Analityka i Eklsploracja Danych

Uczenie maszynowe

Artur Sobolewski

06.01.2023

1 Wstęp

1.1 Cel projektu

Celem projektu było wykonanie modelu predykcyjnego w zależności od problemu. Wbybrano dane o transakcjach dokonanych za pomocą karty kredytowej, na podstawie których model miał za zadanie sklasyfikować czy płatności była prawidłowa, czy należy uznać daną transakcję za fraud. Wykonanie modelu ma polegać na przebadaniu algorytmów klasyfikacji, dostosowując ich paramatery oraz na wstępnym przetworzeniu danych, jak np. feature selection lub też over/under-sampling.

Wybrany zbiór stanowił dodatkowe wyzwanie związane z dużym niezbalansowaniem (liczność klasy pozytywnej stanowiła mniej niż 0,2% wszystkich danych). W takich przypadkach różne klasyfikatory mogą mieć problem z generalizacją wiedzy, czy wyznaczeniu hiperpłaszczyzny separującej klasy. Z tego powodu należało przebadać i zastosować metody balansowania klas.

1.2 Przebieg realizacji

Badania dopasowania modeli bazowały na następujących miarach:

- FNR - False Negative Rate - stosunek liczby fałszywego sklasyfikwania transakcji jako prawidłowe do liczby transakcji będącymi prawdziwie fraudem,
- FPR - False Negative Rate - stosunek liczby fałszywego sklasyfikowania transakcji jako fraud do liczby transakcji będącymi prawdziwie prawidłowymi,
- Sensitivity - czułość modelu na klasę przyjętą za pozytywną - stosunek liczby prawidłowego sklasyfikwania transakcji jako fraud do liczby transakcji będącymi prawdziwie fraudem,
- Specificity - specyficzność modelu określająca na ile odróżnia on od siebie klasy - stosunek liczby prawidłowego sklasyfikowania transakcji jako prawidłowe do liczby transakcji będącymi prawdziwie prawidłowymi,
- ROCAUC - pole pod krzywą wykresu ROC (Receiver operating characteristic).

Zadanie dostosowanie modelu polegało na minimalizacji stopy błędów 'yes' → 'no' (FNR) przy utrzymaniu stopy błędów 'no' → 'yes' (FPR) poniżej 0,5%. Znalezienie najlepiej dopasowanego modelu opierało się również na maksymalizacji czułości modelu i wartości ROCAUC przy jak najwyższej specyficzności.

W badaniu uwzględnione zostały wybrane algorytmy klasyfikacji:

- LDA (Liniowa analiza dyskryminacyjna) i QDA (Kwadratowa analiza dyskryminacyjna),
- Drzewo decyzyjne,
- Random Forest,
- Naive Bayes,
- MLP (Perceptron wielowarstwowy),
- Regresja logistyczna,
- kNN (k najbliższych sąsiadów),
- SVM (Support Vector Machines).

Każdy z ww. algorytmów był dostrajany do jak najlepszej skuteczności w dalszym ciągu kierując się wynikami przyjętych miar. Przykładem takiego dostosowania jest głębokość i szerokość MLP lub miara odległości w metodzie kNN.

Kolejnym etapem projektu były próby rozwiązania problemu związaniego z niezbalansowaniem liczności klas. Zastosowano metody augumentacji danych tj. over-sampling mało licznej klasy pozytywnej (klasy świadczącej o fraudzie), under-sampling klasy bardziej licznej, czy algorytm smote. Po takich zmianach ponownie przeprowadzo badania skuteczności modeli i próby ich lepszego dospasowania.

Zbadany został również wpływ na skuteczność klasyfikacji stosując metody wyboru zmiennych (feature selection), dostosowywanie funkcji kosztu. Przeprowadzone zostało również łączenie (składanie) modeli - ensemble learning - według algorytmu np. AdaBoost.

2 Analiza danych

Zanim przystąpiono do analizy skuteczności klasyfikatorów, wcześniej przeprowadzono analizę danych uczących w celu poznania możliwych problemów z nimi związanych, czy i jak należy je wcześniej przetworzyć.

Rysunek 1 przedstawia kod wraz z wynikami pokazującymi informacje o danych o transakcjach za pomocą kart kredytowych. Dane składają się z 31 kolumn z czego ostatnia informuje do jakiej klasy należy pojedyncza transakcja a pozostałe są zmiennymi dla klasyfikatora. Z nich interpretowalne są jedynie kolumny *Time* oraz *Amount*, inne są wynikiem przeprowadzonej wcześniej metody PCA w celu redukcji wymiarowości i ukryciu danych o klientach (stąd wniosek, że wymiary powinny być ortogonalne, zgodnie z istotą działania PCA). Wszystkich rekordów w zbiorze danych jest 284807 a w nich jedynie 492 zaliczonych do klasy zawierające transakcje będące fraudem. Świadczy to wysokim niezbalansowaniu danych, co było jednym z przedmiotów badania.

```
dataframe = pd.read_csv("./data/creditcard.csv")
dataframe.head(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.219422	0.215153	69.99	0

5 rows × 31 columns

```
dataframe.shape
```

```
(284807, 31)
```

```
dataframe['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

Rysunek 1: Fragment danych i rozmiar zbioru danych

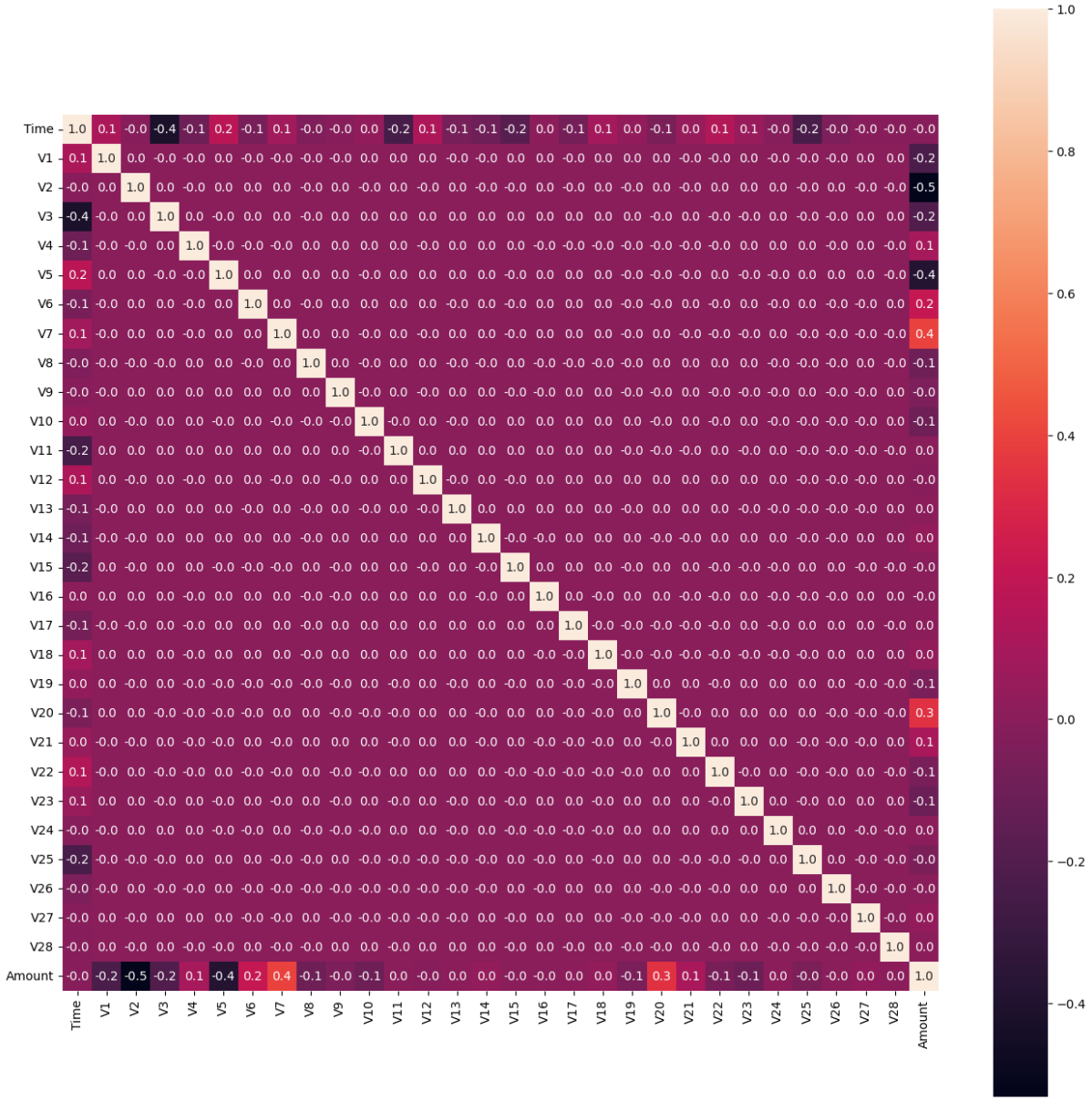
Dodatkowe informacje dostarcza również metoda *describe()* obiektu *DataFrame*. Przede wszystkim wyświetla ona obliczone dla każdej kolumny wartość średnią i standartowe odchylenia. Dzięki tym danym można określić czy zmienne mają spójny rozkład i co za tym idzie, czy należy je ustandaryzować. W przypadku badanego zbioru danych (Rys. 2) zmienne V1-V28 mają bardzo zbliżone rozkłady, jednak kolumny *Time* oraz *Amount* znacznie się od nich różnią. Dlatego przed dopasowywaniem klasyfikatorów przeprowadzono skalowanie wszystkich zmiennych za pomocą obiektu *StandardScaler* biblioteki *scikit-learn*.

```
dataframe.describe()
```

	Time	V1	V2	V3	V4	V5	V6	...	V28	Amount	Class
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	284807.000000	284807.000000
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	...	-1.227390e-16	88.349619	0.001727
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	...	3.300833e-01	250.120109	0.041527
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	...	-1.543008e+01	0.000000	0.000000
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	...	-5.295979e-02	5.600000	0.000000
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	...	1.124383e-02	22.000000	0.000000
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	...	7.827995e-02	77.165000	0.000000
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	...	3.384781e+01	25691.160000	1.000000

Rysunek 2: Charakterystyka zmiennych

Ostatnim etapem analizy zbioru danych treningowych było sprawdzenie korelacji zmiennych, a zatem ich niezależności liniowej. Motywacją takiej analizy było określenie czy na danych można przeprowadzić efektywną redukcję wymiarowości oraz na jakich algorytmach klasyfikacji warto się skupić i jak je dostosowywać do problemu. Rysunek 3 przedstawia wykres w formnie mapy ciepła opisujący wzajemną korelację każdej możliwej pary zmiennych za pomocą współczynnika korelacji Pearsona. Kolumny V1-V28 są liniowo niezależne i jedynie z niektórymi wymiarami kloumna *Time* i *Amount* osiąga wartość bezwzględną do -0.5. W takich przypadkach związek dwóch zmiennych osiąga maksymalny poziom jaki można napotkać w zbiorze, jednak i tak jest on bardzo słaby. Na tej podstawie można stwierdzić, że dane nie stanowią dobraj postawy do redunkcji wymiarów.



Rysunek 3: Korelacja wymiarów

3 Porównanie klasyfikatorów

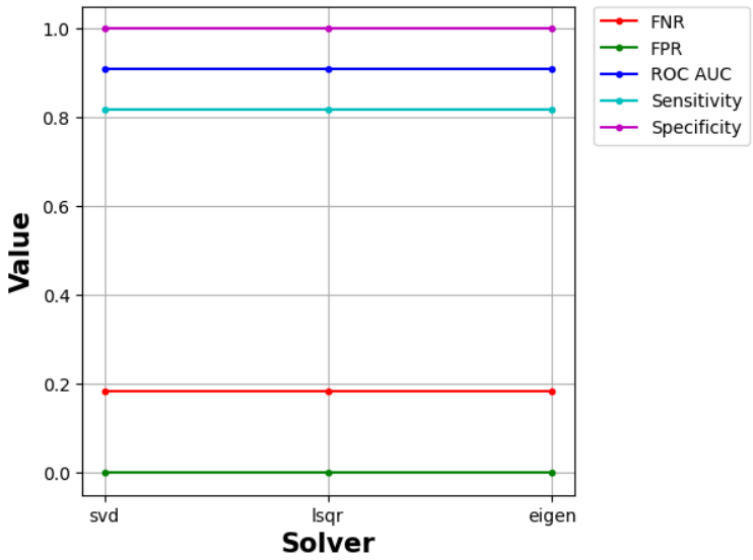
W tym punkcie przedstawione zostaną pomiary skuteczności modeli o domyślnych prarametrach, jakie określa dokumentacja *scikit-learn* oraz zebrane na drodze badania empirycznego polegającego na dostrajaniu ich hiperparametrów. Poniższa tabela 1 przedstawia wyniki osiągnięte przez klasyfikatory wymienione we wstępie. Zostały one uszeregowane wrosnącej na podstawie poziomu błędów FNR, tym samym odpowiada to malejącej kolejności pod względem miary *Sensitivity*. Najlepiej pod tym względem wypadły klasyfikatory QDA oraz Naive Bayess, jednak nie spełniają one przyjętego kryterium o maksymalnym poziomnie błędu FPR równym 0,5%. Dlatego uwzględniając to ograniczenie najlepiej wypadł algorytm kNN.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	11.4650 %	2.5631 %	0.9299	88.5350 %	99.8482 %
Naive Bayess (Gaussian)	14.0127 %	2.2786 %	0.9185	85.9873 %	99.8530 %
kNN	16.5605 %	0.0032 %	0.9172	83.4395 %	99.8606 %
Random Forest	17.1975 %	0.0043 %	0.9140	82.8025 %	99.8616 %
MLP	18.4713 %	0.0107 %	0.9076	81.5287 %	99.8638 %
LDA	18.4713 %	0.0160 %	0.9076	81.5287 %	99.8637 %
Decision Tree	22.2930 %	0.0437 %	0.8883	77.7070 %	99.8701 %
Logistic Regression	28.6624 %	0.0181 %	0.8566	71.3376 %	99.8808 %
SVM	31.2102 %	0.0011 %	0.8439	68.7898 %	99.8850 %

Tabela 1: Porównanie klasyfikatorów o dymyślnych parametrach

3.1 LDA - Liniowa Analiza Dyskryminacyjna

W przypadku Liniowej Analizy Dyskryminacyjnej najczęściej dostosowuje się hiperparametry *solver* (estymator) i *shrinkage* (prametr estymatorów regularyzujący macierze kowariancji i poprawia ich stabilność). Estymatory jakie zostały sprawdzone to: *svd*, *lsqr* oraz *eigen*. Wyniki jakie osiągnęły estymatory przedstawia rysunek 4. Dla każdego z nich wynik się nie zmienił.

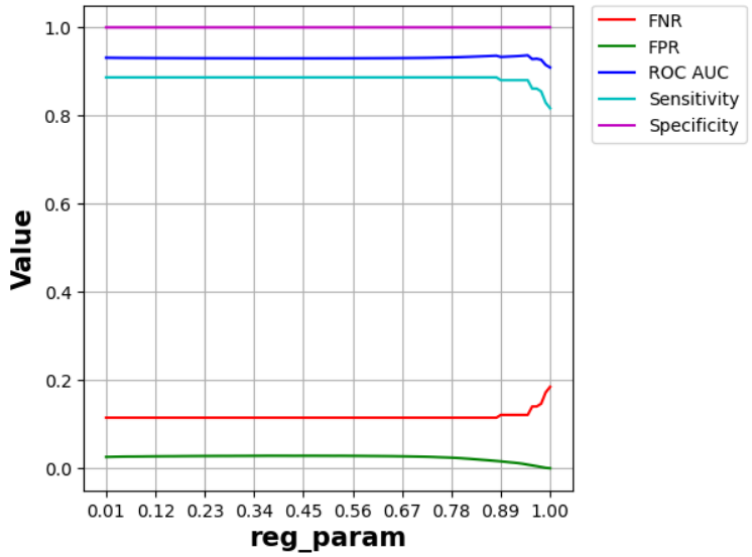


Rysunek 4: Wykres skuteczności LDA w zależności od estymatora

Dostosowując parametr *shrinkage* przy zastosowaniu estymatorów *lsqr* oraz *eigen* rónież nie osiągnięto żadnych zmian pod kątem skuteczności osiągając wartości takie jakie zostały przedstawione w tabeli 1 dla tego algorytmu.

3.2 QDA - Kwadratowa Analiza Dyskryminacyjna

Próbując zmniejszyć zbyt wysoki poziom błędów FPR klasyfikatora QDA dostosowywano parametr *reg_param*, który reguluje oszacowania kowariancji dla poszczególnych klas. Pzyjmuje on wartości z zakresu [0-1] i jak można zauważyć z wykresu przedstawionego na rysunku 5 wykresu skuteczność poprawia się dopiero od wartości 0,78 tego parametru. Najlepszy wynik został osiągnięty dla parametru równego 0,97. Uzyskany wynik dla tego parametru przedstawia tabela 2.



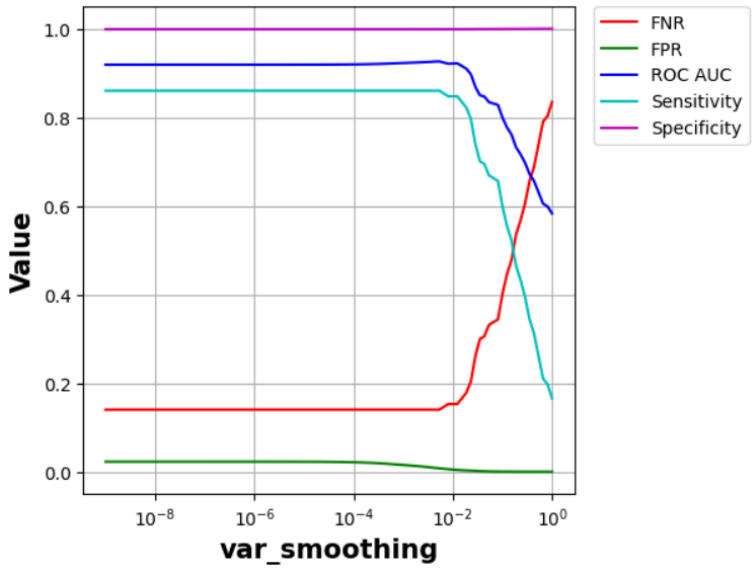
Rysunek 5: Wykres skuteczności QDA w zależności od reg_param

FNR	FPR	ROC AUC	Sensitivity	Specificity
14.0127 %	0.4881 %	0.9275	85.9873 %	99.8556 %

Tabela 2: Wartości miar klasyfikatora QDA o najlepszej skuteczności

3.3 Naive Bayess

Przy zastosowaniu domyśnych parametrów klasyfikator *GaussianNB* osiągną zbyt wysoki poziom błędu FPR. Możliwie lepsze dopasonie można było uzyskać przez dostosowywanie parametru *var_smoothing* - część wariancji zmiennej o największej wariancji dodawana do pozostałych. Przy wartości 0,01 tego parametru uzyskano najlepszą skuteczność (Tabela 3). Przebieg zmian skuteczności klasyfikatora w zależnści tego parametru przedstawia wykres na rysunku 6.



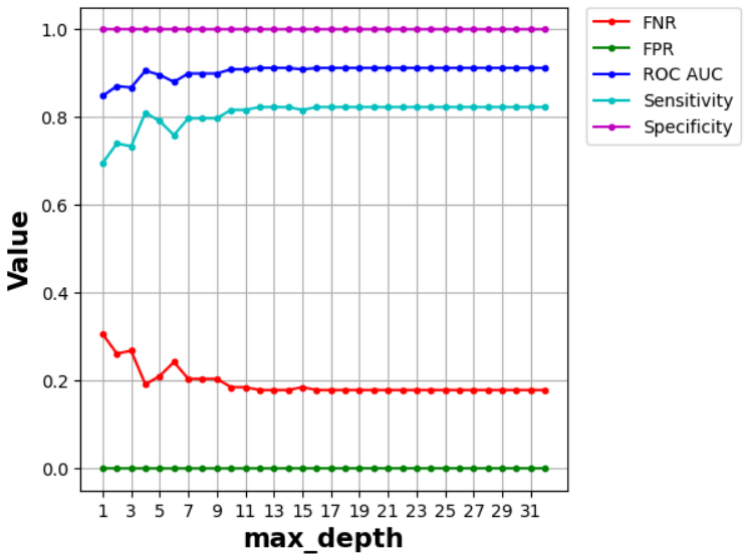
Rysunek 6: Wykres skuteczności Naive Bayess w zależności od var_smoothing

FNR	FPR	ROC AUC	Sensitivity	Specificity
15.2866 %	0.4071 %	0.9215	84.7134 %	99.8579 %

Tabela 3: Wartości miar klasyfikatora Naive Bayess (Gaussian) o najlepszej skuteczności

3.4 Drzewo Decyzyjne

Pierwszym i przynoszącym jedyną poprawę był parametr *max_depth* określający maksymalną wysokość drzewa decyzyjnego. Przebadano również ustawnia *min_samples_split* (wyznacznik przerwania daleszego podziału na węzły) oraz *max_features* (maksymalna liczba cech branych pod uwagę przy podziale węzła). Najlepszą skuteczność (wyniki w tabeli 4) klasyfikatora uzyskano dla *max_depth* równego 12, zachowując pozostałe hiperparametry jako domyślne. Dostosowując inne parametry otrzymywano skuteczności bardzo zbliżne do uzyskanej przy ustawieniach domyślnych. Poniższy rysunek 7 przedstawia jakie skuteczności osiągał klasyfikator dla różnych maksymalnych głębokości drzewa.



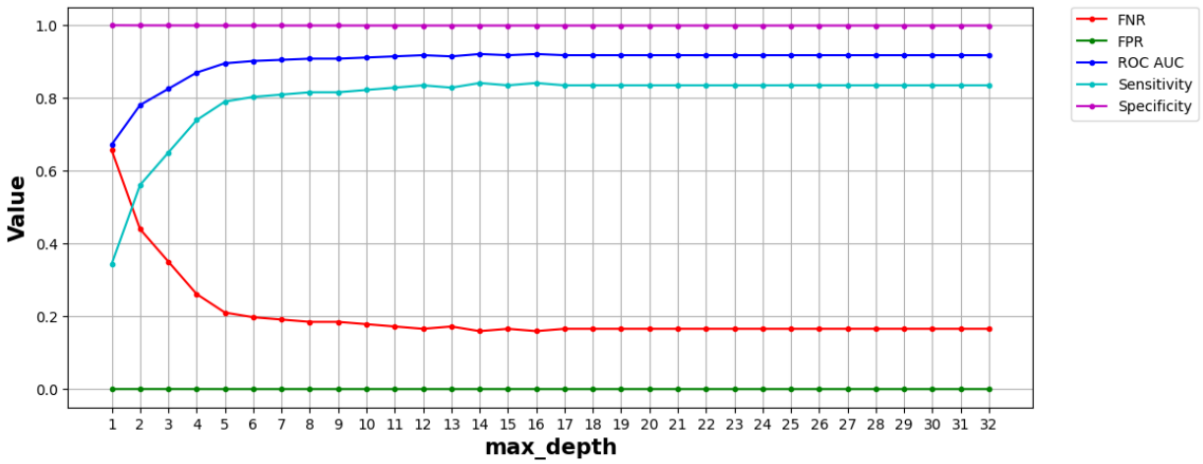
Rysunek 7: Wykres skuteczności Drzewa Decyzyjnego w zależności od max_depth

FNR	FPR	ROC AUC	Sensitivity	Specificity
17.8344 %	0.0171 %	0.9107	82.1656 %	99.8627 %

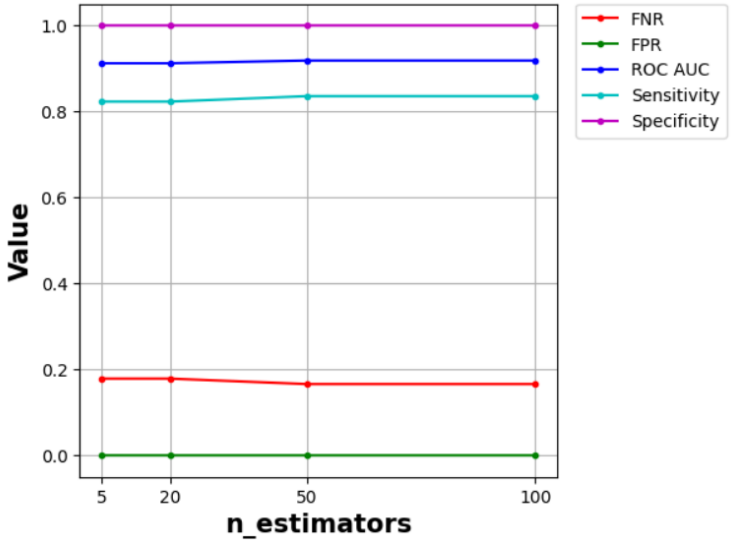
Tabela 4: Drzewo decyzyjne

3.5 Random Forest

Tak jak w przypadku Drzewa Decyzyjnego kluczownym parametrem klasyfikatora Random Forest jest *max_depth* ograniczający z góry wykokość każdego z drzew składowych. Przebadano również parametry *n_estimators* - liczba drzew składowych, *min_samples_split*. Z wykresu z rysunku 8 wynika, że najlepszy wynik osiągnięto już przy wysokości drzew nie większej niż 14. Z kolejnego wykresu z rysunku 11 można zawuażyć, że klasyfikator osiąga taką samą i najlepszą uzyskaną skuteczność dla 50 estymatorów składowych, jak w przypadku, gdy jest ich 100. Wyniki maiar klasyfikatora (dla *n_estimators* = 50 i *max_depth* = 14) o najlepszej skuteczności zostały zawarte w tabeli 5. Dla innych konfiguracji parametrów takich jak *min_samples_split* nie uzyskano żadnych przydatnych wyników.



Rysunek 8: Wykres skuteczności Random Forest w zależności od max_depth



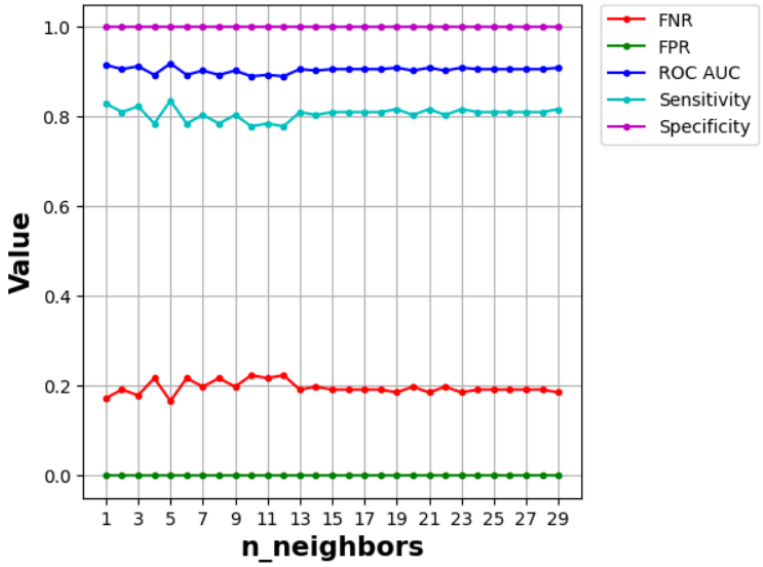
Rysunek 9: Wykres skuteczności Random Forest w zależności od n_estimators

FNR	FPR	ROC AUC	Sensitivity	Specificity
15.9236 %	0.0043 %	0.9204	84.0764 %	99.8595 %

Tabela 5: Wartości miar klasyfikatora Random Forest o najlepszej skuteczności

3.6 kNN - k Najbliższych Sąsiadów

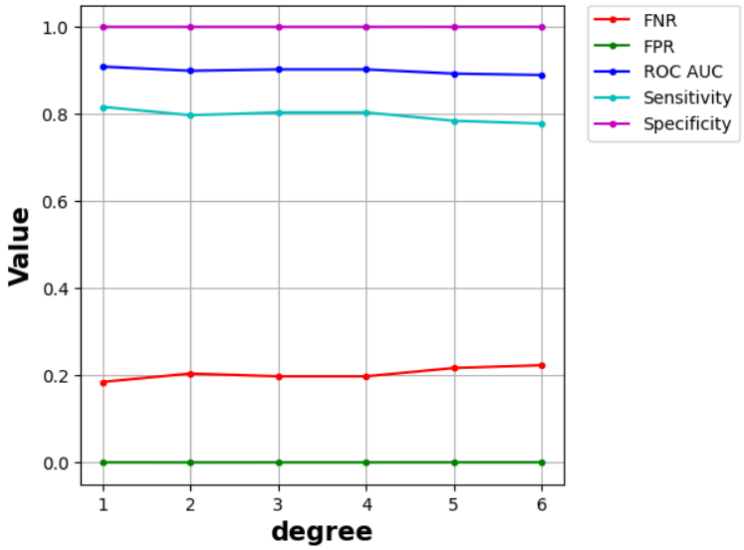
W przypadtku tej metody przetestowane zostały różne wartości dwóch parametrów: *n_neighbors* - liczba najbliższych sądziadów brana pod uwagę przy klasyfikacji - oraz *leaf_size*. W obu przypadkach najlepszymi parametrami jakie uzyskano były parametrami domyślnymi w blibliotce *scikit-learn*. Dlatego nie uzyskano żadniej poprawy w odniesieniu do wyniku zawartwgo w tabeli 1. Sprawdzono również ustawienia klasyfikatora z parametrem *weight = 'distance'* oraz z cosinusową miarą odległości, uzyskano znacznie gorsze wyniki niż przy domyślnych ustawieniach.



Rysunek 10: Wykres skuteczności kNN w zależności od n_neighbors

3.7 SVM - Supported Vector Machine

Dla algorytmu SVM określa się m.in. typ jądra, który ma być użyty (*kernel*) oraz *C* - parametr regularyzacji określający karę przy dopasowyaniu klasyfikatora. Dodatkowo sprawdzono parametr *degree* dla jądra *poly*, w wyniku czego uzyskano najlepsze ustawienie ze wszystkich jakie przetestowano (przy *degree = 1* - tabela 6). W pozostałych przypadkach również udało się uzyskać lepsze skuteczności niż przy ustawieniach domyślnych.



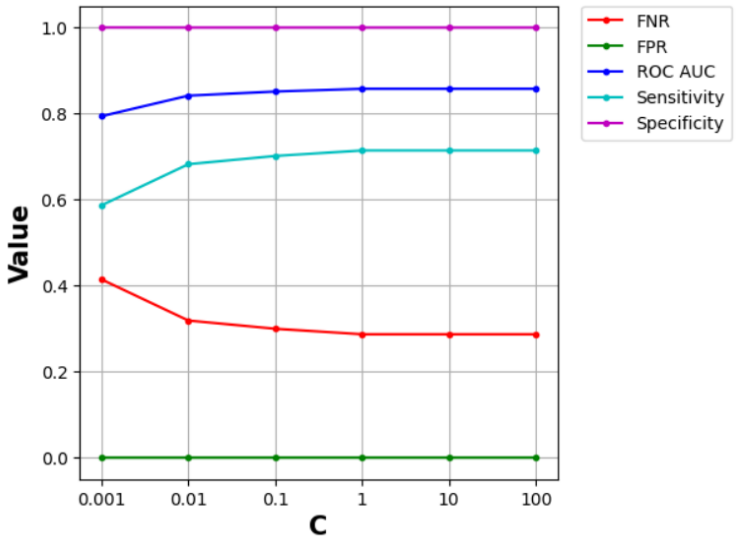
Rysunek 11: Wykres skuteczności SVM w zależności od degree

FNR	FPR	ROC AUC	Sensitivity	Specificity
18.4713 %	0.0181 %	0.9076	81.5287 %	99.8637 %

Tabela 6: Wartości miar klasyfikatora SVM o najlepszej skuteczności

3.8 Regresja Logistyczna

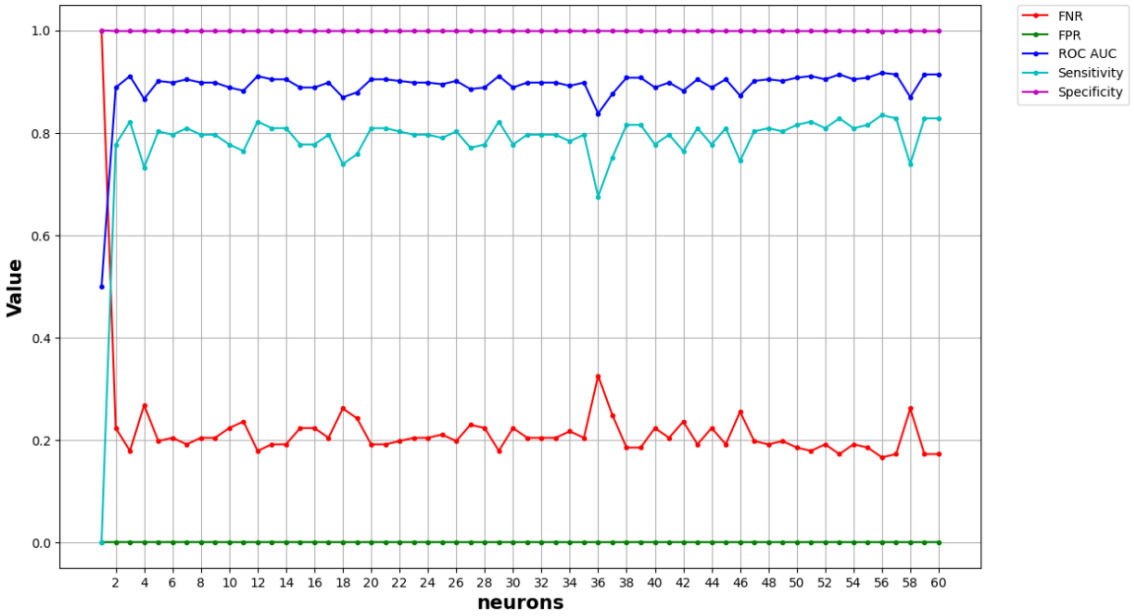
Podobnie jak w przypadku SVM przy Regresji Logistycznej również można dostosowywać parametr regularyzacji *C*. Przy testowaniu zekresu wartości 0.001-100 parametru *C* uzyskano najlepszą skuteczność dla wartości 1, która jest domyślną wartością. W związku z tym nie uzyskano żadnej poprawy.



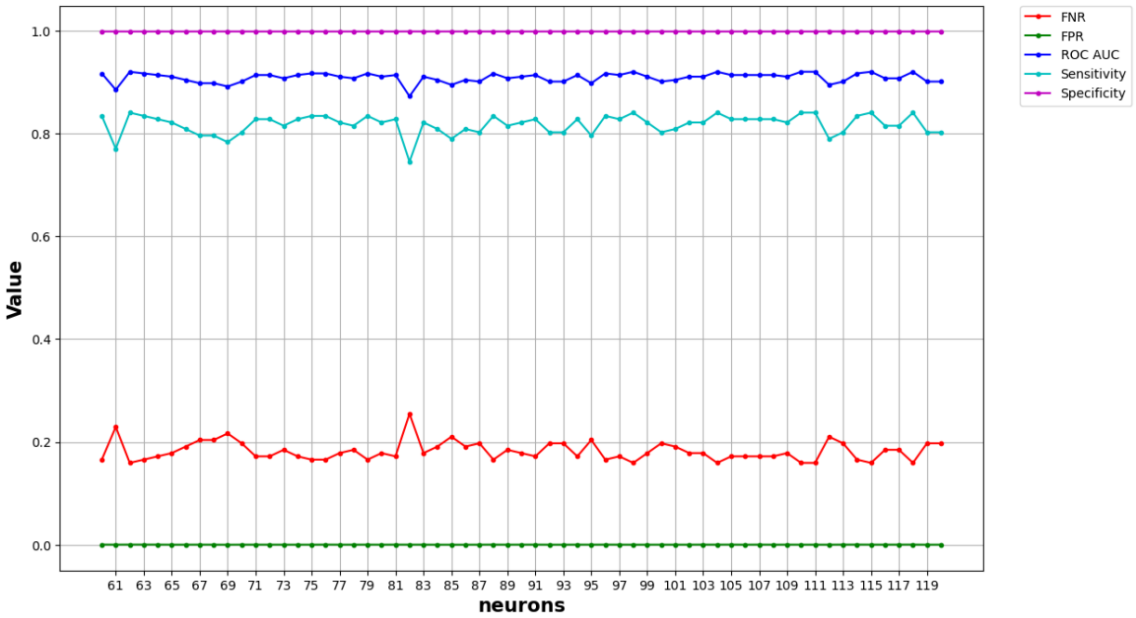
Rysunek 12: Wykres skuteczności Regresji Logistycznej w zależności od parametru C

3.9 MLP - Perceptron Wielowarstwowy

Jak wynika z diagramu korelacji zmiennych w danych praktycznie nie występują zależności liniowe, stąd przy doborze liczby neuronów należy brać pod uwagę ich większą liczbę niż 1. W pierwszej kolejności przeprowadzono test dla jakiej liczby neuronów w pierwszej warstwie skuteczność jest największa (Rysunek 13). Zachowując liczbę neuronów najlepszego wyniku dla pierwszej warstwy, w daleszej kolejności sprawdzono najlepsze ustawnienie drugiej wartwy (Rysunek 14). W ten sposób uzyskano prostą architekturę MLP o najlepszej skuteczności, gdzie pierwsza wartwa zawierała 56 nauronów a druga 62. Sprawdzono również różne ustawienia parametru *batch_size*, uzyskując wysoką skuteczność dla rozmiaru równego 50 (Tabela 7).



Rysunek 13: Wykres skuteczności MLP w zależności od liczby neuronów pierwszej wartwy



Rysunek 14: Wykres skuteczności MLP w zależności od liczby neuronów drugiej wartwy

FNR	FPR	ROC AUC	Sensitivity	Specificity
13.3758 %	0.0330 %	0.9330	86.6242 %	99.8552 %

Tabela 7: Wartości miar klasyfikatora MLP o najlepszej skuteczności

3.10 Zestawienie najlepszych wyników

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
MLP	13.3758 %	0.0330 %	0.9330	86.6242 %	99.8552 %
QDA	14.0127 %	0.4881 %	0.9275	85.9873 %	99.8556 %
Naive Bayess (Gaussian)	15.2866 %	0.4071 %	0.9215	84.7134 %	99.8579 %
Random Forest	15.9236 %	0.0043 %	0.9204	84.0764 %	99.8595 %
kNN	16.5605 %	0.0032 %	0.9172	83.4395 %	99.8606 %
Decision Tree	17.8344 %	0.0171 %	0.9107	82.1656 %	99.8627 %
LDA	18.4713 %	0.0160 %	0.9076	81.5287 %	99.8637 %
SVM	18.4713 %	0.0181 %	0.9076	81.5287 %	99.8637 %
Logistic Regression	28.6624 %	0.0181 %	0.8566	71.3376 %	99.8808 %

Tabela 8: Zestawienie najlepszych wyników klasyfikatorów

Jak można zauważyć z powyższej tabeli 8 największą skuteczność uzyskano dla klasyfikatora MLP. Dlatego w dalszym badaniu wpływu metod *feature selection*, metod dla problemu *class-imbalance*, *cost-sensitive learning* oraz *ensemble learning* będzie sprawdzany w oparciu o klasyfikator MLP, QDA oraz Random Forest (ponieważ uzyskał niższą stopę błędów FPR niż Naive Bayess).

4 Wybór zmiennych

Innym możliwością na poprawę skuteczności klasyfikatorów jest wybór jedynie najbardziej istotnych zmiennych lub też zmniejszenie wymiarowości problemu, w tym przypadku metodą PCA (*Principal Component Analysis*). Natomiat metodami jekie wybrano do wyboru zmiennych były: RFE (*Recursive Feature Elimination*) oraz *Forward feature selection* z wykorzystaniem Regresji Logistycznej.

4.1 RFE

Biorąc pod uwagę klasyfikator Random Forest, jako wyznacznik najbardziej istotnych zmiennych na podstawie Indeksu Gini, celem RFE jest wybór cech poprzez rekurencyjne rozpatrywanie coraz mniejszych zbiorów cech. Indeks Gini określa która cecha jest w stanie najlepiej odseparować rozpatrywane klasy (określenie "nieczystości" podziału). Poniższy fragment kodu (Rysunek 15) przeprowadza wybór zmiennych metodą RFE oraz wyświetla na wyjściu jakie zmienne zostały wybrane jako najbardziej istotne.

```
rfe = RFE(estimator=RandomForestClassifier(), n_features_to_select=5, step=3)

X_rfe = pd.DataFrame(rfe.fit_transform(X, y))
X_rfe.columns = rfe.get_feature_names_out()

X_train, X_test, y_train, y_test = train_test_split(X_rfe, y, random_state=seed, test_size=0.33)

print("train rows: {}, test rows: {}".format(X_train.shape[0], X_test.shape[0]))

X_rfe.columns
```

```
train rows: 190820, test rows: 93987

Index(['V10', 'V11', 'V12', 'V14', 'V17'], dtype='object')
```

Rysunek 15: Fragment kodu realizujący wybór cech metodą RFE

Jak przedstawia tabela 9 wyniki każdego z trzech wybranych do dalszych badań klasyfikatorów pogorszyły się względem tych uzyskanych bez wyboru cech metodą RFE. Ogólna miara poprawności klasyfikacji ROC AUC wskazuje, że selekcja najbardziej istotnych cech tą metodą nie ma korzystnego wpływu na skuteczność klasyfikacji.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	15.9236 %	0.0394 %	0.9202	84.0764 %	99.8595 %
Random Forest	17.1975 %	0.0053 %	0.9140	82.8025 %	99.8616 %
MLP	21.0191 %	0.0053 %	0.8949	78.9809 %	99.8680 %

Tabela 9: Porównanie skuteczności klasyfikatorów po przeprowadzeniu RFE

4.2 Forward feature selection z Regresją Logistyczną

Podobnie jak w przypadku poprzedniej metody oceniane są zmienne, które mają największe znaczenia dla klasyfikacji. O ich ważności świadczą współczynniki liniowej kombinacji cech. Kolejny fragment kodu (Rysunek 16), jak poprzednio wyświetla wybrane za pomocą tej metody cechy. Można zauważyć, że uzyskano takie same kolumny jak w poprzedniej metodzie,

zatem wyniki skuteczności klasyfikatorów nie uległy zmianie. Co za tym idzie nie nastąpiła żadna poprawa.

```
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.linear_model import LogisticRegression

flr = SequentialFeatureSelector(LogisticRegression(random_state=seed), direction='forward', n_features_to_select=5)
X_flr = pd.DataFrame(flr.fit_transform(X, y))

X_flr.columns = rfe.get_feature_names_out()

X_train_flr, X_test_flr, y_train_flr, y_test_flr = train_test_split(X_flr, y, random_state=seed, test_size=0.33)

print("train rows: {}, test rows: {}".format(X_train_flr.shape[0], X_test_flr.shape[0]))

X_flr.columns
```

```
train rows: 190820, test rows: 93987
```

```
Index(['V10', 'V11', 'V12', 'V14', 'V17'], dtype='object')
```

Rysunek 16: Fragment kodu realizujący wybór cech metodą forward multivariate z wykorzystaniem Regresji Logistycznej

4.3 PCA

Metoda PCA w odróżnieniu od poprzednich metod nie daje w rezultacie najbardziej istotnych zmiennych, a tworzy liniowe kombinacje skorelowanych cech. Jak przedstawiono we wstępie, w danych brak istotnych korelacji pomiędzy wymiarami, stąd nie uzyskano żadnej redukcji wymiarów. Próba uproszczenia wymiarów przyniosła częściową stratę informacji, a zatem uzyskane skuteczności klasyfikatorów są mniejsze niż dotychczas uzyskane, z wyjątkiem QDA, gdzie uzyskano taki sam wynik (Tabela 10).

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	14.0127 %	0.4881 %	0.9275	85.9873 %	99.8556 %
Random Forest	19.7452 %	0.0032 %	0.9013	80.2548 %	99.8659 %
MLP	21.0191 %	0.0139 %	0.8948	78.9809 %	99.8680 %

Tabela 10: Porównanie skuteczności klasyfikatorów po przeprowadzeniu PCA

4.4 Podsumowanie metod wyboru cech

Biorąc pod uwagę wszystkie wyniki uzyskane z badania wpływu metod wyboru najbardziej istotnych cech na suktecznośc klasyfikacji udało się obniżyć stopę błędów FNR w przypadku niektórych klasyfikatorów. Zmniejszenie liczby błędnych klasyfikacji jako pozytywne odbyło się kosztem błędnych klasyfikacji do klasy negatywnej. Z tych powodów odrzuca się wybór cech jako metody usprawniające działanie konkretnych klasyfikatorów dla tego problemu.

5 Metody dla problemów *class-imbalanced*

Jednym ze sposobów radzenia sobie z niezbalansowaniem klas danych uczących jest ich preprocessing, w wyniku którego uzyskiwane są równo liczne klasy. Osiągane jest to przez over-sampling: powielanie elementów klas mało licznych lub generowanie nowych na podstawie orginalnych. Możliwy jest również under-sampling, gdzie pomijane elementy klasy liczniejszej w procesie uczenia. W badaniu sprawdzono metody *Random over-sampling*, *Smote over-sampling*, *Random under-sampling* i *Cluster Centroids under-sampling*.

5.1 Random over-sampling

Metoda *Random over-sampling* polega na powielaniu orginalnie występujących elementów klasy mniejszościowej.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	14.0127 %	0.6096 %	0.9269	85.9873 %	99.8554 %
Random Forest	15.9236 %	0.0202 %	0.9203	84.0764 %	99.8595 %
MLP	16.5605 %	0.0554 %	0.9169	83.4395 %	99.8605 %

Tabela 11: Porównanie skuteczności klasyfikatorów po przeprowadzeniu Random over-samplingu

W przypadku klasyfikatora MLP można zauważyć wyraźny spadek skuteczności w porównaniu do najlepszego uzyskanego wówczas wyniku. Skuteczność Random Forest i QDA nieznacznie się zmniejszyła w wyniku podwyższenia stopy błędów FPR (Tabela 11). Ponad to dla QDA przekroczone zostało górne ograniczenie FPR.

5.2 Smote over-sampling

W odróżnieniu od poprzedniego over-samplingu nowo występujące elementy klasy mniej licznej generowane są poprzez łączenie punktów tej klasy odcinkami linii, a następnie umieszcza na tych liniach sztuczne punkty.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	12.7389 %	0.4892 %	0.9339	87.2611 %	99.8535 %
Random Forest	14.0127 %	0.0735 %	0.9296	85.9873 %	99.8562 %
MLP	14.6497 %	0.0533 %	0.9265	85.3503 %	99.8573 %

Tabela 12: Porównanie skuteczności klasyfikatorów po przeprowadzeniu Smote over-samplingu

Podobnie jak przy poprzedniej metodzie over-samplingu spadła ogólna skuteczność MLP. Udało się jednak osiągnąć poprawę dla Random Forest i QDA. Stopa błędów FPR w ich przypadku wzrosła nieznacznie, gdzie wzrosła również skuteczność w wykrywaniu klasy pozytywnej. W ten sposób dla QDA uzyskano ogólną skuteczność zmierzoną metryką **ROC AUC** równą **0.9339**, która jest najwyższą jaką uzyskano do tamtej pory i nie przekroczono granicy błędów dla FPR.

5.3 Random under-sampling

Random under-sampling bazuje na losowym pomijaniu elementów klasy bardziej licznej zbioru uczączego w trakcie dopasowywania klasyfikatora.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
MLP	7.0064 %	4.8247 %	0.9408	92.9936 %	99.8368 %
Random Forest	8.2803 %	2.9767 %	0.9437	91.7197 %	99.8421 %
QDA	14.0127 %	0.6661 %	0.9266	85.9873 %	99.8554 %

Tabela 13: Porównanie skuteczności klasyfikatorów po przeprowadzeniu Random under-samplingu

Na podstawie tabeli 13 można zauważyć, że ta metoda balansowania liczności klas ma duży wpływ na metody Random Forest i MLP. Skuteczność klasyfikacji (*Sensitivity*) osiągnęła w ich przypadku najwyższe do tego momentu wyniki. Jednocześnie wystąpił wzrost błędnych klasyfikacji do klasy pozytywnej. Dla każdego z badanych klasyfikatorów poziom błędów FPR przekroczył przyjęte kryterium. Jednak przyjęcie takiego kryterium jest umowne i należało by uwzględnić tak wytrenowane klasyfikatory przy wyborze najlepszego rozwiązania.

Na tym etapie uzyskano najlepszą do tamtej pory ogólną skuteczność wyrażoną przez **ROC AUC** osiągając dla tej miary wynik **0.9437** przez klasyfikator Random Forest. ***Sensitivity*** wyniosło **91.7197 %**. W przypadku MLP poziom FPR jest nie do przyjęcia.

5.4 Cluster Centroids under-sampling

Technika *Cluster Centroids under-smapling* ta dokonuje undersamplingu poprzez generowanie nowego zbioru na podstawie centroidów metodami klasteryzacji. Algorytm generuje nowy zbiór według centroidów klastra algorytmem *KMeans*.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
Random Forest	6.3694 %	28.7786 %	0.8243	93.6306 %	99.7805 %
MLP	7.6433 %	3.7461 %	0.9431	92.3567 %	99.8397 %
QDA	12.7389 %	0.2846 %	0.9349	87.2611 %	99.8538 %

Tabela 14: Porównanie skuteczności klasyfikatorów po przeprowadzeniu Cluster Centroids under-samplingu

Wyniki zebrane w tabeli 14 są najbardziej obiecujące pod względem czułości na klasę pozytywną. Można również zauważyć, że w przypadku Random Forest osiągną bardzo wysoki poziom FPR, który jest nie do przyjęcia. W kryterium błędu FNR mieści się jedynie QDA.

5.5 Podsumowanie metod dla problemów class-imbalanced

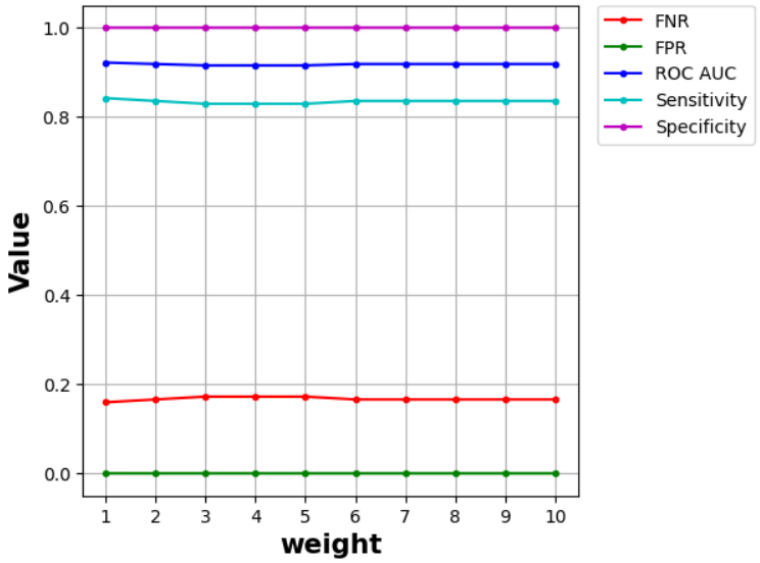
Najlepsze wyniki pod względem skuteczności klasyfikacji uzyskano wykorzystując metodu under-samplingu. W przypadku *Smote over-sampling* rónież uzyskano poprawę skuteczności, jednak nie aż tak dużą. Dla metody *Cluster Centroids under-sampling* pod względem skutecznego wykrywania fraudu z błędem FPR przekraczającym przyjętek kryterium, jednak nie w tak dużym stopniu (3.7461 %) najlepiej wypadł **MLP**. W jego przypadku **ROC AUC** wyniosło **0.9431** a **Sensitivity 92.3567 %**.

Jak już zauważono, przy zastosownaiu ostatniej z omawainych metod klasyfikator QDA znalazł się poniżej kryterium błędu FPR. Uwzględniając ten fakt uzyskał nową najlepszą **czułość** na klasę pozytywną **87.2611 %** i **ROC AUC = 0.9349**, jednocześnie spełniając założony warunek.

6 Cost-sensitive learning

Innym sposobem radzenia sobie z niezbalansowanymi klasami, a w szczególności w problemach niesymetrycznych kosztów błędów, który często pojawia się w medycynie (np. diagnoza chorób). Polega on na stosowaniu większej kary w procesie uczenia klasyfikatora, gdy popełni on błąd przy niepoprawnym sklasyfikowaniu elementu klasy o wysokiej istotności lub klasy mniejszościowej w przypadku problemów *class-imbalance*.

Zastosowanie niesymetrycznych kosztów błędów możliwe jest przy algorytmie Random Forest. Implementacja tego klasyfikatora zawiera parametr *class_weight*, który pozwala z różną wagą karać model podczas uczenia. Przetestowane zostały różne wartości tego parametru z zakresu od 1 do 10 kolejnych liczb naturalnych. W wyniku badań najlepszy rezultat (Tabela 15) uzyskano dla równych wag dla obu klas przyjmujących wartość 1. Uzyskany klasyfikator nie poprawił aktualnie najlepszego rezultatu.



Rysunek 17: Wykres skuteczności Random Forest w zależności od wagi klasy pozytywnej

FNR	FPR	ROC AUC	Sensitivity	Specificity
15.9236 %	0.0043 %	0.9204	84.0764 %	99.8595 %

Tabela 15: Wartości miar klasyfikatora Random Forest przy zastosowaniu niesymetrycznych kosztów błędów

7 Ensemble learning

Ensemble learning jest metodą uczenia maszynowego polegającą na wykorzystaniem kilku modeli klasyfikatorów dzięki której większą skuteczność niż mógłby uzyskać którykolwiek ze składowych klasyfikatorów samodzielnie. Rozwiązania jakie zostałe zbadane to *AdaBoost Classifier* i *Voting Classifier*.

7.1 AdaBoost Classifier

Klasyfikator AdaBoost jest metaestymatorem, który rozpoczyna się od dopasowania klasyfikatora na oryginalnym zbiorze danych, a następnie dopasowuje dodatkowe kopie klasyfikatora na tym samym zbiorze danych, ale w którym wagi niepoprawnie sklasyfikowanych instancji są dostosowane w taki sposób, że kolejne klasyfikatory skupiają się na trudniejszych przypadkach.

FNR	FPR	ROC AUC	Sensitivity	Specificity
22.9299 %	0.0192 %	0.8853	77.0701 %	99.8712 %

Tabela 16: Wartości miar klasyfikatora AdaBoost

Powyższa tabela 16 przedstawia wyniki mar skuteczności modelu kasyfikującego AdaBoost. Błąd FRN osiągnął bardzo słabą czułość na klasę pozytywną, dlatego nie będzie on dalej brany pod uwagę.

7.2 Voting Classifier

Ideą *VotingClassifier* jest połączenie koncepcyjnie różnych klasyfikatorów uczenia maszynowego i wykorzystanie większościowego głosowania lub średnich przewidywanych prawdopodobieństw (soft vote) do przewidywania etykiet klas. Taki klasyfikator może być przydatny dla zestawu równie dobrze działających modeli, aby zrównoważyć ich indywidualne słabości.

Modelami jakie wchodziły w skład *VotingClassifier* były 4 najskuteczniejsze klasyfikatory z tabeli 1: MLP, QDA, Naive Bayess oraz Random Forest. Fragment kodu który implementuje złożenie modelu klasyfikującego został przedstawiony poniżej (Rysunek 18).

```
mlp = MLPClassifier(solver='adam', hidden_layer_sizes=[56, 62,], max_iter=250, batch_size=50, random_state = seed)
qda = QuadraticDiscriminantAnalysis(reg_param=0.97, store_covariance=False)
nb = GaussianNB(var_smoothing=0.01)
rf = RandomForestClassifier(n_estimators=50, max_depth=14, random_state=seed)
```

```
eclf = VotingClassifier(estimators=[('mlp', mlp), ('qda', qda), ('nb', nb), ('rf', rf)], voting='soft', weights=[1, 1, 0, 1])
```

Rysunek 18: Fragment kodu realizujący model VotingClassifier

FNR	FPR	ROC AUC	Sensitivity	Specificity
13.3758 %	0.0213 %	0.9330	86.6242 %	99.8552 %

Tabela 17: Wartości miar klasyfikatora Voting Classifier

Czułość uzyskana przez VotingClassifier (Tabela 17) osiągnęłą taki sam poziom jak MLP po dostosowaniu parametrów bez wykorzystania metod balansowania klas (Tabela 1). Poza tym osiągną od niego niższy próg błędu FNR. Pomimo dobrej skuteczności uzyskanego klasyfikatora nie poprawia on uzyskanego do tamtej pory najlepszego rozwiązania.

8 Podsumowanie

Początkowe przetestowanie wymienionych na wstępie klasyfikatorów pozwoliło w praktyce zauważyć problem jaki stwarzają dane o wysoce niezrównoważonej liczności elementó klas. Wysoka od początku specyficzność, podczas gdy czułość na poziomie (w najlepszym przypadku, przy założonych kryteriach) ok. 83% dawała tym samym stopę błędów na relatywnie wysokom poziomie ~17% biorąc pod uwagę silną potrzebę poprawnego wykrywania fraudu. Obierając za cel poprawę skuteczności klasyfikacji podjęto badanie polegające na dostosowywaniu hiperparametrów modeli predykcyjnych, testowaniu metod rozwiązujących problem *class-imbalanced* (*over/under-sampling* i *cost-sensitive learning*), *Ensemble learningu* oraz metod wyboru zmiennych możliwe poprawiając zdolność do generalizacji.

Najlepsze rezultaty przy domyśnych ustawieniach uzyskały modele kNN, Random Forest oraz MLP, uwzględniając przyjęty próg błędu FPR wynoszący 0,5%. Najgorzej wypadły algorytmy Regresji Logistycznej i SVM.

Po dostrojeniu parametrów metodą eksperymentalną udało się redukować poziom błędów FPR poniżej ustalonego limitu dla wszystkich klasyfikatorów. Na tym etapie najlepsze skuteczności uzyskały MLP, QDA i Naive Bayess, poprawiając wcześniej osiągnięte poziomy czułości. W ramach metod wyboru zmiennych sprawdzono: RFE, metodę forward multivariate oraz PCA. Eksperymentalnie wykazano jednak, że pogarszają one jedynie skuteczność klasyfikacji określanej przez przyjęte miary.

Najbardziej efektywny wpływ na jakość klasyfikacji modeli wybranych do badań miały metody równoważenia liczności klas. Algorytmy over-sampling nie poczyniły tak dużej zmiany w skuteczności klasyfikowania w porównaniu do under-sampling. Algorytm Smote poprawił wcześniej uzyskany najlepszy wynik, jednak to metoda *Cluster Centroids under-sampling* ozkazała się być najlepszym sposobem do osiągnięcia najwyższej skuteczności. Uwzględniając próg błędu FNR z wykorzystaniem tego algorytmu najlepszym modelem jest QDA dla *reg_param* = 0.97. Luzując postawione warunki odnośnie FNR modelem, który miałby jeszcze większą czułość na klasę pozytywną, przy FNR 7.6433 % jest MLP z dwoma warstwami ukrytymi o rozmiarze 56 i 62 neuronów, uczony przy *batch_size* = 50 i 250 epokach. Wyniki pomiaru skuteczności tych modeli przedstawia tabela 18.

Classifier	FNR	FPR	ROC AUC	Sensitivity	Specificity
QDA	12.7389 %	0.2846 %	0.9349	87.2611 %	99.8538 %
MLP	7.6433 %	3.7461 %	0.9431	92.3567 %	99.8397 %

Tabela 18: Zestawienie klasyfikatorów o najlepszej skuteczności

Czułość bliską tej uzyskanej przez ww. QDA osiągnięto stosując klasyfikator *Voting Classifier* złożony z 4 pierwszych i najlepszych modeli z tabeli 1. Stosując metodę *Cost-sensitive learning* nie uzyskano bardziej korzystnych efektów.