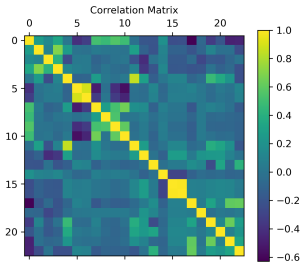
Raport z Projektu

Weronika Orzechowska

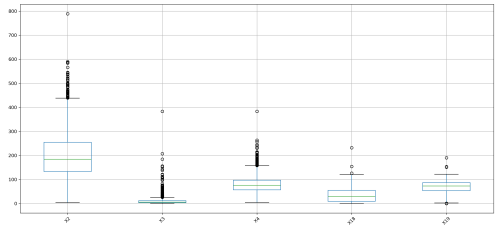
1 Eksploracja zbioru danych

Przygotowanie zbioru danych rozpoczęłam, zgodnie z załączonym opisem od zamiany wartości -9, -8, -7 na braki danych. W związku z tym, istotną częścią zbioru danych stały się braki danych, co wiązało się z koniecznością zastosowania metod imputacji.

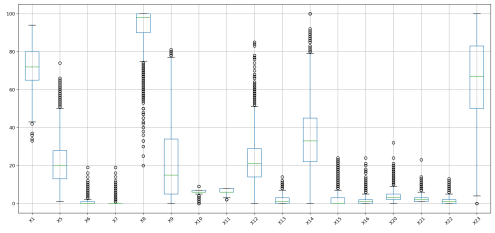
W kolejnym kroku przeanalizowałam macierz korelacji dla zmiennych, aby określić wstępnie, czy ewentualna selekcja zmiennych będzie mogła przynieść pewne korzyści. W macierzy (rys. 1) widzimy zarówno silne dodatnie korelacje (żółte pola) jak i silne ujemne korelacje (fioletowe pola), co wskazuje na obecność zmiennych powiązanych. Analiza macierzy wskazuje na konieczność selekcji zmiennych.



Rysunek 1: Macierz korelacji zmiennych



Rysunek 2: Wykres box plot dla zmiennych X2, X3, X4, X18, X19



Rysunek 3: Wykres box plot dla pozostałych zmiennych

Kolejnym etapem było przeanalizowanie wartości zmiennych w zbiorze danych. W tym celu przygotowałam wykresy typu box plot (rys. 2,3), które pokazały znaczną obecność wartości odstających. W związku z tym w dalszych etapach do skalowania zmiennych stosowałam metody dostosowane do zmiennych z brakami danych. Histogramy dla poszczególnych zmiennych również wykazały potrzebę skalowania, przez występującą skośność.

Zgodnie z opisem danych, zamieniłam typ zmiennych X10, X11 na category, aby w dalszej części dostosować dla nich odpowiednie transformacje.

W wyniku analizy zmiennej celu otrzymałam, że klasy mają rozkład równomierny - klasa 'Bad' stanowi 53% wszystkich obserwacji, w związku z czym nie musimy stosować dodatkowych technik pozwalających zachować rozkład zmiennej przy podziałach. Ponadto, zgodnie z opisem danych 1 = 'Bad'.

2 Przygotowanie zbioru danych

Zbiór danych podzieliłam na testowy i treningowy w proporcji 3:7 (30% danych to dane testowe). Z ustawienie parametru random_state na 327461. Wartość tą stosowałam za kązdym razem, gdy ustawiałam w modelu ten parametr.
Po eksploracji danych w punkcie 1. oczywistym wnioskiem jest, że wymagany jest preprocessing danych. Uwzględniając wnioski z eksploracji danych, przy transformacji zmiennych numerycznych wzięłam pod uwagę: SimplieImputer, KNNImputer, StandardScaler, RobustScaler, PowerTransformer, QuantileTransformer. Dla zmiennych kategorycznych zastosowałam SimpleImputer ze strategią 'most frequent' oraz OneHotEncoder.

3 Wstępna ocena modeli

Ocena modeli, zgodnie z opisem projektu będzie w głównej mierze oparta na miarze balanced accuracy, która zgodnie z podanym wzorem została zaimplementowana w funkcji balanced_accuracy(y, pred)

4 Analiza modeli

4.1 Drzewa decyzyjne

Model drzew decyzyjnych wykorzystałam do przetestowania różnych rodzajów skalowania i imputacji. Dodatkowo wykorzystałam siatkę parametrów: głębokość z przedziału 5-12 oraz minimalna ilość obserwacji w liściu: 3-9.

W wyniku tego eksperymentu otrzymałam, że imputacja KNNImputer wypada lepiej oraz widać poprawę przy stosowaniu skalowania dostosowanego do zmiennych z obecnymi wartościami odstającymi. Warto jednak w tym miejscu zaznaczyć, że KNNImputer wiązał się ze znacznie dłuższym czasem obliczeń oraz, że poprawa wartości balanced accuracy widoczna była dopiero na trzecim miejscu po przecinku.

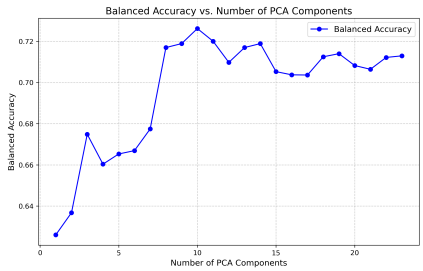
Dlatego też w dalszych eksperymentach stosowałam transformacje numeryczną z imputacją SimpleImputer oraz skalowaniem RobustScaler. Dla modeli wykazujących poprawę jakości predykcyjnej przeprowadzałam ponownie testy z różnymi rodzajami imputacji i skalowania.

4.2 Regresja Logistyczna

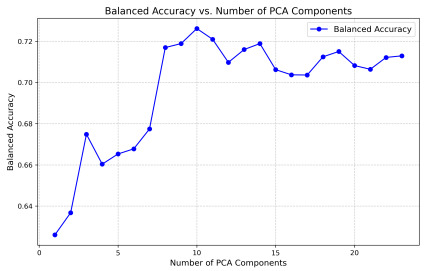
Ze względu na prostotę modelu jak również jego szerokie zastosowanie w faktycznej ocenie ryzyka kredytowego kolejnym analizowanym modelem była regresja logistyczna.

Już zwykły model bez kary wykazał poprawę miary jakości względem drzew decyzjnych: 0.717.

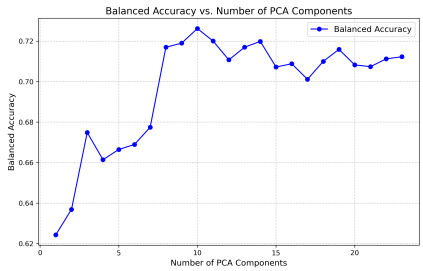
W kolejnym eksperymencie wykorzystałam PCA w celu redukcji wymiarowości danych, aby przetestować wpływ tej techniki na jakość modelu. Przeprowadziłam trzy eksperymenty: nakładając odpowiednio brak kary, karę l1 i l2 oraz testując współczynniki C : 0.2 - 1 (co 0.2).



Rysunek 4: penlaty = None



Rysunek 5: penalty = l2



Rysunek 6: penalty = l1

	penalty = None	**penalty = l2**	**penalty = l1**
PCA	0.726	0.726	0.726
no PCA	0.717	0.711	0.712

Dla każdego z wykresów otrzymaliśmy, że najlepszy wynik miary balanced accuracy był dla pierwszych 10 komponentów. Widzimy więc, że dodanie PCA poprawia miarę balanced accuracy.

4.3 K najbliższych sąsiadów

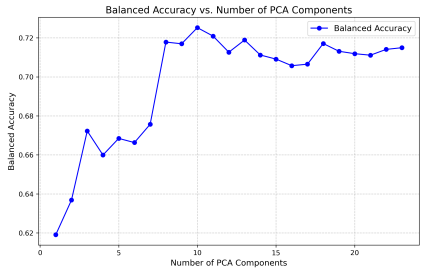
Kolejnym analizowanym modelem był model k najbliższych sąsiadów. Rozpoczęłam od zwykłego modelu z przygotowaniem danych preprocessorem, uzyskując niską miarę balanced accuracy na poziomie 0.65. W następnym kroku zastosowałam inny typ imputacji: KNNIpmuter, co nieznacznie poprawiło miarę balanced accuracy: 0.657. Następnie dodałam kolejny krok do pipeline - selektor, co znowu zwiększyło miarę ba: 0.673. W ostatnim kroku, przetestowałam wpływ PCA na model KNN, ale nie otrzymaliśmy lepszych wyników, niż stosując selektor. Metoda k najbliższych sąsiadów nie dała satysfakcjonujących rezultatów.

4.4 Bagging

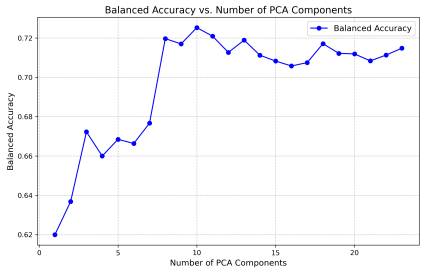
Kolejnym modelem, który analizowałam był Bagging. Początkowo zastosowałam jako estymator KNeighborsClassifier, ale analogicznie jak w punkcie 4.3 nie uzyskałam z nim zadowalających rezultatów.

Biorąc pod uwagę stosunkowo dobre wyniki, które uzyskałam korzystając z regresji logistycznej, w dalszych etapach badania baggingu zdecydowałam właśnie wykorzystać jak estymator Regresję Logistyczną. Ponownie przetestowałam

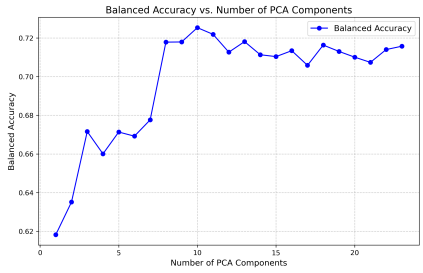
regresję logistyczną bez kary oraz z regularyzacją l1 i l2. Eksperyment przeprowadziłam, wykorzystując PCA oraz badając taką samą siatkę parametru C jak w punkcie 4.2. Przeprowadzając tą analizę, najlepszy wynik otrzymałam dla modelu



Rysunek 7: penlaty = None



Rysunek 8: penalty = l2



Rysunek 9: penalty = l1

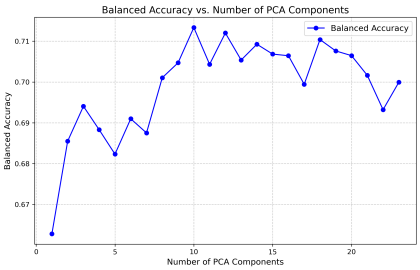
z regularyzacją l1, gdzie balanced accuracy osiągnęło 0.7253, co ze wszystkich analizowanych do tej pory modeli było najlepszym wynikiem.

Korzystając z tego wyniku i chcąc polepszyć wartość miary, powiększyłam siatkę parametrów o wartość liczby estymatorów dla baggingu: [10, 20, 40, 60, 80, 100], co w połączeniu z Regresją Liniową z karą l1 i dopasowanym parametrem C oraz redukcją wymiarowości za pomocą PCA, pozwoliło mi na uzyskanie wyniku balanced accuracy: 0.7252.

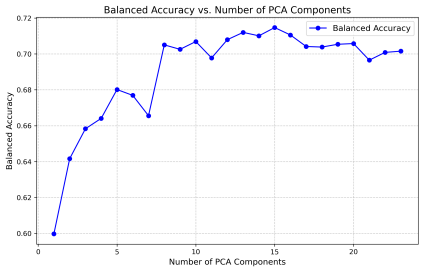
4.5 Las losowy, Extra Trees, Boosting

Dla lasu losowego, stosując selektor zmiennych oraz siatkę hiperparametrów, w której optymalizowałam wartości liczby estymatorów, maksymalnej głębokości, minimalnej liczby obserwacji w liściu czy krytrium podziału otrzymałam wartość 0.714 dla balanced accuracy. Eskperyment ponowiłam zmieniając karę na None w Regresji Logistycznej w selektorze, co pogorszyło miarę balanced accuracy. Analogiczny test z karą l1, przyniósł nieznaczne pogorszenie i uzyskaliśmy w ten sposób wartość miary 0.713.

W następnym kroku chciałam zbadać wpływ PCA dla lasów losowych. Wykorzystałam najoptymalniejsze wartości hiperparametrów, które uzyskałam we wcześniejszej analizie z selektorem zmiennych, wybierajac maksymalną głębokość: 8, minimalna liczbę liści: 3 oraz liczbę estymatorów: 60. Eksperyment ponowiłam bez dobrania potencjanie optymalnych hiperparametrów.



Rysunek 10: dobrane hiperparametry



Rysunek 11: hiperparametry domyślne

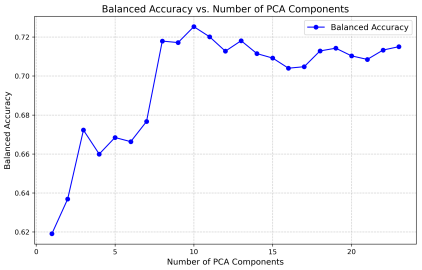
W eksperymentach zaprezentowanych na rys. 10, 11 otrzymaliśmy maksymalną wartość balanced accuracy odpowiednio 0.713 oraz 0.715. Zatem PCA, bez dobranych optymalnych hiperparametrów pozwoliło uzyskać lepszą wartość miary balanced accuracy niż zastosowanie selektora z Regresją Logistyczna i optymalnymi hiperparametrami dla lasu losowego. Analogiczne eksperymenty przeprowadziłam tez dla ExtraTreesClassifier oraz Boostingu, ale nie uzsykałam w ten sposób wyników polepszających miarę balanced accuracy, dlatego też pominę ich dokładniejszy opis.

4.6 Voting

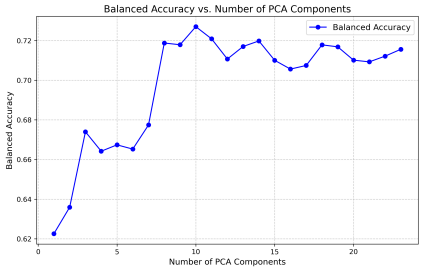
Ostatnią analizą, którą przeprowadziłam były modele z wykorzystaniem VotingClassifier. Na podstawie przeprowadzonych powyżej analiz wybrałam dwie najskuteczniejsze metody - regresję logistyczną oraz bagging z estymatorem będącym regresją logistyczną.

Najpierw przygotowałam pipeline zawierający jedynie preprocessing oraz voting, co dało balanced accuracy o wartości 0.716. W następnym kroku dodałam selekcję zmiennych z SelectFromModel z estymatorem będącym regresją logistyczną, co zmniejszyło wartość balanced accuracy do 0.709.

W kolejnym kroku połączyłam voting z PCA (rys. 12), co dało znacznie lepsze rezultaty: dla 10 komponentów PCA uzyskałam wartość balanced accuracy 0.724.



Rysunek 12: pierwszy model voting



Rysunek 13: drugi model voting

W następnym kroku do VotingClassifier dodałam model lasu losowego, który w punkcie 4.5 uzyskład bardzo dobre wyniki dla domyślnych hiperparametrów. Ponownie przeprowadziłam takie same testy i najlepszy wynik uzyskałam w połączeniu z PCA (rys. 13) - balanced accuracy o wartości 0.727 dla 10 komponentów PCA.

4.7 Podsumowanie

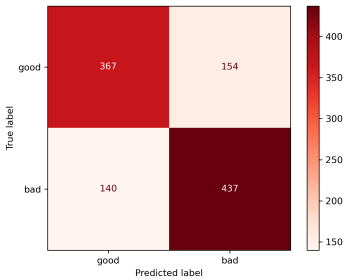
Na podstawie wszystkich wykonanych analiz jako najlepszy model wybrałam VotingClassifier z regresją logistyczną, lasem losowym i bagging, połączony z redukcją wymiarowości za pomocą PCA.

5 Analiza wybranego modelu

W pierwszym etapie przetestowałam dodatkowo różne typy imputacji i skalowania zmiennych dla wybranego modelu. W wyniku tego, otrzymałam, że najlepiej sprawdza się imputacja KNNImputer połączona ze skalowaniem RobustScaler(), czyli faktycznie scaler dopasowany do danych z dużą ilością wartości odstających pozwolił na uzyskanie lepszej predykcji. Ponadto otrzymałam, że model najlepiej sprawdza się dla pięciu komponentów PCA. W ten sposób otrzymałam balanced accuracy o wartości 0.731

Model klasyfikacji osiągnął stabilne wyniki zarówno na zbiorze treningowym, jak i testowym, co sugeruje jego dobre dopasowanie oraz brak oznak nadmiernego dopasowania (overfittingu). Dokładność wynosi 73,3% na zbiorze treningowym i 73,2% na zbiorze testowym, co wskazuje na spójność wyników i zdolność modelu do generalizowania na nowe dane. Zbalansowana dokładność na poziomie 73% oznacza, że model dobrze radzi sobie z niezbalansowanymi danymi, traktując obie klasy w sposób proporcjonalny. Czułość, wynosząca 78,2% na zbiorze treningowym i 75,7% na testowym, jest wysoka, co jest istotne w kontekście minimalizacji liczby fałszywie negatywnych klasyfikacji. Precyzja, która pozostaje na poziomie 72,9% w obu zbiorach, potwierdza skuteczność modelu w ograniczaniu fałszywie pozytywnych przypadków.

wartości metryk	zbiór testowy	zbiór treningowy
dokładność	0.732	0.733
czułość	0.757	0.782
precyzja	0.729	0.729
dokładność zbalansowana	0.731	0.73



Analiza macierzy pomyłek pokazuje, że model wykazuje równowagę w klasyfikacji klas “good” i “bad”, co dodatkowo potwierdza jego zbalansowana dokładność. Brak dużych różnic w wynikach między zbiorami treningowymi i testowymi wskazuje, że model jest dobrze dopasowany i nie przeucza się na danych treningowych. Równowaga pomiędzy czułością a precyzją sugeruje, że model sprawdzi się w sytuacjach, gdzie zarówno fałszywe alarmy, jak i fałszywe negatywy są istotne. Dzięki wysokiej zbalansowanej dokładności oraz równowadze pomiędzy czułością a precyzją model nadaje się do zastosowań w rzeczywistych scenariuszach, takich jak ocena ryzyka kredytowego.