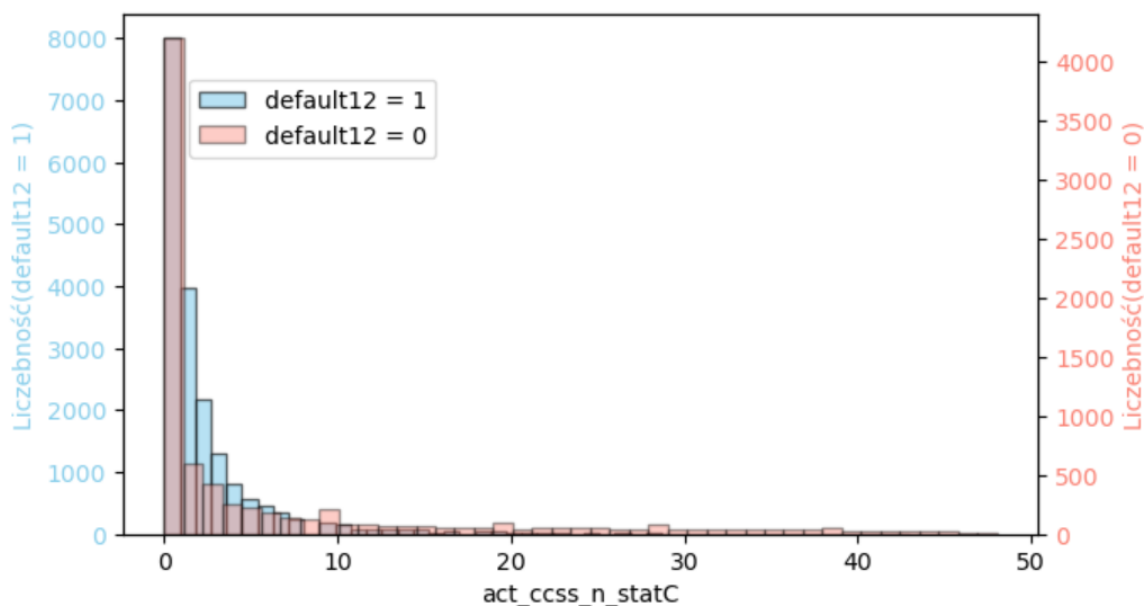
Rysunek 28: Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' na wykresie pudełkowym (źródło opracowanie własne)

Analizę eksploracyjną zakończono na trzeciej najważniejszej zmiennej, według modelu wstępnego, którą była 'act_ccss_n_statC', oznaczająca liczbę poprawnie w pełni spłaconych zobowiązań klienta w momencie składania wniosku kredytowego. W tabeli 8 ponownie można zauważyć, iż istnieje różnica pomiędzy dobrymi, a złymi klientami, możliwa do wywnioskowania na podstawie pozytywnej przeszłości kredytowej klienta.

| Statystyka | default12 = 0 | default12 = 1 |
|------------------------|---------------|---------------|
| Liczba obserwacji | 7889 | 18894 |
| Liczba braków danych | 1902 | 1092 |
| Średnia | 6.196096 | 2.110194 |
| Odchylenie standardowe | 10.407138 | 3.654164 |
| Wartość minimalna | 0 | 0 |
| Centyl 25% | 0 | 0 |
| Centyl 50% | 1 | 1 |
| Centyl 75% | 48 | 3 |
| Wartość maksymalna | 48 | 43 |

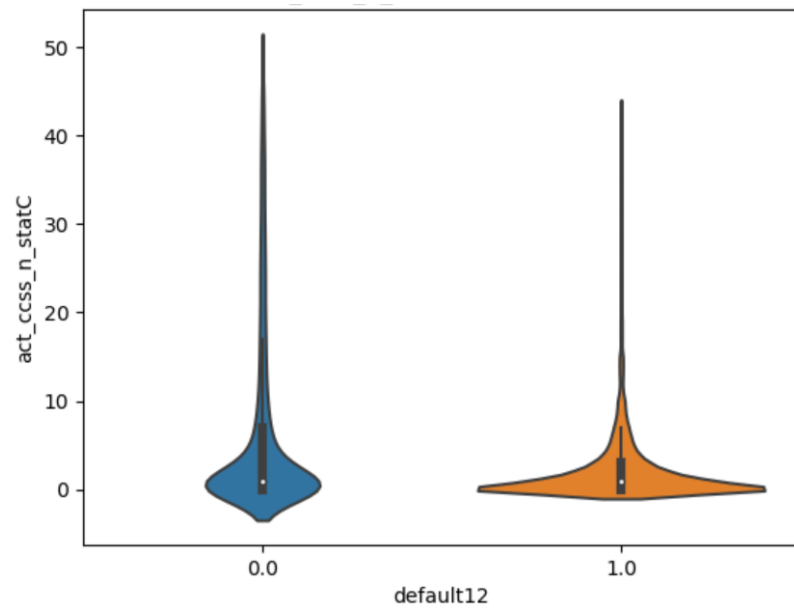
Tabela 8: Podstawowe statystyki zmiennej 'act_ccss_n_statC' (źródło opracowanie własne)

Histogram widoczny na rysunku 29 wykazuje koncentrację większości klientów blisko wartości 0 oraz 1. Warto zwrócić uwagę na sposób skalowania osi y, jako że dla obydwu podgrup zastosowane są różne skale, co może dawać złudzenie większego podobieństwa rozkładów niż ma to miejsce w rzeczywistości.

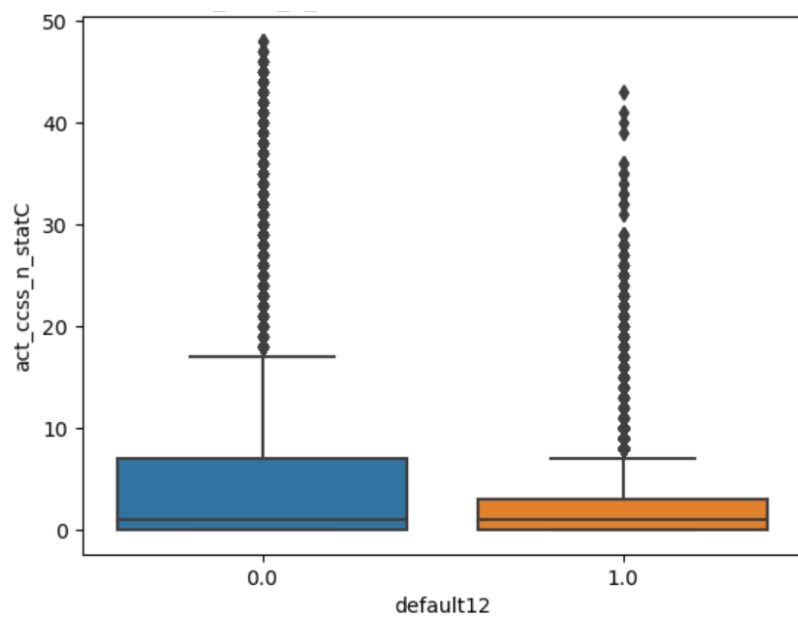


Rysunek 29: Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne)

Warto zwrócić uwagę na wykres wiolinowy, na rysunku 30. Kształty dla obydwu podgrup zupełnie się różnią. Wydłużenie w okolicach wartości 0 utwierdza w przekonaniu o korelacji małego pozytywnego doświadczenia kredytowego w przypadku złych klientów. Natomiast wizualizacja pudełkowa z rysunku 31, dodatkowo zwraca uwagę na szerokie spektrum wartości przyjmowanych przez zmienną 'act_ccss_n_statC'.



Rysunek 30: Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' na wykresie wiolinowym (źródło opracowanie własne)



Rysunek 31: Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' na wykresie pudełkowym (źródło opracowanie własne)

Po zapoznaniu się z charakterystyką wybranego zbioru danych, przetworzeniu go do celów budowy modelu oraz analizie eksploracyjnej najważniejszych zmiennych, przystąpiono do budowy modelu Uczenia Maszynowego wykrywającego potencjalnie nierzetelnych klientów wnioskujących o gotówkowy produkt kredytowy.

3.2 Wybór narzędzi, budowa modelu i analiza wyników

Etap budowy modelu rozpoczęto od określenia odpowiednich prerekwizytów. W części programistycznej skorzystano z języka python w wersji 3.11.1. Rolę edytora kodu oraz silnika obliczeniowego pełniło popularne oprogramowanie Jupyter. Jako model predykcyjny wybrano drzewo decyzyjne, jednakże w odpowiedzi na rozwój rozwiązań drzewiastych, a w szczególności poprawę skuteczności decyzyjnej będącej skutem ich zastosowań, postanowiono skorzystać z modelu XBoost (ang. Extreme Gradient Boosting), nazywanego także wzmocnionym drzewem decyzyjnym.

XGBoost to algorytm Uczenia Maszynowego, który zyskał dużą popularność zarówno w środowisku akademickim, jak i przemysłowym, ze względu na swoją wysoką wydajność w różnych zadaniach związanych z analizą danych. Jest to metoda tzw. uczenia zespołowego, która łączy zalety wzmocniania gradientowego i technik regularyzacji, aby tworzyć bardzo dokładne i efektywne modele predykcyjne. Jest dobrze przystosowany do danych tablicowych i skutecznie radzi sobie z brakami danych. XGBoost wyróżnia się w zadaniach takich jak klasyfikacja, czy regresja(datascience.eu, 2019).

Główną siłą tego algorytmu jest jego zdolność do obsługi złożonych interakcji między cechami. Wykorzystuje on wzmocnianie gradientowe, iteracyjnie poprawiając kolejne drzewa decyzyjne poprzez podniesienie ich skuteczności, jednocześnie minimalizując określoną funkcję straty. Aby dalej poprawić uogólnienie modelu i uniknąć przeuczenia, XGBoost wykorzystuje techniki regularyzacji, takie jak regularyzacja L1 i L2. Wysoka skalowalność i możliwość równoległego przetwarzania sprawiają, że XGBoost dobrze sprawdza się w pracy z dużymi zbiorami danych(datascience.eu, 2019).

Celem skorzystania z algorytmu XGBoost, należało na wstępie zainstalować w środowisku programistycznym pakiet o tej samej nazwie. Dodatkowo zainstalowano także pakiet 'category_encoders', co wykonano poleceniami z rysunku 32.

```
1 !pip install xgboost
2 !pip install category_encoders
```

Rysunek 32: Fragment kodu odpowiadający za instalację niezbędnych pakietów (źródło opracowanie własne)

Pakiet 'category_encoders' to zbiór narzędzi, który pozwala na łatwą i skuteczną transformację danych katégoricznych na liczbowe, aby mogły być używane w modelach Uczenia Maszynowego. Oferuje różnorodne techniki kodowania, takie jak kodowanie one-hot, kodowanie binarne, target encoding, czy kodowanie Helmerta(scikit learn.org, 2022).

Poza instalacją pakietów, niezbędne było również zaimportowanie kilku bibliotek, co wykonano przy pomocy kodu z rysunku 33.

```
1 import pandas as pd
2 import category_encoders as ce
3 from sklearn.model_selection import train_test_split
4 import xgboost as xgb
5 from sklearn.metrics import roc_auc_score, roc_curve
6 import matplotlib.pyplot as plt
7 import seaborn as sns
```

Rysunek 33: Fragment kodu odpowiadający za import niezbędnych bibliotek (źródło opracowanie własne)

Pierwszą z bibliotek jest 'pandas' - otwarta biblioteka Pythona zaprojektowana do manipulacji i analizy danych. Zapewnia struktury takie jak DataFrame i Series, umożliwiające efektywne zarządzanie danymi tabelarycznymi. 'Pandas' ułatwia zadania związane z czyszczeniem, przekształcaniem, filtrowaniem i łączeniem danych. Wspiera wiele formatów wejściowych, takich jak CSV, Excel, czy też bazy danych SQL. 'Pandas' łączy się dobrze z innymi bibliotekami Pythona, co czyni go popularnym wyborem do celów analizy danych(pydata.org, 2023).

O ile funkcjonalność bibliotek 'category_encoders' oraz 'xgboost' jest wprost związana z opisanymi wcześniej pakietami, to dwie funkcjonalności importowane z biblioteki 'sklearn' wymagają chwili uwagi. Funkcja 'train_test_split' służy do podziału danych na zbiory treningowe i testowe w celu oceny modelu. Funkcja losowo przemieszcza i dzieli dane wejściowe na dwie podgrupy, zgodnie z określonymi przez użytkownika proporcjami, który może również losowość tego podziału za pomocą parametru 'random_state'. Jej wszechstronność i łatwość użycia czynią ją niezbędnym narzędziem w pracach związanych z narzędziami predykcyjnymi(scikit learn.org, 2023d).

Kolejną zastosowaną w budowie modelu funkcją z biblioteki sklearn jest 'roc_auc_score', która przeznaczona jest do oceny wydajności modeli klasyfikacji binarnej. Oblicza obszar pod krzywą ROC, a wynik mieści się w przedziale od 0 do 1, gdzie 1 oznacza doskonały kla-

syfikator. Funkcja przyjmuje prawdziwe etykiety binarne oraz przewidywane prawdopodobieństwa jako argumenty i oblicza wynik. Jest szczególnie przydatna przy pracy z niezerównoważonymi danymi i stanowi wiarygodną miarę do porównywania różnych modeli (scikit learn.org, 2023b). Inną niezbędną funkcją z biblioteki sklearn jest 'roc_curve', która służy do generowania krzywych ROC dla modeli klasyfikacji binarnej (scikit learn.org, 2023c). Ostatnią funkcją z tejże biblioteki jest 'accuracy_score', która służy do oceny dokładności modeli klasyfikacyjnych, a realizuje to obliczając stosunek poprawnie przewidzianych zdarzeń do ogólnej liczby zdarzeń w zbiorze danych. Dostarcza prosty sposób na ocenę ogólnej wydajności modelu pod względem poprawnych klasyfikacji (scikit learn.org, 2023a).

W celu wizualizacji wyników zaimportowano moduł 'pyplot' z biblioteki 'matplotlib', który daje możliwość tworzenia różnych typów wykresów, diagramów i wizualizacji. Umożliwia kreowanie figur, dodawanie osi, rysowanie punktów danych, dostosowywanie elementów wizualnych (np. kolorów, znaczników i stylów linii) oraz dodawanie etykiet, tytułów i legend do wykresów. Jest wszechstronny i może być używany do tworzenia wykresów liniowych, punktowych, słupkowych, histogramów i wielu innych. Jest szeroko stosowany w analizie danych, badaniach naukowych i wizualizacji danych (matplotlib.org, 2022).

Zaimportowano również bibliotekę 'seaborn', która służy do tworzenia wykresów statystycznych. 'Seaborn' jest przydatny do wizualizacji skomplikowanych zbiorów danych za pomocą niewielkiej ilości kodu, co sprawia, że jest popularnym wyborem wśród analityków. Oferuje wykresy punktowe, słupkowe, heatmapy, czy wykresy wiolinowe. Biblioteka posiada także motywy i palety kolorów, które poprawiają estetykę wykresów (seaborn.pydata.org, 2022). Ostatnią zastosowaną biblioteką jest 'math', udostępniająca funkcje matematyczne oraz stałe do różnych obliczeń, w tym operacji arytmetycznych, trygonometrii, czy logarytmów (python.org, 2023).

Mając dostęp do wyżej wymienionych narzędzi, rozpoczęto proces budowy modelu. Skorzystano z przetworzonego zbioru 'data_css' oraz list zmiennych jakościowych i ilościowych przygotowanych kodem z rysunku 24. Model XGBoost nie jest w stanie operować na zmiennych jakościowych, co wymusza konieczność zastosowania kategoryzacji tychże zmiennych. W tym celu skorzystano z funkcji 'BinaryEncoder' pakietu 'category_encoders' realizującej kodowanie binarne. 7 zmiennych jakościowych zostało poddane temu procesowi, na skutek czego powstało 16 nowych zmiennych binarnych. Wynikiem operacji

przedstawionej na rysunku 34 jest zbiór zbinowanych zmiennych 'data_encoded', natomiast lista ich nazw została zapisana jako 'variables_char_encoded_list'.

```
1 encoding = ce.BinaryEncoder(cols = variables_char_list)
2 data_encoded = encoding.fit_transform(data_css[variables_char_list])
3 variables_char_encoded_list = list(data_encoded)
4 print("Lista", len(variables_char_encoded_list), "zbinowanych zmiennych jakościowych:")
5 variables_char_encoded_list
```

Lista 16 zbinowanych zmiennych jakościowych:

```
['app_char_branch_0',
 'app_char_gender_0',
 'app_char_gender_1',
 'app_char_job_code_0',
 'app_char_job_code_1',
 'app_char_job_code_2',
 'app_char_marital_status_0',
 'app_char_marital_status_1',
 'app_char_marital_status_2',
 'app_char_city_0',
 'app_char_city_1',
 'app_char_city_2',
 'app_char_home_status_0',
 'app_char_home_status_1',
 'app_char_cars_0',
 'app_char_cars_1']
```

Rysunek 34: Fragment kodu realizujący binowanie zmiennych jakościowych (źródło opracowanie własne)

W kolejnym kroku dokonano konkatencji zbiorów 'data_css' oraz 'data_encoded' tworząc ramkę danych 'data_full'. Utworzono również dwie nowe listy nazw zmiennych - 'variables_full_wo_target', zawierająca jedynie zmienne numeryczne oraz zmienne jakościowe binowane; 'variables_full', składająca się dodatkowo ze zmiennej celu, zmiennej czasowej oraz identyfikatora. Następnie zrealizowano podział finalnego zbioru na podzbiory treningowy i testowy przy pomocy funkcji 'train_test_split'. Wymiary podzielonych zbiorów wraz z kodem realizującym to działanie zaprezentowano na rysunku 35.

```
1 train, test = train_test_split(data_full[variables_full], random_state = 2001, test_size = 0.3)
2 print("Rozmiar zbioru treningowego:", train.shape)
3 print("Rozmiar zbioru testowego:", test.shape)
```

Rozmiar zbioru treningowego: (20843, 213)

Rozmiar zbioru testowego: (8934, 213)

Rysunek 35: Fragment kodu realizujący podział na zbiory treningowy i testowy (źródło opracowanie własne)

Ostatnią operacją niezbędną do rozpoczęcia tworzenia algorytmu XGBoost i testów jego wyników, było utworzenie ściśle określonych struktur danych. Z tego powodu wygenerowano ramki danych tj. 'X_train', zawierająca obserwacje oparte na zmiennych z listy

'variables_full_wo_target', a także 'Y_train', gdzie umieszczono wektor zmiennej celu. Analogicznie zrealizowano ramki danych dla zbioru testowego: 'X_test' oraz 'Y_test'. Finalne struktury otrzymano jako wynik funkcji 'DMatrix()' z biblioteki 'xgboost', co zrealizowano kodem z rysunku 36.

```
1 xdm_train = xgb.DMatrix(X_train, Y_train, missing = True)
2 xdm_test  = xgb.DMatrix(X_test, Y_test, missing = True)
```

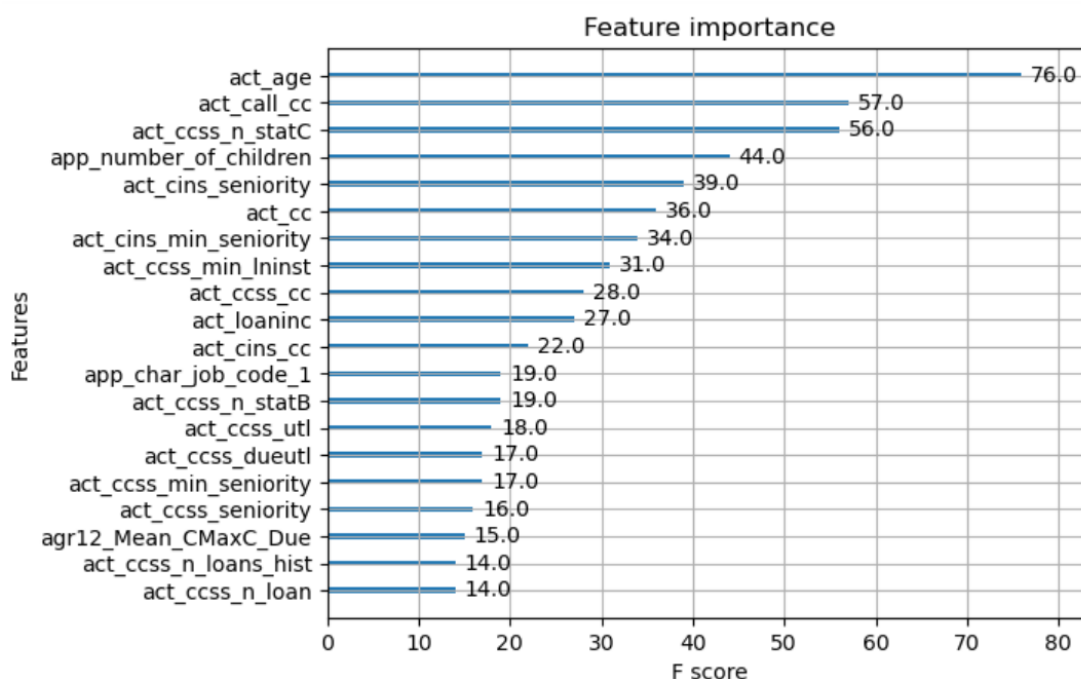
Rysunek 36: Fragment kodu realizujący przygotowanie struktur danych dla modelu XGBoost (źródło opracowanie własne)

Strategia modelowa obejmowała wytrenowanie modelu na zbiorze treningowym, a następnie jego walidacja na zbiorze testowym. Przeprowadzono szereg treningów i walidacji celem znalezienia najlepszego modelu, w tym celu manipulując parametrami uczenia. W następnym kroku użyto nauczonego modelu do utworzenia nowych zmiennych, w których przechowywano klasyfikację wykonaną przez XGBoost. Finalnie podsumowano obliczeniem mocy predykcyjnej modelu dla zbioru testowego i treningowego wyliczając współczynnik GINI przy pomocy funkcji 'roc_auc_score' z biblioteki 'sklearn'.

Pierwszy zbudowany model, stanowił element poglądowy do dalszej analizy. Jego głównym zadaniem, było zbudowanie prostego modelu XGBoost na przygotowanych danych, celem zgrubnego oszacowania istotności zmiennych przy pomocy statystyki F-score, co jest najważniejszym wnioskiem z budowy modelu wstępnego reprezentowanym przez wizualizację na rysunku 37.

Iteracyjne testy jakości zróżnicowanych modeli XGBoost, celem wyboru najlepszego z nich, rozpoczęto od zdefiniowania benchmarków, które muszą być przestrzegane przez wybrany algorytm. Zdefiniowano je w dwóch parametrach:

- 'gini_benchmark' = 0.8 - wartość oznaczająca minimalną wartość współczynnika Gini, która musi zostać spełniona zarówno przez model opracowany na danych treningowych, jak i na danych testowych;
- 'gini_diff_benchmark' = 0.04 - ze względu na złożoność i specyfikę rynku kredytowego, modele predykcyjne Credit Scoring powinny skupiać się nie tylko na odpowiednim rozróżnianiu dobrych i złych klientów, ale również muszą być stabilne w zależności od danych. Stąd potrzeba zdefiniowania odpowiednio niskiej wartości



Rysunek 37: Analiza statystyki Feature Importance dla modelu wstępnego (źródło opracowanie własne)

benchmarku, wychwytyującego jedynie modele o niewielkiej różnicy w jakości predykcji pomiędzy zbiorem treningowym, a testowym.

Do wytrenowania modelu zastosowano funkcję `'train()'` z biblioteki `'xgboost'`. Oferuje ona możliwość manipulowania wieloma wewnętrznymi ustawieniami modelu. W ramach budowy zdecydowano się na manipulację trzema z nich:

- `'max_depth'` - określa maksymalną głębokość pojedynczego drzewa w modelu, przez co kontroluje on złożoność modelu i potencjalne przetrenowanie. Wyższa wartość parametru pozwala drzewu na zastosowanie bardziej skomplikowanych reguł, co może zwiększyć dopasowanie do danych treningowych. Dla niższej wartości, model będzie bardziej regularny, co może pomóc w ogólnej generalizacji i stabilności względem nowych danych. Defaultowa głębokość drzewa to 6, ale dozwolone są wartości z zakresu liczb nieujemnych;
- `'learning_rate'` - kontroluje krok, o jaki model się dostosowuje podczas każdej iteracji. Niska wartość zmniejsza wpływ pojedynczych drzew na model, zapobiegając przetrenowaniu, natomiast wysoka wartość przyspiesza uczenie, lecz może prowadzić do przeuczenia. Optymalna wartość zależy od danych i zadania. Ważne jest strojenie tego parametru wraz z innymi, takimi jak liczba drzew i głębokość, by

uzyskać najlepszą jakość predykcji. Z defaultu przyjmuje się 'learning_rate' równe 0.3, a zakres dozwolonych wartości to [0,1];

- 'min_split_loss' - wartość minimalnej utraty dzielenia węzła podczas budowy drzewa. Jeśli wzrost w funkcji zysku nie przekracza tej wartości, węzeł nie będzie dalej podzielony. Pomaga to kontrolować strukturę drzewa, zapobiegając nadmiernemu dopasowaniu. Wyższa wartość parametru prowadzi do bardziej konserwatywnego modelu, ograniczając ryzyko przeuczenia. Defaultowo założone jest 'min_split_loss' równe 0, jednakże może przyjmować wartości z zakresu liczb nieujemnych.

Poza parametrami, które należało dostroić w ramach testów iteracyjnych, zdefiniowano również inne wymagane ustawienia funkcji 'train', które pozostawały niezmienione w całej fazie badań:

- 'objective' - określa rodzaj problemu, który model ma rozwiązywać, np. czy to jest problem klasyfikacji, regresji, czy inny. Dostępne opcje to m.in. 'reg:squarederror' dla regresji, 'binary:logistic' lub 'binary:logitraw' dla klasyfikacji binarnej, czy też 'multi:softmax' dla wieloklasowej klasyfikacji;
- 'eval_metric' - definiuje miarę, na podstawie której algorytm ocenia efektywność modelu na zbiorze walidacyjnym. Przykładowe metryki to 'rmse' (średni błąd kwadratowy), 'mae' (średni błąd bezwzględny), 'logloss' (logarytmiczna funkcja straty), czy 'auc' (obszar pod krzywą ROC);
- 'num_boost_round' - określa liczbę iteracji w procesie treningu, gdzie każda runda dodaje nowy drzewowy model do kompozycji, poprawiając predykcje. Ważne jest, aby wybrać optymalną wartość, unikając niedouczenia lub przeuczenia. Zbyt mała wartość może prowadzić do niedokładnych predykcji, podczas gdy zbyt duża może spowodować nadmierne dopasowanie.

Wybrane parametry lub testowane ich zakresy zostały przedstawione w tabeli 9.

| Parametr | Testowane wartości |
|-----------------|--------------------|
| max_depth | 3 - 5 |
| seed | 1998 |
| objective | binary:logitraw |
| learning_rate | 0.5 - 0.9 |
| min_split_loss | 3 - 7 |
| eval_metric | auc |
| num_boost_round | 1000 |

Tabela 9: Parametry funkcji 'train' z biblioteki 'xgboost' zastosowane do wyboru odpowiedniego modelu (źródło opracowanie własne)

Na podstawie wybranych parametrów zbudowano kod, przedstawiony na rysunku 38, którego wynikiem była ramka danych opisująca statystyki współczynników Giniego.

```

1 df = pd.DataFrame(
2     {
3         'GINI_train': [],
4         'GINI_test': [],
5         'max_depth': [],
6         'learning_rate': [],
7         'min_split_loss': [],
8         'GINI_diff': []
9     }
10 )
11
12 for i in range(3):
13     for j in range(5):
14         for k in range(5):
15             M = xgb.train(
16                 {
17                     'max_depth': i+3,
18                     'seed': 1998,
19                     'objective': 'binary:logitraw',
20                     'learning_rate': 0.1 * (j + 5),
21                     'min_split_loss': k + 3,
22                     'eval_metric': 'auc'
23                 },
24                 xdm_train,
25                 num_boost_round = 1000,
26                 evals = [
27                     (xdm_test, 'eval'),
28                     (xdm_train, 'train')
29                 ],
30                 verbose_eval = 0
31             )
32
33             Y_M_train = M.predict(xdm_train)
34             Y_M_test = M.predict(xdm_test)
35
36             GINI_train = 2 * roc_auc_score(Y_train, Y_M_train) - 1
37             GINI_test = 2 * roc_auc_score(Y_test, Y_M_test) - 1
38
39             df.loc[len(df)] = {
40                 'GINI_train': GINI_train,
41                 'GINI_test': GINI_test,
42                 'max_depth': i + 3,
43                 'learning_rate': 0.1 * (j + 5),
44                 'min_split_loss': k + 3,
45                 'GINI_diff': abs(GINI_train - GINI_test)
46             }
47 df

```

Rysunek 38: Kod realizujący iteracyjne trenowanie kolejnych modeli celem wyboru najodpowiedniejszego (źródło opracowanie własne)

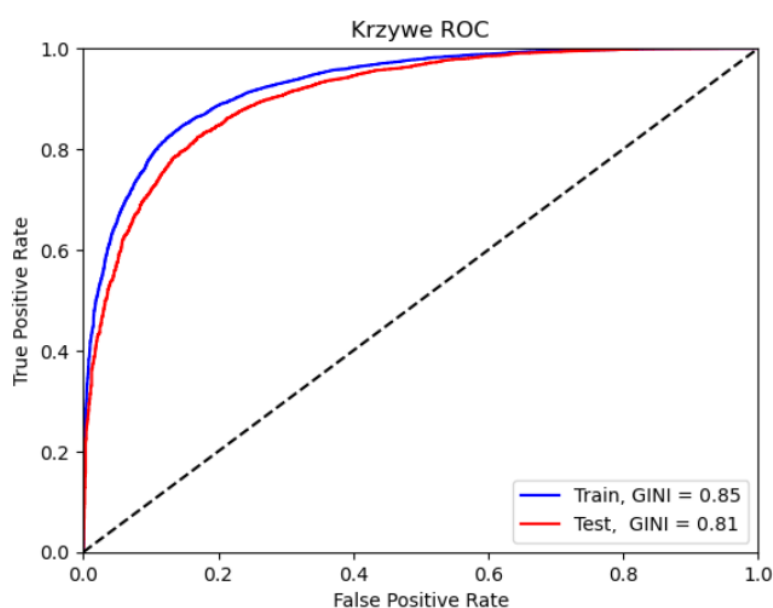
Wykorzystując wcześniej zdefiniowane benchmarki wyfiltrowano modele spełniające założone kryteria wysokiej skuteczności predykcji oraz stabilności. Następnie posortowano je rosnąco względem różnicy między współczynnikami Giniego na zbiorze treningowym, a zbiorze testowym. Najlepszy model zapisano jako 'M_chosen_model' celem późniejszego

| Parametr | Wartości |
|----------------|----------|
| max_depth | 3 |
| learning_rate | 0.5 |
| min_split_loss | 7 |
| GINI_train | 85.28% |
| GINI_test | 81.29% |
| GINI_diff | 4.00% |

Tabela 10: Parametry finalnego modelu XGBoost (źródło opracowanie własne)

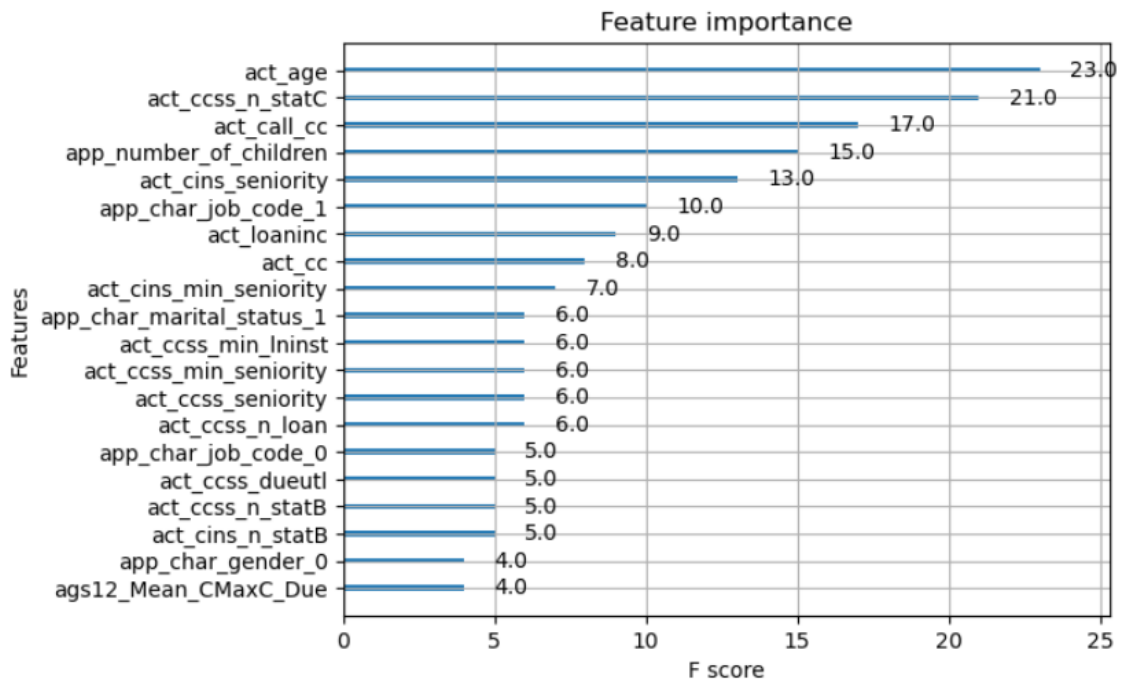
wykorzystania, a jego parametry przedstawiono w tabeli 10.

Podczas testów iteracyjnych zauważono, że model XGBoost jest w stanie generować bardzo wysoką skuteczność predykcji dla zbioru treningowego, osiągając wartości przekraczające 90%, jednakże wraz ze wzrostem współczynnika Giniego rosła także niestabilność modelu. Chcąc uzyskać znacznie bardziej satysfakcjonującą różnicę w jakości klasyfikacji pomiędzy zbiorami, następowały istotne spadki Giniego, co zabierało istotny argument za wyborem algorytmów Uczenia Maszynowego zamiast standardowych kart scoringowych. W związku z powyższym zaakceptowano 4% różnicy, zachowując wysoką skuteczność predykcji przekraczającą 80%. Zauważono również, że zwiększanie liczby iteracji treningowych modelu powyżej użytej wartości 1000 nie przynosi korzyści, jednakże odnotowano istotną różnicę jakości predykcji w zależności od parametru 'num_boost_round'. Jakość klasyfikacji wybranego modelu zwizualizowano na rysunku 39 za pomocą krzywych ROC.



Rysunek 39: Wizualizacja wybranego modelu krzywymi ROC (źródło opracowanie własne)

Analizując wykresy ROC dla zbiorów treningowego oraz testowego odnotowano podobny, stabilny przebieg oraz kształt krzywych, co pozwoliło upewnić się o poprawnie zrealizowanej budowie. Jako ostatni element tego etapu pracy, zbadano ważność zmiennych modelu finalnego, celem porównania jego wskazań z analizą wstępną, co zrealizowano na rysunku 40.



Rysunek 40: Analiza Feature Importance dla finalnego modelu (źródło opracowanie własne)

Mimo zmian wartości statystyki i globalnego zmniejszenia się F-score, pierwsze pięć najbardziej istotnych zmiennych, według modelu finalnego, nie różni się od wskazań wstępnych. Co więcej, z dwudziestu najważniejszych zmiennych, piętnaście powtarza się w obydwu analizach, choć w zmienionej kolejności. Zauważono, iż model finalny widzi większy potencjał niż model wstępny w zmiennych zbinowanych, wskazując cztery z nich wśród zwizualizowanych dwudziestu, zamiast wstępnej sugestii o jednej takiej zmiennej.

Opracowany model klasyfikacji binarnej XGBoost do celów predykcji wystąpienia zdarzenia 'default' w ciągu pierwszych 12 miesięcy od zaciągnięcia zobowiązania kredytowego przez klienta, zostanie przetestowany pod kątem jego odporności na ingerencje z dziedziny Adversarial Machine Learning.

4 Atak na opracowany model

4.1 Opis hipotez badawczych oraz strategii ataku

Analiza zrealizowana na łamach rozdziału drugiego umożliwiła zapoznanie się z aktualnym stanem wiedzy i badań dotyczących Kontradyktoryjnego Uczenia Maszynowego, uwzględniając szerszy opis typów ataków, jakie można stosować w próbach zaburzenia działania modeli ML. W rozdziale trzecim opisano proces budowy i walidacji algorytmu XGBoost, wykorzystanego w klasyfikacji wniosków kredytowych względem wystąpienia zdarzenia default w ciągu pierwszych dwunastu miesięcy od zaciągnięcia zobowiązania. Natomiast rozdział czwarty stanowi opis weryfikacji hipotez badawczych, które dotyczą zachowania modelu klasyfikacji binarnej pod wpływem zastosowania czynności z dziedziny Adversarial Machine Learning.

W celu rzetelnego zbadania wrażliwości przygotowanego w rozdziale trzecim modelu XGBoost dla Credit Scoring, postawiono cztery hipotezy badawcze:

- Hipoteza 1 - odwrócenie wartości binarnej zmiennej celu w 1 % danych zbioru treningowym uniemożliwia dalsze wykorzystanie modelu XGBoost w bankowym środowisku produkcyjnym;
- Hipoteza 2 - zdublowanie z odwróconymi wartościami binarnej zmiennej celu 1 % danych w zbiorze treningowym uniemożliwia dalsze wykorzystanie modelu XGBoost w bankowym środowisku produkcyjnym;
- Hipoteza 3 - zmanipulowanie jednocześnie trzema z pięciu najważniejszych zmiennych modelowych w zakresie $\pm 1-5$ % wartości tych zmiennej pozwala na odwrócenie klasyfikacji binarnej otrzymanej jako odpowiedź z modelu XGBoost w ponad 10 % przypadków;
- Hipoteza 4 - zmanipulowanie jedną z pięciu najważniejszych zmiennych modelowych w zakresie $\pm 1-10$ % wartości tej zmiennej pozwala na odwrócenie klasyfikacji binarnej otrzymanej jako odpowiedź z modelu XGBoost w ponad 10 % przypadków;

Hipoteza 1 została zweryfikowana przy pomocy zbioru danych zastosowanego do budowy modelu XGBoost w rozdziale trzecim. Podział na podzbiory treningowy i testowy zrealizowano również identycznie, aby mieć pewność, iż zarówno dane użyte do modelu

finalnego, jak również ataku białoskrzynkowego były ze sobą tożsame, dzięki czemu badanie było rzetelne. W kolejnym kroku dokonano szeregu testów polegających na losowym wyborze procentowo zdefiniowanej części zbioru testowego, odwróceniu wartości binarnej zmiennej celu i wytrenowaniu nowego modelu, zachowując parametry modelu finalnego. Wyniki przedstawiono za pomocą tabeli wartości opartych na współczynniku Giniego w zależności od procentowo zdefiniowanej wielkości atakowanej części zbioru treningowego. Dodatkowo rezultaty zwizualizowano za pomocą zbiorowego wykresu liniowego dla współczynnika Giniego dla zbioru testowego oraz treningowego w zależności od procenta atakowanych danych.

Hipoteza 2 również została sprawdzona przy użyciu zbioru danych z rozdziału trzeciego. Ponownie podzielono podzbiory w ten sam sposób. Podział na podzbiory treningowy i testowy zrealizowano również identycznie, aby mieć pewność, iż zarówno dane użyte do modelu finalnego, jednakże etap testowy nieco się różni. Wylosowana próba została skopiowana na zewnątrz zbioru modelowego, a podmiana wartości binarnej zmiennej celu nastąpiła bez modyfikacji zbioru treningowego. Następnie spreparowany podzbiór skonkatenowano z treningowym zestawem modelowym. Kolejne kroki zrealizowano analogicznie jak dla hipotezy 1.

Do weryfikacji hipotezy 3 wciąż korzystano z tych samych podzbiorów jak przy badaniu innych hipotez. W zakresie części praktycznej wybrano losowo pewną liczbę obserwacji oznaczonych jako wnioski odrzucone w danych wejściowych, jednocześnie będące zakwalifikowane jako 'defaulty' przez model finalny. Następnie dla każdej z tych obserwacji podjęto próbę wygenerowania Wrogich Próbek, poprzez odchylenia wartości trzech z pięciu najistotniejszych zmiennych według statystyki Feature Importance modelu XGBoost wytrenowanego w rozdziale drugim. Wartości wybranych zmiennych modyfikowano poprzez dodanie lub odjęcie pewnych procentowych wartości tych zmiennych, gdzie najmniejsza manipulacja wynosiła 1 % wartości, a największa 50 %. Tę część sfinalizowano poprzez zweryfikowanie odpowiedzi modelu na wprowadzone, zmanipulowane próbki oraz sprawdzenie skuteczności ataku porównując liczbę kwalifikacji zapytań do modelu jako wniosek zaakceptowano w stosunku do liczby wszystkich zapytań/próbek.

Badania nad hipotezą 4 przebiegały analogicznie jak dla hipotezy 3. Różnica polegała na generacji Wrogich Próbek, w których tym razem manipulacji poddano nie wszystkie

trzy zmienne jednocześnie, a każdą z osobna. Zakres odchyień wartości ponownie wyniósł od 1 % do 50 %, a porównanie odpowiedzi modelu finalnego zrealizowano dla poszczególnych zmiennych.

W części programistycznej weryfikacji postawionych hipotez skorzystano z kilku narzędzi nie stosowanych w poprzednich etapach pracy. W ramach kodu kilkakrotnie wykorzystano funkcję `reset_index()`, której zadaniem było ponowne nadanie numeracji wierszy w zbiorze wylosowanych obserwacji, co ułatwiło iterowanie po zestawie danych oraz realizowanie zarówno modyfikacji, jak i kopiowanie. Skorzystano również z funkcji `query()`, która umożliwia prostą filtrację zbioru po konkretnych wartościach zmiennej. Funkcja `insert()`, dała możliwość wprowadzenia nowej zmiennej z wartościami na określonej przez użytkownika pozycji w docelowej ramce danych. Skorzystano także z funkcjonalności `drop()`, która usuwa zawartość zbioru, jednocześnie pozostawiając jego pustą strukturę. Z kolei stosując funkcję `accuracy_score` z biblioteki `'sklearn'`, oceniono skuteczność Wrogich Próbek, porównując klasyfikację nadaną im przez model z rzeczywistymi oznaczeniami ze zbioru treningowego. Używano także funkcji `concat()` z biblioteki `'pandas'`, która służyła do dopisywania kolejnych wierszy do ramki danych. Kluczowym aspektem w weryfikacji hipotez badawczych było wykorzystanie funkcji `sample()`, pozwalającej na losowy wybór zdefiniowanej liczby wierszy ze wskazanego zbioru danych. Mając dostęp do tych narzędzi, przeprowadzono szereg badań, celem przetestowania rozpatrywanych scenariuszy ataku na model XGBoost, które opisano w kolejnych podrozdziałach.

4.2 Hipoteza 1

Hipoteza pierwsza stanowiła, iż odwrócenie wartości binarnej zmiennej celu w 1 % danych zbioru treningowego uniemożliwia dalsze wykorzystanie modelu XGBoost w bankowym środowisku produkcyjnym. Oznacza to, że dostęp do niewielkiej części danych, na których następuje uczenie, jest wystarczający do skutecznej degradacji wiarygodności zawartych w nim informacji i zmusza analityków odpowiedzialnych za skuteczność modelu do przededefiniowania aktualnie stosowanych parametrów. Hipoteza zakłada, iż atak na dane znacząco obniży współczynnik Giniego zarówno dla weryfikacji na próbie treningowej, jak i testowej, jednocześnie doprowadzając do nieakceptowalnego powiększenia się różnicy w Ginim pomiędzy podzbiorami. Założenie, że dostęp do zaledwie 1 % zbioru jest dostateczny zostało zasugerowane w cytowanych w rozdziale drugim badaniach nad atakami na systemy filtracji antyspamowej w wiadomościach mailowych (Kuchipudi et al., 2020), co postanowiono sprawdzić w przypadku modelu XGBoost. Dla bardziej rzetelnej oceny, zrealizowano szereg testów, zakładając dostęp różnych zakresów danych, sięgając do 15 % zawartości.

W ramach weryfikacji hipotezy 1 wybrano w zbiorze treningowym pewną część znajdujących się w nim obserwacji i zmieniono wartość zmiennej celu 'default12' na odwrotną. Rozpoczęto od utworzenia ramek danych właściwych dla tego badania co zrealizowano kodem z rysunku 41:

```
1 train_AML1, test_AML1 = train_test_split(data_full[variables_full], random_state = 2001, test_size = 0.3)
2 print("Rozmiar zbioru treningowego:", train_AML1.shape)
3 print("Rozmiar zbioru testowego:", test_AML1.shape)
```

Rozmiar zbioru treningowego: (20843, 213)
Rozmiar zbioru testowego: (8934, 213)

Rysunek 41: Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 1 (źródło opracowanie własne)

Wybrano zakres wartości procentowej modyfikowanych danych jako parametr 'attacked_proc1', co dla "attacked_proc1" = 15" oznacza, że wykonano 15 testów, gdzie kolejne wartości procentowe zmienionych danych znajdowały się w zakresie liczb całkowitych od 1 do 15. Zdefiniowano ramkę danych 'df_AML1', w której były umieszczane statystyki dla kolejnych testów. Następnie w pętli 'for' dokonywano losowania liczby manipulowanych obserwacji, obliczając ją na podstawie zdefiniowanej wcześniej wartości 'attacked_proc1'. Dla wylosowanej próby dokonywano podmianę wartości zmiennej 'default12' bezpośrednio

w zbiorze danych. W kolejnym kroku dokonano standardowych operacji przygotowania zmanipulowanego zbioru do wykonania na nim procesu przetrenowania modelu, stosując parametry identyczne jak w modelu finalnym. Kod realizujący powyżej opisane operacje przedstawiono na rysunku 42:

```
for x in range(attacked_proc1 + 1):
    nofRows = math.ceil(x * len(train_AML1.index) / 100)
    train_AML1_attacked = train_AML1.copy()
    attacked_sample1 = train_AML1.sample(n = nofRows,
                                         random_state = 1234,
                                         replace = False)

    for j in attacked_sample1.index:
        if train_AML1.loc[j, 'default12'] == 0:
            train_AML1_attacked.loc[j, 'default12'] = 1
        else: train_AML1_attacked.loc[j, 'default12'] = 0

    X_train_AML1 = train_AML1_attacked[variables_full_wo_target]
    Y_train_AML1 = train_AML1_attacked[target_variable]
    X_test_AML1 = test_AML1[variables_full_wo_target]
    Y_test_AML1 = test_AML1[target_variable]

    xdm_train_AML1 = xgb.DMatrix(X_train_AML1, Y_train_AML1, missing = True)
    xdm_test_AML1 = xgb.DMatrix(X_test_AML1, Y_test_AML1, missing = True)

    M_chosen_model_AML1 = xgb.train(
        {
            'max_depth' : 3,
            'seed' : 1998,
            'objective' : 'binary:logitraw',
            'learning_rate' : 0.5,
            'min_split_loss' : 7,
            'eval_metric' : 'auc'
        },
        xdm_train_AML1,
        num_boost_round = 1000,
        evals = [
            (xdm_test_AML1, 'eval'),
            (xdm_train_AML1, 'train')
        ],
        verbose_eval = 0
    )

    Y_M_train_chosen_model_AML1 = M_chosen_model_AML1.predict(xdm_train_AML1)
    Y_M_test_chosen_model_AML1 = M_chosen_model_AML1.predict(xdm_test_AML1)

    GINI_train_chosen_model_AML1 = 2 * roc_auc_score(Y_train_AML1, Y_M_train_chosen_model_AML1) - 1
    GINI_test_chosen_model_AML1 = 2 * roc_auc_score(Y_test_AML1, Y_M_test_chosen_model_AML1) - 1
```

Rysunek 42: Fragment kodu realizujący procesowanie danych celem weryfikacji hipotezy 1 (źródło opracowanie własne)

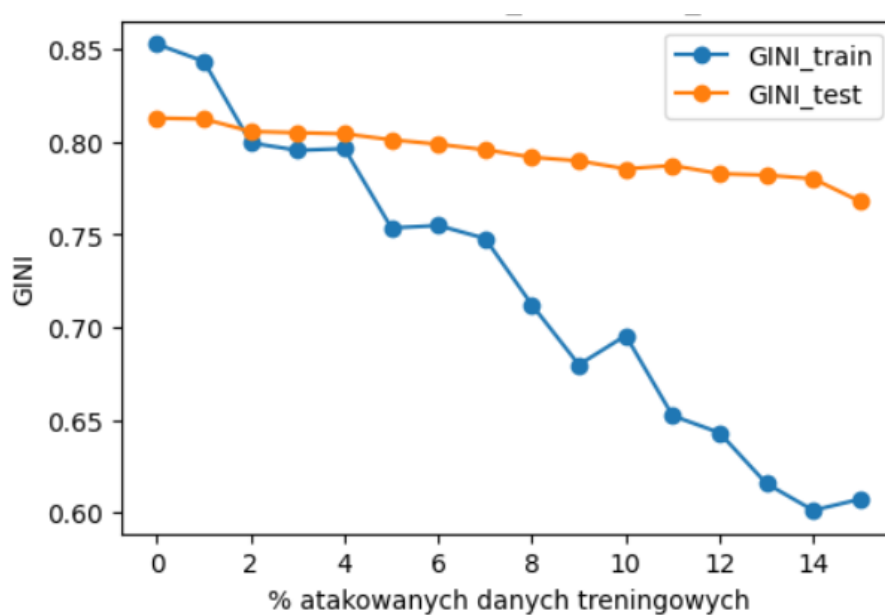
Po zweryfikowaniu przetrenowanych modeli na zbiorze walidacyjnym, statystyki współczynnika Giniego umieszczono w przygotowanej wcześniej ramce danych 'df_AML1' i przedstawiono w tabeli 14. Łatwo zauważyć, że manipulacja 1 % danych zaburza działanie modelu w stopniu niedostatecznym do przyjęcia hipotezy 1. XGBoost wciąż przestrzega założone benchmarki, co więcej, algorytm może wydawać się bardziej stabilny, co widać poprzez niższą wartość 'GINI_diff', przy praktycznie nie zmienionej wartości 'GINI.test'. Dużo większy wpływ widać przy zastosowaniu manipulacji 2 % danych treningowych, gdyż to zaburzenie wybija statystyki modelu poza założone benchmarki. Ogólna analiza wyników jednoznacznie wskazuje istotny wpływ ataku na zbiór uczący na wskaźniki 'GINI_train', przy względnej stabilności Giniego dla zbioru testowego. Współczynnik

'GINI_test', mimo manipulacji w 15 % danych obniżył się tylko o 4.5 %, jednakże należy pamiętać o nieakceptowalnie dużej wartości różnicy Giniego pomiędzy podzbiorami.

| | GINI_train | GINI_test | GINI_diff | % atakowanych danych treningowych |
|----|------------|-----------|-----------|-----------------------------------|
| 0 | 85.28% | 81.29% | 4.00% | 0% |
| 1 | 84.35% | 81.24% | 3.11% | 1% |
| 2 | 79.95% | 80.57% | -0.62% | 2% |
| 3 | 79.55% | 80.50% | -0.96% | 3% |
| 4 | 79.64% | 80.44% | -0.80% | 4% |
| 5 | 75.36% | 80.12% | -4.76% | 5% |
| 6 | 75.49% | 79.88% | -4.39% | 6% |
| 7 | 74.79% | 79.58% | -4.80% | 7% |
| 8 | 71.20% | 79.18% | -7.98% | 8% |
| 9 | 67.97% | 78.98% | -11.00% | 9% |
| 10 | 69.55% | 78.55% | -9.00% | 10% |
| 11 | 65.22% | 78.73% | -13.50% | 11% |
| 12 | 64.30% | 78.29% | -13.99% | 12% |
| 13 | 61.56% | 78.21% | -16.65% | 13% |
| 14 | 60.11% | 78.02% | -17.91% | 14% |
| 15 | 60.71% | 76.79% | -16.08% | 15% |

Tabela 11: Wartość współczynnika Giniego w zależności od procenta atakowanych danych dla hipotezy 1 (źródło opracowanie własne)

Analiza przebiegu wartości współczynnika Giniego dla obydwu podzbiorów w zależności od wielkości zmanipulowanej części podzbioru, zwizualizowana na rysunku 43, jednoznacznie upewnia o konieczności odrzucenia hipotezy 1.



Rysunek 43: Wykres wartości współczynnika Giniego dla zbioru treningowego i testowego dla hipotezy 1 (źródło opracowanie własne)

4.3 Hipoteza 2

Hipoteza druga głosiła, iż skopiowanie 1 % danych zbioru treningowego i odwrócenie wartości binarnej zmiennej celu w kopii, a następnie dołączenie tego zmodyfikowanej części informacji do oryginalnych danych uniemożliwia dalsze wykorzystanie modelu XGBoost w bankowym środowisku produkcyjnym. Ponownie zbadanie tej hipotezy pozwoliło na weryfikację osądu, iż dostęp do niewielkiej części danych uczących, jest wystarczający do zdeprecjonowania przydatności modelu predykcyjnego XGBoost. Hipoteza zakłada, iż atak istotnie wpłynie na wartość współczynnika Giniego tak dla weryfikacji na podzbiorze treningowym, jak i testowym, przy jednoczesnym zwiększeniu się różnicy w Ginim. Badania urzeczono poprzez realizację, szeregu prób testowych, zakładając dostęp do różnych wielkości zestawu danych, maksymalnie osiągając 15 % zbioru uczącego.

Rozpoczęto od ponownego podzielenia zbioru wejściowego na części identyczne jak w przypadku weryfikacji pierwszej z hipotez, a następnie przypisania ich do zmiennych 'train_AML2' oraz 'test_AML2', co zrealizowano kodem z rysunku 44:

```
1 train_AML2, test_AML2 = train_test_split(data_full[variables_full], random_state = 2001, test_size = 0.3)
2 print("Rozmiar zbioru treningowego:", train_AML2.shape)
3 print("Rozmiar zbioru testowego:", test_AML2.shape)
```

Rozmiar zbioru treningowego: (20843, 213)
Rozmiar zbioru testowego: (8934, 213)

Rysunek 44: Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 2 (źródło opracowanie własne)

Wybrano zakres w jakim manipulowano wartością procentową atakowanych danych jako parametr 'attacked_proc2', co dla "attacked_proc2" = 15" oznacza, że wykonano 15 testów, gdzie kolejne wartości procentowe zmienionych danych znajdowały się w zakresie liczb całkowitych od 1 do 15. Zdefiniowano ramkę danych 'df_AML2', w której umieszczano wskazania współczynników dla kolejnych prób. W kolejnym kroku zapętłono losowanie liczby manipulowanych obserwacji, wylosowany podzbiór kopiowano, podmieniano wartości binarnej zmiennej celu 'default12' i finalizowano doklejeniem spreparowanej próby do trenowanego zbioru. Ostatnim etapem było wykonanie czynności przygotowujących utworzony zestaw testowy do zrealizowania na nim procesu treningu modelu, stosując parametry tożsame z algorytmem wybranym w rozdziale trzecim. Kod wykonujący wyżej opisane etapy przedstawiono na rysunku 45.

```

for x in range(attacked_proc2 + 1):
    nofRows = math.ceil(x * len(train_AML2.index) / 100)
    train_AML2_attacked = train_AML2.drop(train_AML2.index)
    attacked_sample2 = train_AML2.sample(n = nofRows, random_state = 5678, replace = False)
    attacked_sample2.reset_index(drop=True, inplace=True)

    for j in range(len(attacked_sample2.index)):
        if attacked_sample2.loc[j, 'default12'] == 0:
            attacked_sample2.loc[j, 'default12'] = 1
        else:
            attacked_sample2.loc[j, 'default12'] = 0

    train_AML2_attacked = pd.concat([train_AML2, attacked_sample2])

    X_train_AML2 = train_AML2_attacked[variables_full_wo_target]
    Y_train_AML2 = train_AML2_attacked[target_variable]
    X_test_AML2 = test_AML2[variables_full_wo_target]
    Y_test_AML2 = test_AML2[target_variable]

    xdm_train_AML2 = xgb.DMatrix(X_train_AML2, Y_train_AML2, missing = True)
    xdm_test_AML2 = xgb.DMatrix(X_test_AML2, Y_test_AML2, missing = True)

    M_chosen_model_AML2 = xgb.train(
        {
            'max_depth' : 3,
            'seed' : 1998,
            'objective' : 'binary:logitraw',
            'learning_rate' : 0.5,
            'min_split_loss' : 7,
            'eval_metric' : 'auc'
        },
        xdm_train_AML2,
        num_boost_round = 1000,
        evals = [
            (xdm_test_AML2, 'eval'),
            (xdm_train_AML2, 'train')
        ],
        verbose_eval = 0
    )

    Y_M_train_chosen_model_AML2 = M_chosen_model_AML2.predict(xdm_train_AML2)
    Y_M_test_chosen_model_AML2 = M_chosen_model_AML2.predict(xdm_test_AML2)

    GINI_train_chosen_model_AML2 = 2 * roc_auc_score(Y_train_AML2, Y_M_train_chosen_model_AML2) - 1
    GINI_test_chosen_model_AML2 = 2 * roc_auc_score(Y_test_AML2, Y_M_test_chosen_model_AML2) - 1

```

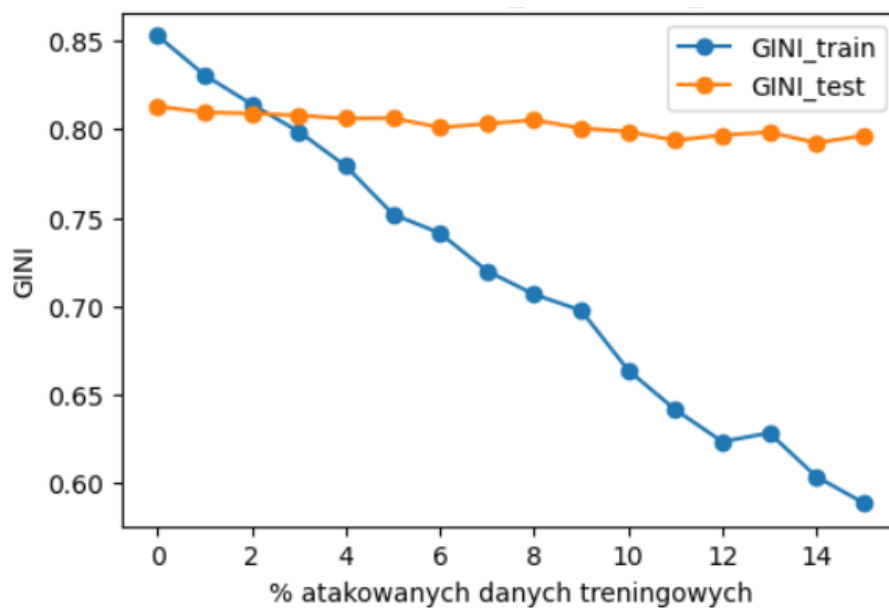
Rysunek 45: Fragment kodu realizujący procesowanie danych celem weryfikacji hipotezy 2 (źródło opracowanie własne)

Po zbadaniu właściwości modeli na zbiorze walidacyjnym, rozkład wartości współczynnika Giniego zapisano w ramce danych 'df_AML2' oraz przedstawiono w tabeli 12. Ponownie manipulacja jedynie 1 % danych treningowych pogarsza wyniki modelu w zbyt małym stopniu, nawiązując do postawionych w rozdziale trzecim benchmarków. W przypadku testu dla hipotezy 2, zauważono obniżenie współczynnika Giniego o wartość nieco większą dla obydwu podzbiorów niż dla hipotezy pierwszej. Ponadto, można wywnioskować, iż atak zakładający dołożenie dodatkowych obserwacji do zbioru treningowego, jest w stanie zmniejszyć wartość 'GINI_train' do poziomu nawet do dwóch procent niższego niż dla podmiany binarnej wartości zmiennej celu bezpośrednio w zbiorze treningowym. Warto również zauważyć, że rozpatrywany rodzaj ataku jest znacznie mniej inwazyjny dla zbioru testowego, ponieważ jesteśmy w stanie obniżyć skuteczność klasyfikacji poniżej 80 % dopiero przy manipulacji dziesięcioma procentami oryginalnych danych, gdzie dla ingerencji bez kopiowania obserwacji na zewnątrz osiągnięto to zjawisko przy dostępie do 5 % podzbioru, a dla 15 % stopnia manipulacji 'GINI_test' jest niższy jedynie o 1.65 %.

| | GINI_train | GINI_test | GINI_diff | % atakowanych danych treningowych |
|----|------------|-----------|-----------|-----------------------------------|
| 0 | 85.28% | 81.29% | 4.00% | 0% |
| 1 | 83.05% | 80.96% | 2.09% | 1% |
| 2 | 81.41% | 80.89% | 0.52% | 2% |
| 3 | 79.86% | 80.78% | -0.92% | 3% |
| 4 | 77.91% | 80.61% | -2.70% | 4% |
| 5 | 75.19% | 80.63% | -5.44% | 5% |
| 6 | 74.12% | 80.10% | -5.98% | 6% |
| 7 | 71.98% | 80.31% | -8.32% | 7% |
| 8 | 70.66% | 80.53% | -9.87% | 8% |
| 9 | 69.75% | 80.05% | -10.30% | 9% |
| 10 | 66.37% | 79.87% | -13.50% | 10% |
| 11 | 64.14% | 79.37% | -15.23% | 11% |
| 12 | 62.33% | 79.67% | -17.34% | 12% |
| 13 | 62.82% | 79.83% | -17.01% | 13% |
| 14 | 60.35% | 79.23% | -18.88% | 14% |
| 15 | 58.87% | 79.64% | -20.77% | 15% |

Tabela 12: Wartość współczynnika Giniego w zależności od procenta atakowanych danych dla hipotezy 2 (źródło opracowanie własne)

Analiza przebiegu wartości współczynnika Giniego dla obydwu podzbiorów w zależności od wielkości zmanipulowanej części podzbioru, zwizualizowana na rysunku 46, jednoznacznie upewnia o konieczności odrzucenia hipotezy 2. Co więcej, zauważalna jest różnica w przebiegu 'GINI_train', gdzie obniżanie się tej wartości ma charakter niemal liniowy, w odróżnieniu od sytuacji dla hipotezy 1, gdzie obserwowano trudne do przewidzenia skoki.



Rysunek 46: Wykres wartości współczynnika Giniego dla zbioru treningowego i testowego dla hipotezy 2 (źródło opracowanie własne)

4.4 Hipoteza 3

Trzecia hipoteza dotyczyła manipulacji wartościami zmiennych objaśniających celem oszukania XGBoosta i zmuszenia go do zmiany klasyfikacji. W tym przypadku mowa o odchyleniu jednocześnie trzech z pięciu najważniejszych zmiennych modelowych w zakresie $\pm 1-5\%$ ich wartości w ponad 10% przypadków. Tym razem zbadanie hipotezy zakładało atak z typu czarnoskrzynkowych, gdzie nie adversarz nie ma dostępu do danych modelowych i nie może ich w żaden sposób modyfikować. Poprzez wielokrotne odpytywanie modelu danymi jedynie minimalnie różniącymi się od siebie można ocenić, czy oszukiwanie na polu którejkolwiek z nich ma sens, a jeśli tak, to w którym momencie następuje zmiana klasyfikacji.

Rozpoczęto od wyboru atakowanych zmiennych. Na podstawie statystyki Feature Importance, wyliczanej dla modelu finalnego i zwizualizowanej na rysunku 40 w rozdziale trzecim, zdefiniowano pięć najistotniejszych zmiennych jako:

- 'act_age';
- 'act_ccss_n_statC';
- 'act_call_cc';
- 'app_number_of_children';
- 'act_cins_seniority'.

Spośród wyżej wymienionych zmiennych, wybrano trzy, a dokonano tego kierując się nie tylko ich ważnością, lecz także typem i rozkładem, co miało znaczenie dla kolejnych kroków. W kolejnym etapie, założono wybranie losowych obserwacji z atakowanego zbioru, co ułatwiło wybór wartości wszystkich opisujących go zmiennych i zapobiegło konieczności tworzenia sztucznych wniosków przez adversarza nie znającego zakresu możliwych danych. Spowodowało to, iż konieczne było odrzucenie zmiennej 'act_ccss_n_statC', która mimo iż jest ważna, to posiada znaczne braki danych, co skomplikowałoby proces generowania Wrogich Próbek. Jako drugą odrzucono zmienną 'app_number_of_children', ze względu na fakt, iż może ona przyjmować jedynie liczby całkowite oraz uniemożliwione jest wprowadzenie szerokiego zakresu wartości.

Do dalszych badań wybrano zatem zmienne:

- 'act_age';
- 'act_call_cc';
- 'act_cins_seniority'.

Następny krok procesu, wyglądał analogicznie jak dla poprzednich hipotez. Rozpoczęto od utworzenia ramek danych właściwych dla danego testu, co zrealizowano kodem z rysunku 47.

```
train_AML3, test_AML3 = train_test_split(data_full[variables_full], random_state = 2001, test_size = 0.3)
print("Rozmiar zbioru treningowego:", train_AML3.shape)
print("Rozmiar zbioru testowego:", test_AML3.shape)

Rozmiar zbioru treningowego: (20843, 213)
Rozmiar zbioru testowego: (8934, 213)
```

Rysunek 47: Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 3 (źródło opracowanie własne)

Następnie wylosowano ze zbioru treningowego obserwacje, które będą poddane manipulacji celem zmiany ich klasyfikacji. Rozpatrywano jedynie wnioski odrzucone ("default12 == 1"), ponieważ celem ataku jest zakwalifikowanie klienta potencjalnie niewiarygodnego, jako wiarygodnego. Liczbę losowanych próbek dobrano w sposób taki, aby po odrzuceniu wylosowanych obserwacji, dla których wartość rzeczywista zmiennej 'default12' jest różna od klasyfikacji wynikającej z modelu finalnego, liczba modyfikowanych dalej obserwacji była równa 100. W tym przypadku należało wybrać 115 obserwacji, dla parametru 'random_state' równego 3456. Na końcu zresetowano numery wierszy w ramce danych, aby ułatwić iterowanie po niej. Wyżej opisane operacje, zrealizowano kodem z rysunku 48.

W ramach badania niezbędne było utworzenie dużego zbioru Wrogich Próbek, korzystając z wylosowanych wniosków odrzuconych. Przeprowadzono to poprzez dodanie lub odjęcie od wartości każdej ze zmiennych pewnej niewielkiej części jego wartości. W przypadku zmiennych ciągłych, do których należą 'act_call_cc' oraz 'act_cins_seniority', obliczano 1 % wartości dla każdej obserwacji, a następnie poprzez dodanie bądź odjęcie tej wartości, generowano nowy wiersz danych. Dla każdej ze zmiennych definiowano 'count', którego wartość oznaczała wartość graniczną dla manipulacji, tak że np. dla "count = 10"

```

train_AML3_zeros = train_AML3.query("default12 == 1").sample(n = 115, random_state = 3456, replace = False)
train_AML3_zeros.reset_index(drop=True, inplace=True)

X_train_preAML3 = train_AML3_zeros[variables_full_wo_target]
Y_train_preAML3 = train_AML3_zeros[target_variable]
xdm_train_preAML3 = xgb.DMatrix(X_train_preAML3, Y_train_preAML3, missing = True)

Y_M_train_chosen_model_preAML3 = M_chosen_model.predict(xdm_train_preAML3)

pre_rounded_predictions = [1 if pred >= 0.5 else 0 for pred in Y_M_train_chosen_model_preAML3]

pre_true_labels = train_AML3_zeros['default12']
accuracy = accuracy_score(pre_true_labels, pre_rounded_predictions)
print(f"Dokładność przewidywań: {accuracy:.2f}")

Dokładność przewidywań: 0.87

train_AML3_zeros.insert(0, "preModelResponse", pre_rounded_predictions, True)
train_AML3_zeros = train_AML3_zeros.query("default12 == preModelResponse")
train_AML3_zeros.reset_index(drop=True, inplace=True)
print(f"Liczba obserwacji z klasyfikacjami zgodnymi z rzeczywistością: {train_AML3_zeros.shape[0]}")

Liczba obserwacji z klasyfikacjami zgodnymi z rzeczywistością: 100

```

Rysunek 48: Fragment kodu realizujący losowanie próby 100 dłużników, zakwalifikowanych przez model finalnych jako defaulty, celem weryfikacji hipotezy 3 (źródło opracowanie własne)

generowało się 20 nowych obserwacji, dla każdego ze 100 wylosowanych wniosków, co dawało 2000 wierszy ramki danych, które były zbiorem wejściowym dla wykonania tej samej operacji dla kolejnej zmiennej. W przypadku zdefiniowania wartości "count = 5", dla każdej ze zmiennych, finalnie otrzymano zbiór zawierający aż 100 tysięcy obserwacji. W związku z istotnym wzrostem rozmiarów ramki danych zawierających Wrogie Próbkę wraz ze inkrementacją wartości 'count', a co za tym idzie wydłużeniem się czasu obliczeń, wybrano 5 jako wielkość graniczną i analizowano jedynie pięć testów. Warto również zauważyć, że dla zmiennej 'act_age', nie było możliwe modyfikowanie jest wielkości o procent jej wartości, ponieważ wiek przez nią opisywany może przyjmować jedynie liczby całkowite, stąd niewielkie różnice w kodzie, którego przykład dla 'act_call_cc' przedstawiono na rysunku 49.

```

train_AML3_attacked = train_AML3_zeros.drop(train_AML3_zeros.index)

for ind in range(len(train_AML3_var1.index)):
    row = train_AML3_var1[train_AML3_var1.index == ind].copy()
    count = 5
    var_name = 'act_call_cc'
    cons1 = row[var_name] / 100

    for i in reversed(range(count)):
        minus = row.copy()
        minus[var_name] = row[var_name] - cons1 * (i + 1)
        train_AML3_attacked = pd.concat([train_AML3_attacked, minus])

    for i in range(count):
        plus = row.copy()
        plus[var_name] = row[var_name] + cons1 * (i + 1)
        train_AML3_attacked = pd.concat([train_AML3_attacked, plus])

train_AML3_var2 = train_AML3_attacked

```

Rysunek 49: Fragment kodu realizujący generację Wrogich Próbek dla zmiennej 'act_call_cc', celem weryfikacji hipotezy 3 (źródło opracowanie własne)

Finalnym krokiem było przebadanie odpowiedzi modelu z rozdziału trzeciego na wprowadzone, zaburzone dane. Wykonano 5 testów, gdzie w ramach pierwszego z nich wychylano wielkości zmiennych jedynie o 1 procent lub wartość 1, a dla testu piątego sprawdzano kombinacje od 1 do 5 procent, jak również od 1 do 5. Wyniki takie jak liczba obserwacji, dla których model zmienił klasyfikację, jak również względna skuteczność XGBoosta zamieszczono w tabeli 14, a kod realizujący ten etap przedstawiono na rysunku 50.

| Procent/Wartość odchylenia | Względna skuteczność modelu | Liczba ataków | Liczba zmienionych klasyfikacji |
|----------------------------|-----------------------------|---------------|---------------------------------|
| 1 | 100 % | 800 | 0 |
| 2 | 99.75 % | 6400 | 16 |
| 3 | 99.50 % | 21600 | 108 |
| 4 | 99.38 % | 51200 | 320 |
| 5 | 99.20 % | 100000 | 800 |

Tabela 13: Wyniki testów weryfikacyjnych hipotezy 3 (źródło opracowanie własne)

```
X_train_AML3 = train_AML3_attacked[variables_full_wo_target]
Y_train_AML3 = train_AML3_attacked[target_variable]
xdm_train_AML3 = xgb.DMatrix(X_train_AML3, Y_train_AML3, missing = True)

Y_M_train_chosen_model_AML3 = M_chosen_model.predict(xdm_train_AML3)

rounded_predictions = [1 if pred >= 0.5 else 0 for pred in Y_M_train_chosen_model_AML3]

true_labels = train_AML3_attacked['default12']
accuracy = accuracy_score(true_labels, rounded_predictions)
print(f"Dokładność przewidywań: {accuracy:.4f}")

Dokładność przewidywań: 0.9920
```

Rysunek 50: Fragment kodu realizujący odpytywanie modelu finalnego wygenerowanymi wrogimi danymi, celem weryfikacji hipotezy 3 (źródło opracowanie własne)

Na podstawie analizy wyników wywnioskowano, iż model XGBoost do celów Credit Scoring wykazuje się wysoką odpornością na względnie nieduże manipulacje wartości kilku kluczowych zmiennych objaśniających i wymusza odrzucenie hipotezy 3. Jednakże warto zauważyć, że zwiększanie stopnia zmiany wartości podnosi liczbę udanych ataków. Przy większych manipulacjach należy wziąć pod uwagę rzetelność zmiennych, gdyż niewielkie oszustwo w trudniej weryfikowanych cechach klienta takich jak zarobki czy wydatki może zostać niezauważone przez oddział analityczny, co jest niełatwe do osiągnięcia np. w kwestii wieku wnioskodawcy.

4.5 Hipoteza 4

Hipoteza czwarta, podobnie jak trzecia, polegała na manipulacji wartościami zmiennych objaśniających celem oszukania modelu XGBoost i zmuszenia go do zmiany klasyfikacji, jednakże tym razem badanie przeprowadzono dla każdej zmiennej z osobna i w znacznie większym zakresie. Wielkości każdej z cech zmieniono o $\pm 1\text{--}50\%$, co wykraczało poza obszar rozważań w ramach hipotezy 4, w której założono modyfikację do 10% , co miało wygenerować odwrócenie się binarnej etykiety na zmiennej celu w co najmniej jednym na dziesięć ataków. Ponownie założono brak możliwości wprowadzania zaburzeń bezpośrednio w danych uczących. Wrogie Próbkę generowano w oparciu o zmienne wybrane dla hipotezy trzeciej:

- 'act_age';
- 'act_call_cc';
- 'act_cins_seniority'.

Budowę zbioru danych atakujących ponownie uzyskano w ten sam sposób jak dla poprzednich hipotez, a przypisanie do zmiennych 'train_AML4' oraz 'test_AML4' zrealizowano kodem z rysunku 51.

```
train_AML4, test_AML4 = train_test_split(data_full[variables_full], random_state = 2001, test_size = 0.3)
print("Rozmiar zbioru treningowego:", train_AML4.shape)
print("Rozmiar zbioru testowego:", test_AML4.shape)

Rozmiar zbioru treningowego: (20843, 213)
Rozmiar zbioru testowego: (8934, 213)
```

Rysunek 51: Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 4 (źródło opracowanie własne)

W drugim kroku wylosowano ze zbioru treningowego 100 obserwacji opisujących dłużników, przypisanych przez model finalny do "default12' == 1", jednakże w odróżnieniu do hipotezy 3, gdzie dokonano jednego losowania, w tym przypadku zdecydowano się przeprowadzić je niezależnie dla każdej ze zmiennych. Na rysunku 52 przedstawiono kod realizujący wyżej opisaną operację dla jednej z wybranych zmiennych, którą była 'act_age':

```

train_AML4_zeros_var1 = train_AML4.query("default12 == 1").sample(n = 112, random_state = 9876, replace = False)
train_AML4_zeros_var1.reset_index(drop=True, inplace=True)

X_train_preAML4_var1 = train_AML4_zeros_var1[variables_full_wo_target]
Y_train_preAML4_var1 = train_AML4_zeros_var1[target_variable]
xdm_train_preAML4_var1 = xgb.DMatrix(X_train_preAML4_var1, Y_train_preAML4_var1, missing = True)

Y_M_train_chosen_model_preAML4_var1 = M_chosen_model.predict(xdm_train_preAML4_var1)

pre_rounded_predictions_var1 = [1 if pred >= 0.5 else 0 for pred in Y_M_train_chosen_model_preAML4_var1]

pre_true_labels_var1 = train_AML4_zeros_var1['default12']
accuracy_var1 = accuracy_score(pre_true_labels_var1, pre_rounded_predictions_var1)
print(f"Dokładność przewidywań: {accuracy_var1:.2f}")

Dokładność przewidywań: 0.89

train_AML4_zeros_var1.insert(0, "preModelResponse", pre_rounded_predictions_var1, True)
train_AML4_zeros_var1 = train_AML4_zeros_var1.query("default12 == preModelResponse")
train_AML4_zeros_var1.reset_index(drop=True, inplace=True)
print(f"Liczba obserwacji z klasyfikacjami zgodnymi z rzeczywistością: {train_AML4_zeros_var1.shape[0]}")

Liczba obserwacji z klasyfikacjami zgodnymi z rzeczywistością: 100

```

Rysunek 52: Fragment kodu realizujący losowanie próby 100 dłużników, zakwalifikowanych przez model finalnych jako defaulty, dla zmiennej 'act_age', celem weryfikacji hipotezy 4 (źródło opracowanie własne)

W celu zbadania hipotezy czwartej, należało utworzyć zbiór Wrogich Próbek dla każdej było utworzenie zbioru Wrogich Próbek dla każdej z analizowanych cech, opracowując go na podstawie wylosowanych wniosków odrzuconych. Przeprowadzono to analogicznie do procedury zastosowanej w przypadku trzeciej z hipotez, jak również ponownie dla każdego z testów zdefiniowano zmienną 'count', jednakże w przypadku manipulacji tylko jedną cechą, wielkość zbioru atakującego nie była problemem. W przypadku zdefiniowania wartości "count = 10", dla każdej ze zmiennych otrzymano zbiory o zawartości dwóch tysięcy obserwacji. Przykład kodu generującego zbiór atakujący dla zmiennej 'act_cins_seniority' przedstawiono na rysunku 53.

```

train_AML4_attacked_var3 = train_AML4_zeros_var3.drop(train_AML4_zeros_var3.index)

for ind in range(len(train_AML4_zeros_var3.index)):
    row = train_AML4_zeros_var3[train_AML4_zeros_var3.index == ind].copy()
    count = 50
    var_name = 'act_cins_seniority'
    cons1 = row[var_name] / 100

    for i in reversed(range(count)):
        minus = row.copy()
        minus[var_name] = row[var_name] - cons1 * (i + 1)
        train_AML4_attacked_var3 = pd.concat([train_AML4_attacked_var3, minus])

    for i in range(count):
        plus = row.copy()
        plus[var_name] = row[var_name] + cons1 * (i + 1)
        train_AML4_attacked_var3 = pd.concat([train_AML4_attacked_var3, plus])

```

Rysunek 53: Fragment kodu realizujący generację Wrogich Próbek dla zmiennej 'act_cins_seniority', celem weryfikacji hipotezy 4 (źródło opracowanie własne)

W ostatnim etapie zbadano odpowiedzi modelu finalnego na zbiór atakujący. Wykonano po 6 testów dla każdej z wybranych zmiennych, gdzie w ramach pierwszego z nich edytowano wielkości cech o 5 procent lub wartość 5, a dla testu szóstego założono 50-stopniową manipulację. Rezultaty badań, reprezentowane przez wskaźniki takie jak liczba obserwacji, dla których XGBoost został oszukany, czy też względna skuteczność modelu, umieszczono w tabeli 12, a kod procesujący tę fazę testów przedstawiono na rysunku 54.

| Zmienna | Procent/ wartość odchylenia | Względna skuteczność modelu | Liczba ataków | Liczba zmienionych klasyfikacji |
|--------------------|-----------------------------------|-----------------------------------|------------------|---------------------------------------|
| act_age | 5 | 100 % | 1000 | 0 |
| | 10 | 99.4 % | 2000 | 12 |
| | 20 | 97.8 % | 4000 | 89 |
| | 30 | 96.7 % | 6000 | 200 |
| | 40 | 95.9 % | 8000 | 331 |
| | 50 | 95.3 % | 10000 | 471 |
| act_call_cc | 5 | 100 % | 1000 | 0 |
| | 10 | 100 % | 2000 | 0 |
| | 20 | 100 % | 4000 | 0 |
| | 30 | 100 % | 6000 | 0 |
| | 40 | 100 % | 8000 | 0 |
| | 50 | 99.9 % | 10000 | 14 |
| act_cins_seniority | 5 | 100 % | 1000 | 0 |
| | 10 | 99.8 % | 2000 | 4 |
| | 20 | 99.7 % | 4000 | 14 |
| | 30 | 99.5 % | 6000 | 30 |
| | 40 | 99.3 % | 8000 | 58 |
| | 50 | 99.1 % | 10000 | 91 |

Tabela 14: Wyniki testów weryfikacyjnych hipotezy 4 (źródło opracowanie własne)

```
X_train_AML4_var3 = train_AML4_attacked_var3[variables_full_wo_target]
Y_train_AML4_var3 = train_AML4_attacked_var3[target_variable]
xtrain_AML4_var3 = xgb.DMatrix(X_train_AML4_var3, Y_train_AML4_var3, missing = True)

Y_M_train_chosen_model_AML4_var3 = M_chosen_model.predict(xtrain_AML4_var3)

rounded_predictions_var3 = [1 if pred >= 0.5 else 0 for pred in Y_M_train_chosen_model_AML4_var3]

true_labels_var3 = train_AML4_attacked_var3['default12']
accuracy_var3 = accuracy_score(true_labels_var3, rounded_predictions_var3)
print(f"Dokładność przewidywań: {accuracy_var3:.3f}")

Dokładność przewidywań: 0.991
```

Rysunek 54: Fragment kodu realizujący odpytywanie modelu finalnego wygenerowanymi wrogimi danymi, dla zmiennej 'act_cins_seniority', celem weryfikacji hipotezy 4 (źródło opracowanie własne)

Analizując skuteczność ataków dla trzech zmiennych z osobna, łatwo zauważyć, iż najwięcej błędnych klasyfikacji wygenerował model atakowany różnymi wartościami wieku wnioskodawcy. Ingerencja w wartości zmiennych ciągłych nie dała satysfakcjonujących rezultatów nawet przy możliwości odchylenia o 50 %. Model finalny wydaje się być najodporniejszy na manipulację cechą 'act_call_cc', gdzie udało się wygenerować zaledwie 14 obserwacji z odwróconą klasyfikacją. W kontekście hipotezy 4, musi zostać ona odrzucona, co wskazuje na wyższy stopień skomplikowania algorytmu XGBoost dla Credit Scoringu, niż dla systemów antyspamowych. Co więcej, opierając się na wynikach tego testu można przyjąć, iż model jest bardzo stabilny i niewrażliwy na odchylenia jednej ze zmiennych objaśniających. Mimo osiągnięcia kilkuprocentowej skuteczności ataku dla odchylenia wieku wnioskodawcy o 50, jednakże w praktyce jest to cecha, która może być nierzeczywista dla tak dużych manipulacji.

4.6 Podsumowanie badań AML

W ramach badań AML zrealizowano zestaw kompleksowych testów, mających na celu negatywne wpłynięcie na osiągi modelu zbudowanego w rozdziale trzecim, weryfikując prawdziwość czterech postawionych na wstępie hipotez badawczych. Realizacja manipulacji modelem XGBoost przeznaczonym do oceny wiarygodności kredytowej została przeprowadzona z zaawansowanym wykorzystaniem procesów przetwarzania danych zaimplementowanych w języku python. Każda hipoteza poruszała tematykę podatności modelu na ataki adversarzy, mając na celu wykrycie potencjalnych punktów wrażliwych i ocenę niezawodności algorytmu.

Hipoteza 1 weryfikowała wpływ odwrócenia wartości binarnej zmiennej 'default12' w niewielkiej części danych treningowych. Celem takiego działania było ustalenie, czy tego typu manipulacja może znacząco wpłynąć na osiągi modelu i spowodować istotny dla przyszłego wykorzystania spadek współczynnika Giniego. Przeprowadzenie licznych testów ujawniło, że przy manipulacji tylko 1% danych treningowych, skuteczność modelu pozostała stosunkowo stabilna, zgodnie z ustalonymi benchmarkami. Analiza współczynników Giniego wskazała, że odporność XGBoost została utrzymana, a hipoteza została ostatecznie odrzucona. Jednakże rewidując osiągi modelu na zbiorze walidacyjnym zauważono, iż realna jest możliwość wywarcia wpływu na mechanizm predykcyjny, choć wymaga ona dużej ingerencji w zestaw danych uczących, bądź bezpośrednio w informacje testowe, zatem prawdziwe jest stwierdzenie, że algorytm XGBoost jest odporny na ataki wewnątrz kilkuprocentowej próbki danych treningowych.

Hipoteza 2 poruszała temat duplikowania i odwracania zmiennej celu w skopiowanym podzbiorze danych treningowych. Celem była ocena, czy ta strategia ataku może prowadzić do obniżenia dokładności modelu. Stwierdzono, iż ten typ ataku jest mniej skuteczny w kontekście wpływu na osiągi predykcji na zbiorze testowym. Znaczące osłabienie się działania modelu na zbiorze treningowym, osiągnięte dla wyższych wartości objętości zainfekowanych danych, ma marginalny wpływ dla skuteczności w danych testowych. Druga hipoteza została definitywnie odrzucona, a dodatkowym wnioskiem jest przewaga ataku bez kopiowania danych, który silniej pogarsza wyniki walidacyjne, a dodatkowo generuje bardzo niestabilny przebieg wartości współczynnika Giniego w zależności od procenta atakowanych danych dla etapu uczenia.

Hipoteza 3 wymagała zagłębienia się w manipulowanie wartościami kluczowych zmiennych objaśniających w celu oszukania klasyfikacji otrzymanej z modelu. Weryfikacja polegała na wielokrotnym odpytywaniu o wynik z nieznacznymi zmienionymi danymi. Analiza wykazała, że model cechuje się wysokim poziomem odporności na tego rodzaju ataki przy modyfikacji kluczowych cech o maksymalnie 5%. Wyniki sugerują, że drobne manipulacje nie mają znaczącego wpływu na wydajność modelu, ale większe zmiany mogą wpłynąć na jego przewidywania. Jednakże w hipotezie trzeciej zauważono największy potencjał, jednakże wymagane są tutaj szersze badania, w ramach których zawierałoby się zaatakowanie większej liczby zmiennych jednocześnie, na wyższych poziomach manipulacji np. 10 %. Przy dokładniejszej analizie wychwycono możliwą tendencję do coraz szybszego wzrostu liczby skutecznie spreparowanych danych, jednakże w tym celu należałoby bliżej przyjrzeć się wydajności kodu generującego Wrogie Próbkę, który jest zbyt wolny by tworzyć większe atakujące zbiory.

Weryfikacja hipotezy 4 zakładała większe manipulacje na pojedynczych zmiennych. Analiza wykazała zdolność modelu do radzenia sobie ze zmianami od -50% do +50% wartości każdej zmiennej. Celem było ustalenie, czy duże zmiany w pojedynczych zmiennych mogą prowadzić do znacznych wypaczeń w przewidywaniach modelu. Wyniki ukazały, że model pozostał stosunkowo stabilny nawet wtedy, gdy zmienne były manipulowane w szerokim zakresie. Nie odnotowano jakichkolwiek sygnałów, aby ta metoda mogła być przydatna w realnych atakach, jako że XGBoost dobrze radzi sobie z tego typu pojedynczymi zaburzeniami, nawet gdy osiągają duże wartości, jednakże by móc to jednoznacznie stwierdzić zaleca się przetestowanie pod tym kątem wszystkich zmiennych modelowych.

Podsumowując, analiza prezentuje dogłębne badanie odporności modelu XGBoost na różne ataki typu AML w kontekście oceny wiarygodności kredytowej. Wyniki ukazują, że model wykazuje wysoką odporność na większość scenariuszy ataków. Choć niektóre manipulacje prowadziły do niewielkich zmian współczynników Giniego lub sugerowanego prawdopodobieństwa zjawiska default w ciągu pierwszych dwunastu miesięcy od zaciągnięcia zobowiązania kredytowego, ogólna wydajność pozostała w akceptowalnych granicach. Niemniej jednak badanie sugeruje, że w niektórych przypadkach większe manipulacje mogą prowadzić do zauważalnych zmian. W związku z tym analiza podkreśla znaczenie ciągłego monitorowania i stałej walidacji skuteczności predykcji, aby zapewnić, że algorytmy oceny kredytowej pozostaną skuteczne i niezawodne w rzeczywistym środowisku bankowym.

Podsumowanie

Praca została rozpoczęta od zrozumienia sposobu, w jaki ocenia się zdolność kredytową. Wyjaśnienie i przedstawienie poszczególnych etapów tej procedury ujawniło jej istotną rolę w umożliwianiu instytucjom finansowym podejmowania trafnych decyzji związanych z udzielaniem kredytów. Następnie omówiono temat tworzenia punktacji kredytowej, ujawniając kryteria wykorzystywane do oceny ryzyka w branży bankowej. Następnie skupiono się na ewolucji wynikającej z coraz większej dostępności ogromnych ilości danych oraz trudnościach w ich analizie w procesie weryfikacji kredytowej. Zrealizowano również wprowadzenie do technik Uczenia Maszynowego stosowanych w dziedzinie Credit Scoring, co ukazało, aktualne osiągnięcia technologiczne w dziedzinie wyjaśnialnych modeli predykcyjnych.

Kolejnym krokiem było skupienie się na teorii dotyczącej zabezpieczania procesu Uczenia Maszynowego przed atakami. Omówiono zagadnienia związane z ryzykiem i bezpieczeństwem algorytmów. Poznanie tych konceptów poszerzyło świadomość o aktualnym stanie wiedzy w tej dziedzinie oraz ukazało, jak duże jest zagrożenie wynikające z działań przeciwników, którzy starają się wykorzystać słabości modeli do manipulacji. Przedstawienie różnych rodzajów ataków na algorytmy Uczenia Maszynowego rzuciło światło na tę kwestię i podkreśliło potrzebę analizy oraz wdrożenia środków obronnych. W części praktycznej opisano zrealizowany proces tworzenia modelu przewidywania ryzyka niewypłacalności w ciągu pierwszych dwunastu miesięcy po udzieleniu kredytu, wykorzystując do tego celu algorytm XGBoost. Wykorzystany zestaw danych zawierał informacje o klientach ubiegających się zarówno o kredyty gotówkowe, jak i ratalne. Przeanalizowano dobór danych, jakość zbiorów oraz użyte narzędzia programistyczne, a skuteczność modelu oceniano przy użyciu wskaźnika Gini. Pracę sfinalizowano poprzez weryfikację postawionych hipotez badawczych, które bezpośrednio nawiązywały do kolejnych scenariuszy ataku.

Literatura

1. Bajek, R. (2011). Wykorzystanie metod eksploracji danych do budowy modeli scoringowych. Politechnika Śląska, Instytut Informatyki.
2. bankier.pl. (2007). Ryzykowny sektor. <https://www.bankier.pl/wiadomosc/Ryzykowny-sektor-1662142.html>, 03.12.2007.
3. bankier.pl. (2012). Tajemnicza liczba, czyli credit scoring. <https://www.bankier.pl/wiadomosc/Tajemnicza-liczba-czyli-credit-scoring-2512458.html>, 02.04.2012.
4. Biggio, B., Nelson, B., & Laskov, P. (2012). Poisoning attacks against support vector machines.
5. bik.pl. (2022). Jak poprawić swoją zdolność kredytową? <https://www.bik.pl/poradnik-bik/jak-poprawic-swoja-zdolnosc-kredytowa>, 5.10.2022.
6. britannica.com. (2019). Dendral. <https://www.britannica.com/technology/DENDRAL>, 19.09.2019.
7. Bujak, L. (2008). Drzewa decyzyjne. <http://www.is.umk.pl/duch/Wyklady/CIS/Prace%20zalicz/08-Bujak.pdf>, 2008.
8. Caire, D., Barton, S., de Zubiria, A., Alexiev, Z., & Dyer, J. (2006). A handbook for developing credit scoring systems in a microfinance context. Washington, Development Alternatives, Inc.
9. calcxml.com. (2023). Financial calculators. <https://www.calcxml.com/do/credit-score-calculator-new?skn=results>, 2023.
10. Castagno, P. (2020). How to identify spam using natural language processing (nlp)? <https://towardsdatascience.com/how-to-identify-spam-using-natural-language-processing-nlp-af91f4170113>, 2020.
11. Cheng, Q., Xu, A., Li, X., & Ding, L. (2022). Adversarial email generation against spam detection models through feature perturbation. Information Security Institute, Johns Hopkins University, Baltimore, MD; Department of Computer Science, American University, Washington, D.C.
12. crif.pl. (2018). Rola „machine learning” w procesach kredytowych. <https://www.crif.pl/wiadomo%C5%9Bci/dla-prasy/2020/sierpie%C5%84/rola-machine-learning-w-procesach->

- kredytowych/, 2018.
13. datascience.eu. (2019). Why the xgboost machine learning algorithm is taking over? <https://datascience.eu/computer-programming/xgboost/>, 2019.
 14. Deryło, L. (2021). Regresja logistyczna - co to jest? <https://www.lukaszderlylo.pl/blog/regresja-logistyczna.html>, 31.01.2021.
 15. direct.money.pl. (2022). Co to jest scoring kredytowy? jak banki ustalają credit scoring i jakich używają systemów? <https://direct.money.pl/artykuly/porady/czym-jest-credit-scoring>, 18.01.2022.
 16. ecomparemo.com. (2020). A brief history of credit scoring in the world. <https://www.ecomparemo.com/info/a-brief-history-of-credit-scoring-in-the-world>, 23.11.2020.
 17. elektronikab2b.pl. (2021). Czym jest uczenie maszynowe i jak można je wykorzystać? <https://elektronikab2b.pl/biznes/53039-czym-jest-uczenie-maszynowe-i-jak-mozna-je-wykorzystac>, 11.12.2020.
 18. experian.com. (2021). What is a good credit score? <https://www.experian.com/blogs/ask-experian/credit-education/score-basics/what-is-a-good-credit-score/>, 11.02.2021.
 19. Fawcett, T. (2005). An introduction to roc analysis. Institute for the Study of Learning and Expertise, 2164 Staunton Court, Palo Alto, CA 94306, USA.
 20. Feng, B., & Xue, W. (2021). Adversarial semi-supervised learning for corporate credit ratings. Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Science, Beijing China, School of Artificial Intelligence, University of Chinese Academic of Science.
 21. fico.com. (2023). Corporate information. <https://fico.gcs-web.com/corporate-information/>, 2023.
 22. fotc.com. (2022). Machine learning — czym jest uczenie maszynowe? <https://fotc.com/pl/blog/machine-learning/>, 16.11.2022.
 23. Gajowniczek, K., Ząbkowski, T., & Szupiluk, R. (2014). Estimating the roc curve and its significance for classification models' assessment. Warszawa, Department of Informatics, Faculty of Applied Informatics and Mathematics, Warsaw

University of Life Sciences - SGGW; Szkoła Główna Handlowa.

24. gov.pl. (2021). Co to jest uczenie maszynowe – inteligentna analiza danych? <https://www.gov.pl/web/popcwsparcie/co-to-jest-uczenie-maszynowe-inteligentna-analiza-danych>, 15.06.2021.
25. Group, T. W. B. (2021). Credit scoring approaches guidelines. Washington, The World Bank Group.
26. habza.com.pl. (2022). Zdolność kredytowa a stopy procentowe. <https://habza.com.pl/zdolnosc-kredytowa-a-stopy-procentowe/>, 15.06.2022.
27. ibm.com. (2021). Sposób działania algorytmu svm. <https://www.ibm.com/docs/pl/spss-modeler/saas?topic=models-how-svm-works>, 17.08.2021.
28. Karolak, Z. (2014). Dynamiczne ujęcie ryzyka kredytowego z ujęciem analizy przeżycia. https://www.sas.com/content/dam/SAS/pl_pl/image/events/mdb/prezentacje/zuzanna-karolak-dynamiczne-ujecie-ryzyka.pdf, 18.11.2014.
29. Kuchipudi, B., Nannapaneni, R. T., & Liao, Q. (2020). Adversarial machine learning for spam filters. Department of Computer Science Central Michigan University Mt. Pleasant, Michigan, USA.
30. Mamczur, M. (2019). Czym jest uczenie maszynowe? i jakie są rodzaje? <https://miroslawmamczur.pl/czym-jest-uczenie-maszynowe-i-jakie-sa-rodzaje/>, 30.11.2019.
31. matplotlib.org. (2022). matplotlib.pyplot documentation. https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html, 2022.
32. Matuszyk, A. (2009). Dotychczasowe oraz nowe trendy w metodzie "credit scoring".
33. mfiles.pl. (2020). Credit scoring. https://mfiles.pl/pl/index.php/Credit_scoring, 19.05.2020.
34. naukowiec.org. (2014). Regresja logistyczna - opis. <https://www.naukowiec.org/wiedza/statystyka/regresja-logistyczna.466.html>, 14.04.2014.
35. newsblog.pl. (2022). Wyjaśnienie regresji a klasyfikacja w uczeniu maszynowym. https://newsblog.pl/wyjasnienie-regresji-a-klasyfikacja-w-uczeniu-maszynowym/Regresja_logistyczna, 1.12.2022.

36. openai.com. (2017). Attacking machine learning with adversarial examples. <https://openai.com/research/attacking-machine-learning-with-adversarial-examples>, 24.02.2017.
37. openai.com. (2023). About openai. <https://openai.com/about>, 2023.
38. Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., & Celik, B. (2017). Practical black-box attacks against machine learning. ACM Asia Conference on Computer and Communications Security, Abu Dhabi, UAE.
39. Pawlicki, M. (2020). Zastosowanie metod uczenia maszynowego do wykrywania ataków sieciowych. Bydgoszcz, Uniwersytet Technologiczno-Przyrodniczy im. Jana i Jędrzeja Śniadeckich.
40. pl.economy pedia.com. (2021). Punktacja kredytowa. <https://pl.economy-pedia.com/11030209-credit-scoring>, 2021.
41. Poon, M. (2007). Scorecards and devices for consumer credits: The case of fair, isaac and company incorporated. The Sociological Review, Issue Supplement S2, 55, s. 284–306.
42. prawnicydotblog.wordpress.com. (2019). Credit scoring. <https://prawnicydotblog.wordpress.com/2019/04/08/credit-scoring/>, 08.04.2019.
43. Przanowski, K. (2014). Credit scoring w erze big-data. Warszawa, Szkoła Główna Handlowa.
44. Przanowski, K. (2015). Credit scoring : Studia przypadków procesów biznesowych. Warszawa, Szkoła Główna Handlowa.
45. Przanowski, K. (2023). Credit scoring - automatyzacja procesu biznesowego - prezentacja do przedmiotu. Warszawa, Szkoła Główna Handlowa.
46. pydata.org. (2023). Pandas documentation. <https://pandas.pydata.org/docs/>, 2023.
47. python.org. (2023). math — mathematical functions. <https://docs.python.org/3/library/math.html>, 2023.
48. researchgate.net. (2017). 5 v's of big data. https://www.researchgate.net/figure/The-5V-of-Big-Data-Characteristics_fig1_321050765, październik 2017.
49. sas.com. (2018). Cztery typy uczenia maszynowego.

- https://www.sas.com/pl_pl/news/informacje-prasowe-pl/2018/cztery-typy-uczenia-maszynowego.html, 22.08.2018.
50. scikit learn.org. (2022). Category encoders. http://contrib.scikit-learn.org/category_encoders/, 2022.
 51. scikit learn.org. (2023a). sklearn.metrics.accuracy_score documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html, 2023.
 52. scikit learn.org. (2023b). sklearn.metrics.roc_auc_score documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html, 2023.
 53. scikit learn.org. (2023c). sklearn.metrics.roc_curve documentation. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html, 2023.
 54. scikit learn.org. (2023d). sklearn.model_selection.train_test_split documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, 2023.
 55. scoringexpert.pl. (2018). 8 ważnych informacji potrzebnych do zrozumienia nowej oceny punktowej, którą bik sprzedaje konsumentom. <http://scoringexpert.pl/2018/02/21/ocena-punktowa-bik-skala-do-100/>, 21.02.2018.
 56. seaborn.pydata.org. (2022). seaborn: statistical data visualization. <https://seaborn.pydata.org/>, 2022.
 57. Shi, Y., Sagduyu, Y., Davaslioglu, K., & Li, J. (2019). Generative adversarial networks for black-box api attacks with limited training data.
 58. Shi, Y., Sagduyu, Y., & Grushin, A. (2017). How to steal a machine learning classifier with deep learning. Rockville, MD 20855, USA, Intelligent Automation, Inc.,.
 59. Short, A., Pay, T. L., & Gandhi, A. (2019). Defending against adversarial examples. y Sandia National Laboratories, operated for the United States Department of Energy by National Technology Engineering Solutions of Sandia, LLC.
 60. Siddiqi, N. (2016). Credit risk scorecards developing and implementing intelligent credit scoring. New Jersey, John Wiley Sons, Inc.
 61. Statsoft. (2010). Metody skoringowe w biznesie i na-

- uce. https://media.statsoft.pl/_old_dnn/downloads/mo-dele_skoringowe_w_biznesie.pdf, 2010.
62. StatSoft. (2010). Zastosowanie metod scoringowych w działalności bankowej. https://media.statsoft.pl/_old_dnn/downloads/zast_met_skoringowych_w_dz_bankowej.pdf, 2010.
 63. statystyka.az.pl. (2021). Regresja logistyczna. <https://www.statystyka.az.pl/regresja-logistyczna.php>, 17.08.2021.
 64. Surma, J. (2020). Prezentacja pt. hakowanie sztucznej inteligencji. Warszawa, Szkoła Główna Handlowa.
 65. techtarget.com. (2021). 5 v's of big data. <https://www.techtarget.com/searchdatamanagement/definition/5-Vs-of-big-data>, marzec 2021.
 66. Thomas, L., Edelman, D., & Crook, J. (2002). Credit scoring and its applications. Philadelphia, Society for Industrial and Applied Mathematics.
 67. Thonabauer, G., & Nosslinger, B. (2004). Guidelines on credit risk management. credit approval process and credit risk management. Oesterreichische Nationalbank and Austrian Financial Market Authority.
 68. totalmoney.pl. (2020). Ocena punktowa bik-u – jaka wartość scoringu bik-u jest dobra i daje szansę na kredyt? <https://www.totalmoney.pl/artykuly/179147,kredyty-gotowkowe,ocena-punktowa-bik-u—jaka-wartosc-scoringu-bik-u-jest-dobra-i-daje-szanse-na-kredyt,1,1>, 03.10.2020.
 69. towardsdatascience.com. (2021). What is adversarial machine learning? <https://towardsdatascience.com/what-is-adversarial-machine-learning-dbe7110433d6>, 12.07.2021.
 70. Weston, L. (2012). Your credit score. New Jersey, Pearson Education, Inc.
 71. Wikipedia. (2022). Fico. <https://en.wikipedia.org/wiki/FICO>, 24.12.2022.
 72. Wyśiński, P. (2013). Zastosowanie scoringu kredytowego w bankowości. Gdańsk, Uniwersytet Gdański.
 73. zephyrnet.com. (2022). Co to jest kontradyktoryjne uczenie maszynowe? <https://zephyrnet.com/pl/co-to-jest-kontradyktoryjne-uczenie-maszynowe/>, 3.03.2022.

Spis rysunków

| | | |
|----|---|----|
| 1 | Przykład cechy wykorzystanej w Credit Scoring(bankier.pl, 2012) | 7 |
| 2 | Diagram oceny wiarygodności kredytowej według firmy FICO(forbes.com, 2021) | 9 |
| 3 | Fragment kalkulacji ze strony calcxml.com (calcxml.com, 2023) | 11 |
| 4 | Wizualizacja oceny punktowej w BIK (źródło opracowanie własne) | 11 |
| 5 | Krzywa Profit zależna od mocy predykcyjnej(Przanowski, 2023) | 16 |
| 6 | Krzywa ROC i jej możliwe warianty(Gajowniczek et al., 2014) | 17 |
| 7 | Schemat 5V Big Data(researchgate.net, 2017) | 19 |
| 8 | Drzewo decyzyjne zastosowane do kategoryzacji potencjalnych kredytobiorców pod kątem ryzyka kredytowego (Bujak, 2008) | 23 |
| 9 | Podstawowe techniki Uczenia Maszynowego(Mamczur, 2019) | 28 |
| 10 | Przykład ataku na system rozpoznawania obrazów (openai.com, 2017) . . . | 30 |
| 11 | Widok z kamery przedniej samochodu autonomicznego. Właściwie rozpoznany znak STOP (Surma, 2020) | 30 |
| 12 | Widok z kamery przedniej samochodu autonomicznego. Niewłaściwie rozpoznany znak STOP (Surma, 2020) | 31 |
| 13 | Widok z kamery przedniej samochodu autonomicznego. Znak STOP błędnie rozpoznany jako znak bezwzględnego pierwszeństwa przy skręcie w lewo (Surma, 2020) | 31 |
| 14 | Nowe ataki na Uczenie Maszynowe (Pawlicki, 2020) | 34 |
| 15 | Proces uczenia według ASSL4CCR (Feng B., 2021) | 37 |
| 16 | Przykłady obrazów wykorzystanych w ataku na model głębokiej sieci neuronowej firmy MetaMind(Papernot et al., 2017) | 39 |
| 17 | Dwa histogramy obrazujące rozkład liczby znaków wiadomości w zależności od liczby znaków dla treści spam oraz tekstów zaufanych(Kuchipudi et al., 2020) | 43 |
| 18 | Fragment kodu odpowiadający za import zbioru danych (źródło opracowanie własne) | 47 |
| 19 | Fragment kodu odpowiadający za ograniczenie zaimportowanego zbioru danych do produktu gotówkowego (źródło opracowanie własne) | 47 |

| | | |
|----|--|----|
| 20 | Fragment kodu odpowiadający za definicję zmiennej celu jako parametr globalny (źródło opracowanie własne) | 48 |
| 21 | Fragment kodu odpowiadający za analizę liczbową rozkładu wartości zmiennej celu (źródło opracowanie własne) | 48 |
| 22 | Fragment kodu odpowiadający za usunięcie braków danych zlokalizowanych w zmiennej celu, oraz posortowanie zbioru (źródło opracowanie własne) . . | 48 |
| 23 | Fragment kodu odpowiadający za wyfiltrowanie wybranych zmiennych objaśniających (źródło opracowanie własne) | 49 |
| 24 | Fragment kodu odpowiadający za wstępną analizę typów wybranych zmiennych objaśniających (źródło opracowanie własne) | 49 |
| 25 | Rozkład zmiennej 'act_age' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne) | 50 |
| 26 | Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne) | 51 |
| 27 | Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' na wykresie wiolinowym (źródło opracowanie własne) | 52 |
| 28 | Rozkład zmiennej 'act_call_cc' w zależności od wartości zmiennej 'default12' na wykresie pudełkowym (źródło opracowanie własne) | 52 |
| 29 | Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' (źródło opracowanie własne) | 53 |
| 30 | Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' na wykresie wiolinowym (źródło opracowanie własne) | 54 |
| 31 | Rozkład zmiennej 'act_ccss_n_statC' w zależności od wartości zmiennej 'default12' na wykresie pudełkowym (źródło opracowanie własne) | 54 |
| 32 | Fragment kodu odpowiadający za instalację niezbędnych pakietów (źródło opracowanie własne) | 55 |
| 33 | Fragment kodu odpowiadający za import niezbędnych bibliotek (źródło opracowanie własne) | 56 |
| 34 | Fragment kodu realizujący binowanie zmiennych jakościowych (źródło opracowanie własne) | 58 |
| 35 | Fragment kodu realizujący podział na zbiory treningowy i testowy (źródło opracowanie własne) | 58 |

| | | |
|----|--|----|
| 36 | Fragment kodu realizujący przygotowanie struktur danych dla modelu XGBoost (źródło opracowanie własne) | 59 |
| 37 | Analiza statystyki Feature Importance dla modelu wstępnego (źródło opracowanie własne) | 60 |
| 38 | Kod realizujący iteracyjne trenowanie kolejnych modeli celem wyboru najodpowiedniejszego (źródło opracowanie własne) | 62 |
| 39 | Wizualizacja wybranego modelu krzywymi ROC (źródło opracowanie własne) | 63 |
| 40 | Analiza Feature Importance dla finalnego modelu (źródło opracowanie własne) | 64 |
| 41 | Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 1 (źródło opracowanie własne) | 68 |
| 42 | Fragment kodu realizujący procesowanie danych celem weryfikacji hipotezy 1 (źródło opracowanie własne) | 69 |
| 43 | Wykres wartości współczynnika Giniego dla zbioru treningowego i testowego dla hipotezy 1 (źródło opracowanie własne) | 70 |
| 44 | Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 2 (źródło opracowanie własne) | 71 |
| 45 | Fragment kodu realizujący procesowanie danych celem weryfikacji hipotezy 2 (źródło opracowanie własne) | 72 |
| 46 | Wykres wartości współczynnika Giniego dla zbioru treningowego i testowego dla hipotezy 2 (źródło opracowanie własne) | 73 |
| 47 | Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 3 (źródło opracowanie własne) | 75 |
| 48 | Fragment kodu realizujący losowanie próby 100 dłużników, zakwalifikowanych przez model finalnych jako defaulty, celem weryfikacji hipotezy 3 (źródło opracowanie własne) | 76 |
| 49 | Fragment kodu realizujący generację Wrogich Próbek dla zmiennej 'act_call_cc', celem weryfikacji hipotezy 3 (źródło opracowanie własne) | 76 |
| 50 | Fragment kodu realizujący odpytywanie modelu finalnego wygenerowanymi wrogimi danymi, celem weryfikacji hipotezy 3 (źródło opracowanie własne) | 77 |
| 51 | Fragment kodu realizujący utworzenie ramek danych ze zbioru wejściowego do potrzeb weryfikacji hipotezy 4 (źródło opracowanie własne) | 78 |

| | | |
|----|--|----|
| 52 | Fragment kodu realizujący losowanie próby 100 dłużników, zakwalifikowa- nych przez model finalnych jako defaulty, dla zmiennej 'act_age', celem weryfikacji hipotezy 4 (źródło opracowanie własne) | 79 |
| 53 | Fragment kodu realizujący generację Wrogich Próbek dla zmiennej 'act_cins_seniority', celem weryfikacji hipotezy 4 (źródło opracowanie własne) | 79 |
| 54 | Fragment kodu realizujący odpytywanie modelu finalnego wygenerowanymi wrogimi danymi, dla zmiennej 'act_cins_seniority', celem weryfikacji hipo- tezy 4 (źródło opracowanie własne) | 80 |

Spis tabel

| | | |
|----|--|----|
| 1 | Interpretacja oceny punktowej BIK(scoringexpert.pl, 2017) | 13 |
| 2 | Klasyczna karta oceny punktowej(Przanowski, 2023) | 18 |
| 3 | Klasyczna karta oceny punktowej(Feng B., 2021) | 37 |
| 4 | Wrogie próbki zastosowane na atakowanym modelu filtra antyspamowego (Kuchipudi et al., 2020) | 41 |
| 5 | Opis zmiennych oraz grup zmiennych w zbiorze danych (źródło opracowanie własne) | 46 |
| 6 | Podstawowe statystyki zmiennej 'act_age' (źródło opracowanie własne) | 50 |
| 7 | Podstawowe statystyki zmiennej 'act_call_cc' (źródło opracowanie własne) . . | 51 |
| 8 | Podstawowe statystyki zmiennej 'act_ccss_n_statC' (źródło opracowanie wła- sne) | 53 |
| 9 | Parametry funkcji 'train' z biblioteki 'xgboost' zastosowane do wyboru od- powiedniego modelu (źródło opracowanie własne) | 62 |
| 10 | Parametry finalnego modelu XGBoost (źródło opracowanie własne) | 63 |
| 11 | Wartość współczynnika Giniego w zależności od procenta atakowanych da- nych dla hipotezy 1 (źródło opracowanie własne) | 70 |
| 12 | Wartość współczynnika Giniego w zależności od procenta atakowanych da- nych dla hipotezy 2 (źródło opracowanie własne) | 73 |
| 13 | Wyniki testów weryfikacyjnych hipotezy 3 (źródło opracowanie własne) . . | 77 |
| 14 | Wyniki testów weryfikacyjnych hipotezy 4 (źródło opracowanie własne) . . | 80 |

Załączniki

1. Plik VariablesDescription.xlsx z opisem zmiennych w zbiorze danych
2. Plik XGBoostModel_MK116111_1.html z kodem
3. Plik XGBoostModel_MK116111_1.ipynb z kodem
4. Plik abt_app.sas7bdat z danymi do modelu