Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Zarządzania

Sieci neuronowe i uczenie maszynowe

Piotr Otręba, Aleksander Jasiński, Jakub Kaźmierczyk, Kacper Łapot, Rafał Łubkowski

Kraków, maj 2025

Spis treści

Wprowadzenie

Na przestrzeni ostatnich dziesięcioleci obserwuje się wzrost znaczenia sektora lotnictwa pasażerskiego dla światowej turystyki. Liczba pasażerów wybierających samolot jako szybki i wygodny środek transportu stale rośnie. Problemem dla decydujących się na podróż lotniczą pozostaje jednak cena biletu lotniczego. Ustalając cenę na dany lot, linie lotnicze wykorzystują zaawansowane algorytmy, dzięki którym cena obliczana jest w sposób dynamiczny. Oznacza to, że bilety na ten sam lot zakupione w bardzo krótkim odstępie czasu mogą różnić się od siebie ceną. Zmienna natura cen połączeń lotniczych i mnogość czynników, które mogą na nie wpływać, utrudnia pasażerom znalezienie optymalnej oferty.

Opis zbioru danych

Przedmiotem analizy są dane dotyczące cen biletów lotniczych. Zostały pozyskane (metodą web scrapingu, za pomocą programu napisanego w Pythonie) ze strony momondo.pl. Pobieranie danych rozpoczęto 01.11.2024 i wyszukiwano informacje o lotach z Warszawy i Krakowa do czterech europejskich stolic: Berlina, Paryża, Londynu i Rzymu. Każdego dnia pobierane były dane o wylotach z okresu 31 dni następujących po danym dniu.

Analizie poddane zostały loty bezpośrednie i z jedną przesiadką, obsługiwane przez zarówno tanie linie lotnicze: Ryanair, Wizz Air, EasyJet, jak i przez te droższe: LOT, Lufthansa, British Airways, KLM, Air France.

Opis zmiennych w zbiorze danych

Zmienna	Opis
Extraction Weekday	dzień tygodnia, gdy pobrano daną obserwację
Flight Weekday	dzień tygodnia lotu
Departure time	godzina odlotu
Arrival time	godzina przylotu
dep city	miasto wylotu
arr city	miasto przylotu
Price	cena biletu (w zł)
Cabin bag	liczba sztuk bagażu podręcznego
Checked bag	liczba sztuk bagażu rejestrowanego
Days to departure	liczba dni pomiędzy dniem pobrania obserwacji a dniem wylotu
layover duration	czas przesiadki (h)
Airline1	linia lotnicza wykonująca pierwszy lot
Airline2	linia lotnicza wykonująca drugi lot (jeśli lot był bezpośredni, to występuje wartość ’-’)

W celu przygotowania danych do analizy konieczne było wykonanie szeregu dodatkowych czynności. Obejmowały one m.in. usunięcie wartości odstających (szczególnie pod względem nieracjonalnie długiego czasu przesiadki) i nielicznych braków (dosłownie kilka obserwacji), ujednolicenie wartości w kolumnach i ich zamianę na odpowiedni format. Następnie dla zmiennych*: Extraction Weekday, Flight Weekday, dep city, arr city, Airline1* i *Airline2* zastosowano onehot encoding w celu zamiany zmiennych nominalnych na liczbowe.

Przegląd literatury

Pierwszym rozważanym w projekcie problemem jest predykcja ceny biletu lotniczego. To zagadnienie stało się przedmiotem licznych publikacji naukowych. W [1] prognozowano ceny biletów linii Aegan Airlines na trasie Saloniki-Stuttgart. Zastosowane modele trenowano z zastosowaniem 10-krotnej walidacji krzyżowej i sprawdzano jakość predykcji dla pełnego zbioru danych, jak i jego podzbiorów z usuniętymi poszczególnymi zmiennymi. Najlepszym modelem okazał się baggingowy las losowy, którego MAPE dla całego zbioru wyniosło 87,42%. MAPE zwykłego lasu losowego było niższe o zaledwie 3 pkt. proc., przy znacznie krótszym czasie trenowania modelu. Perceptronowa sieć neuronowa trenowana na całym zbiorze osiągnęła MAPE na poziomie 80,28%, a jej trenowanie zajęło najwięcej czasu spośród rozważanych modeli. [2] stanowi kontynuację i rozszerzenie badań z publikacji [1]. Problem predykcji ceny rozważano osobno dla każdej z 24 kombinacji (linia lotnicza, destynacja). Po raz pierwszy do problemu predykcji ceny biletu lotniczego zastosowano modele QML i QDL, czyli kwantowego uczenia maszynowego i kwantowego uczenia głębokiego. Kwantowa perceptronowa sieć neuronowa jako najlepszy model osiągnęła średnie R^2=91%. Średnie R^2 tradycyjnej sieci perceptronowej było od niego niższe o 2 p.p. przy jednoczesnym znacznie krótszym czasie trenowania. W [3] do predykcji cen biletów na indyjskie loty krajowe użyto m.in. konwolucyjnych sieci neuronowych – w tym nowatorskiej bayeskowskiej konwolucyjnej sieci neuronowej. Wyniki porównano z tymi uzyskanymi za pomocą modeli uczenia maszynowego: lasu losowego, wzmocnienia gradientowego (XGBoost) i drzewa decyzyjnego. Badania pokazały potencjał metod bayesowskich w omawianym problemie. Autorzy [4] zauważyli, że istotną wadą licznych opisanych powyżej metod jest ich ograniczona skuteczność w wykrywaniu zależności w czasie. W związku z tym do predykcji zastosowali rekurencyjne sieci neuronowe, które nadają się do prognozowania szeregów czasowych.

Sieci neuronowe

Regresja

Metodologia

W ramach projektu testowano jakość predykcji perceptronowej sieci neuronowej w zależności od licznych parametrów. Dla każdej kombinacji parametrów proces trenowania był powtarzany 5-krotnie (w niektórych przypadkach 10-krotnie), aby zmniejszyć losowość uzyskiwanych wyników. Istotne było też zapobieżenie przeuczeniu sieci, dlatego

zaimplementowany został mechanizm zatrzymania trenowania modelu w przypadku braku poprawy walidacyjnego MSE przez kolejne 50 epok. Jakość predykcji mierzona była za

pomocą różnych metryk: MSE, MAPE oraz R^2. Pierwotną siecią, której parametry zmieniano, była sieć z 37 neuronami w warstwie wejściowej (ta liczba jest równa liczbie zmiennych w zbiorze danych), 16 neuronami w pojedynczej warstwie ukrytej i jednym neuronem w warstwie wyjściowej. Funkcją aktywacji była funkcja ReLu, a szybkość uczenia (learning rate) ustalono na 0,1. Liczba epok była zmienna i determinowana przez wspomniany mechanizm wczesnego zatrzymania. Wyniki uzyskane dla bazowej sieci ilustruje poniższa tabela.

train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
248871,6	238467,9	237090,9	326,76	0,48	0,78

Testowanie parametrów

W ramach projektu testowano wpływ wielu parametrów na dokładność predykcji. Część z nich (liczba neuronów, liczba warstw ukrytych czy funkcja aktywacji) była związana z architekturą sieci, pozostałe (dobór zmiennych, proporcje podziału zbioru na zbiór treningowy, testowy i walidacyjny czy oversampling) dotyczyły natomiast danych, na których sieć była trenowana. Ważne: przy badaniu architektury sieci w kolumnie num layers są zapisane liczby w nawiasach kwadratowych. Każda liczba odpowiada pojedynczej warstwie ukrytej, a jej wartość odpowiada liczbie neuronów w tej warstwie ukrytej.

Liczba neuronów

a) Liczba neuronów w pierwszej warstwie ukrytej

Początkowo sprawdziliśmy na 5 powtórzeniach liczbę neuronów w 1 warstwie ukrytej, najmniejsze błędy mogliśmy zaobserwować dla liczby neuronów z przedziału 10-20. Następnie wykonaliśmy na 10 powtórzeniach test liczby neuronów z tego przedziału.

num_layers	learning_rate	Activation function	train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
[10]	0.1	relu	249814.1	239454.9	238716.8	319.99	0.56	0.78
[11]	0.1	relu	232347.1	225090.9	224020.9	313.89	0.59	0.79
[12]	0.1	relu	244188.6	237412	236164.7	323.66	0.53	0.78
[13]	0.1	relu	228744.6	223534.9	222117.4	320.18	0.52	0.79
[14]	0.1	relu	238696.7	232599.1	231032.2	323.26	0.48	0.78
[15]	0.1	relu	245806.4	237689.4	236461.8	325.15	0.55	0.78
[16]	***0.1***	***relu***	***248871.6***	***238467.9***	***237090.9***	***326.76***	***0.48***	***0.78***
[17]	0.1	relu	253760.2	241810.8	240680.9	328.56	0.55	0.77
[18]	0.1	relu	244954.9	237914	236463.1	326.34	0.56	0.78
[19]	0.1	relu	254306.3	246714.9	245158.5	335.06	0.52	0.77
[20]	0.1	relu	257848.2	246993.8	245507.5	335.58	0.54	0.77

Najniższe błędy zaobserwowaliśmy dla 16 neuronów, a więc później testowaliśmy dla 16 neuronów w pierwszej warstwie i różnej liczbie neuronów w kolejnych warstwach.

b) Liczba neuronów w drugiej warstwie ukrytej:

num_layers	train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
[16, 1]	931251,43	892600,31	891429,90	540,22	0,57	0,15
[16, 2]	630902,01	583006,34	580975,58	440,02	0,65	0,45
[16, 3]	328052,14	293587,36	289762,90	353,00	0,67	0,72
[16, 4]	340485,32	309831,11	305449,39	360,85	0,59	0,71
[16, 5]	332781,24	289660,69	285785,89	357,13	0,61	0,73
[16, 6]	380265,51	328542,97	323278,06	378,27	0,60	0,69
[16, 7]	484110,15	432384,77	418981,23	438,39	0,64	0,60
[16, 8]	449095,80	390139,07	384750,12	394,39	0,59	0,63
[16, 9]	453577,66	376580,73	371127,52	388,78	0,54	0,65
[16, 10]	460413,44	400559,90	395038,17	395,04	0,54	0,62
[16, 11]	551366,75	457561,87	451757,99	413,85	0,53	0,57
[16, 12]	589751,04	468713,36	461087,00	415,15	0,52	0,56
[16, 13]	575925,37	516293,77	510295,50	426,47	0,56	0,51
[16, 14]	609723,94	486167,53	468893,20	415,88	0,52	0,55
[16, 15]	571621,60	483212,58	477282,78	420,93	0,54	0,54
[16, 16]	587978,94	457820,82	438291,28	410,14	0,54	0,58
[16, 17]	623807,98	520605,91	487057,26	419,10	0,54	0,53
[16, 18]	720479,28	560422,21	554358,35	440,13	0,53	0,47
[16, 19]	763660,74	540162,33	531274,52	433,68	0,54	0,49
[16, 20]	785431,76	592263,76	586144,01	450,02	0,56	0,44
[16, 21]	734350,32	610417,29	604856,25	444,68	0,55	0,42
[16, 22]	762047,76	599505,60	593605,15	447,58	0,55	0,43
[16, 23]	803060,01	596445,00	590764,48	444,05	0,53	0,44
[16, 24]	748887,70	607714,98	601711,48	451,66	0,58	0,43
[16, 25]	698484,05	585500,18	579795,93	441,01	0,58	0,45
[16, 26]	760194,28	617811,56	612030,63	451,81	0,57	0,42
[16, 27]	823327,83	629567,52	623763,86	454,47	0,57	0,40
[16, 28]	814294,25	602046,18	595910,30	452,25	0,57	0,43
[16, 29]	800939,11	625328,07	619467,62	453,65	0,57	0,41
[16, 30]	763009,45	628511,28	623174,58	447,31	0,57	0,40

Otrzymane wyniki są gorsze od metryk dla sieci z pojedynczą warstwą ukrytą, która osiąga $R^2 \approx 0{,}78$ i MSE≈237 000. Wśród układów dwuwarstwowych najlepszy rezultat uzyskano dla zestawu, w którym w pierwszej warstwie ukrytej znajduje się 16 neuronów, a w drugiej 5, czyli [16, 5], ($R^2 \approx 0{,}73$, MSE≈285 800), lecz wciąż jest to mniej niż pojedyncza warstwa.

Większa liczba neuronów w drugiej warstwie (powyżej 5) prowadzi do rosnącego błędu i spadku R^2, co sugeruje przeuczenie i gorszą generalizację.

Dobór zmiennych w zbiorze testowym

W celu lepszego zrozumienia, które zmienne wejściowe mają największy wpływ na skuteczność modelu sieci neuronowej, przeprowadzono serię eksperymentów polegających na stopniowym usuwaniu pojedynczych kolumn z danych uczących. Po każdym takim zabiegu model był ponownie trenowany, a jego jakość oceniano na zbiorze testowym przy pomocy kilku metryk regresyjnych (MSE, MAE, $MAPE$ oraz R^2).

Podejście to pozwala określić, które cechy zawierają najwięcej istotnych informacji pomocnych w przewidywaniu ceny biletu lotniczego. Ponadto, daje to również wskazówkę, które zmienne mogą być potencjalnie redundantne lub mocno skorelowane z innymi – ich brak nie wpływa bowiem znacząco na jakość prognozy.

Usunięta zmienna	train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
Departure_time	295481,52	271277,34	263813,64	326,32	0,34	0,76
Arrival_time	285163,08	267870,78	260479,67	321,88	0,33	0,76
Flight_time	305516,30	273747,28	266199,34	330,03	0,35	0,75
Num_Layovers	312154,52	280764,72	272346,26	334,44	0,36	0,75
Ticket_class	377664,33	354763,02	348041,87	363,67	0,38	0,68
Cabin_bag	327789,03	300402,96	292249,05	356,58	0,41	0,73
Checked_bag	284649,45	259977,7	255909,88	328,16	0,35	0,76
Days_to_departure	325882,04	297804,77	290556,6	358,54	0,4	0,73
layover_duration	299831,41	273868,69	266324,15	331,42	0,35	0,75

Interpretacja wyników:

- *Ticket_class* zawiera dużo informacji istotnych dla przewidywania ceny biletu, ponieważ jej brak powoduje największy spadek jakości predykcji.
- *Cabin_bag, Days_to_departure* oraz *Num_Layovers* wskazują, że ich brak ogranicza zdolność sieci do przewidywania zmienności ceny.
- *Checked_bag, Arrival_time* oraz *Departure_time* to mniej istotne zmienne – współczynnik R^2 wciąż pozostaje na poziomie powyżej 0,75.

Warto zauważyć, że *Flight_time, Arrival_time* oraz *Departure_time* dają **bardzo podobne wyniki**. Oznacza to, że zmienne te mogą być ze sobą mocno skorelowane, co znajduje również uzasadnienie logiczne.

Funkcja aktywacji

Przeprowadzono testy czterech funkcji aktywacji: ReLU, tanh, sigmoidalnej oraz liniowej, przy zachowaniu niezmienności pozostałych parametrów.

Porównanie wyników różnych funkcji aktywacji

Funkcja Aktywacji	train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
relu	233911,02	225621,60	224281,63	317,14	0,55	0,79
tanh	957135,04	921910,13	938612,49	458,95	0,55	0,13
sigmoid	785337,95	834841,12	779326,15	412,53	0,46	0,28
linear	358170,05	318765,97	322210,64	393,33	1,76	0,70

Na podstawie przeprowadzonych eksperymentów można wyciągnąć następujące wnioski:

- **Funkcja ReLU** okazała się najbardziej efektywna spośród testowanych wariantów, osiągając:
  - Najniższe wartości błędów MSE: 233911.02 (trening), 225621.6 (walidacja) i 224281.63 (test)
  - Najwyższą wartość współczynnika determinacji $R^2 = 0.792866692$
  - Stosunkowo niskie wartości błędów absolutnych (MAE = 317.1384937, MAPE = 0.56)
- **Funkcja liniowa** zajęła drugie miejsce pod względem skuteczności, co sugeruje częściowo liniowy charakter analizowanych danych. Wyniki:
  - MSE na poziomie 358170.05 (trening) i 322210.64 (test)
  - Wartość $R^2 = 0.70$
  - Wysoki błąd procentowy MAPE = 1.76, wskazujący na problemy z przewidywaniem wartości o małej skali
- **Funkcje sigmoidalna i tanh** wykazały znacząco gorsze wyniki:
  - Dla sigmoid: MSE testowe = 779326.1474, $R^2 = 0.28$
  - Dla tanh: MSE testowe = 938612.49, $R^2 = 0.13$

ReLU jest zdecydowanie najlepszym wyborem funkcji aktywacji dla modelu.

Proporcje podziału danych

Właściwy podział danych na zbiory treningowy, walidacyjny i testowy to podstawa rzetelnej oceny modelu i zapobiegania przeuczeniu.

Metryki jakości predykcji dla różnych proporcji podziału danych

Proporcje podziału (treningowy:walidacyjny:testowy	Test_MSE	Test_MAE	Test_MAPE (%)	Test_R2	Train_MSE	Val_MSE
0,6: 0,2 : 0.2	1179222,63	661,76	85,67	0,56	481640457,83	1869550180,42
0,7 : 0.15 : 0.15	1119589,72	623,67	79,50	0,68	7601748,03	2632456,47
0,75 : 0.125 : 0.125	1137287,17	665,78	90,87	0,55	1057950317,05	3823501127,38
0,8 : 0,1 : 0,1	1138444,75	636,85	78,77	0,49	58989622,73	5551696023,67

- **Najlepsze Wyniki dla Podziału 0.7 : 0.15 : 0.15:** Model osiągnął najbardziej optymalne wyniki dla podziału, gdzie zbiór treningowy stanowi 70%, walidacyjny 15%, a testowy 15%.
    - **Najniższe Błędy Testowe:** Odnotowano najniższe wartości błędów na zbiorze testowym:
        - **Test_MSE:** 1,119,589.72
        - **Test_MAE:** 623.67
        - **Test_MAPE:** 79.50%
    - **Najwyższy Współczynnik Determinacji:** Współczynnik R2 wyniósł 0.68, co wskazuje na najlepsze dopasowanie modelu do danych testowych w porównaniu do innych podziałów.
    - **Niskie Błędy Treningowe i Walidacyjne:** Wartości **Train_MSE** (7,601,748.03) i **Val_MSE** (2,632,456.47) dla tego podziału są znacznie niższe niż w przypadku innych konfiguracji, co sugeruje, że model dobrze nauczył się wzorców z danych treningowych i jednocześnie generalizuje na danych walidacyjnych.
- **Wpływ Zwiększenia Udziału Zbioru Treningowego:**
    - Zwiększenie udziału zbioru treningowego do 75% i 80% (przy jednoczesnym zmniejszeniu zbiorów walidacyjnego i testowego) skutkuje pogorszeniem wyników na zbiorze testowym. Wzrasta **Test_MAE** i **Test_MAPE**, a **Test_R2** spada, co może wskazywać na przetrenowanie (overfitting) modelu.
    - Ekstremalnie wysokie wartości **Train_MSE** dla podziałów 0.75:0.125:0.125 (1,057,950,317.05) i 0.6:0.2:0.2 (481,640,457.83) oraz również wysokie **Val_MSE** sugerują znaczące problemy z dopasowaniem modelu lub z samymi danymi w tych konfiguracjach.
- **Wpływ Zwiększenia Udziału Zbioru Walidacyjnego:**

- Zwiększenie udziału zbioru walidacyjnego do 20% (podział 0.6:0.2:0.2) również skutkuje gorszymi wynikami niż w przypadku 15% (podział 0.7:0.15:0.15). Wartości błędów na zbiorze testowym są wyższe, a R2 niższe.

Oversampling na zbiorze treningowym

Wartą uwagi cechą zbioru danych, którym dysponujemy, jest jego duże niezbilansowanie. Przejawia się ono szczególnie w stosunku liczby lotów w klasie ekonomicznej do liczby lotów w klasie biznes (tych drugich jest kilkadziesiąt razy mniej). W związku z tym, osobno dla każdej zmiennej kategorycznej dokonano bilansowania według następującej procedury.

Najpierw podzielono liczbę obserwacji w zbiorze treningowym przez liczbę K unikatowych kategorii danej zmiennej - ten iloraz oznaczymy jako L. Następnie pogrupowano zbiór treningowy na K podzbiorów według wartości rozważanej zmiennej. Z każdego z nich losowano ze zwracaniem L obserwacji. Łącząc je, otrzymano nowy zbiór treningowy o liczności równej liczności pierwotnego zbioru, w którym nie występował już problem niezbilansowania klasy. Na tak przekształconym zbiorze treningowym trenowano sieć neuronową. Wyniki ilustruje poniższa tabela.

treningowe MSE	walidacyjne MSE	testowe MSE	testowe MAE	testowe MAPE	testowe R^2	Zmienna
245358,40	289382,90	273814,80	351,43	0,62	0,74	Airline1
218756,70	246339,10	243515,00	330,04	0,50	0,77	Cabin_bag
468256,00	224229,40	291315,30	332,83	0,58	0,79	Checked_bag
241346,80	233960,00	234106,20	328,52	0,55	0,79	Extraction_Weekday
251863,50	235959,20	238142,90	324,82	0,54	0,78	Flight_weekday
243780,20	205999,90	284738,30	350,73	0,56	0,73	Num_Layovers
1176076,00	850393,50	586475,00	449,03	0,58	0,49	Ticket_class

Z powyższej tabeli _wynika, że opisana procedura oversamplingu nie przynosi poprawy jakości predykcji. Wartości poszczególnych metryk nie są lepsze niż dla sieci uczonej na nieprzekształconym w żaden sposób zbiorze danych. W szczególności, znaczne pogorszenie zostało zaobserwowane dla oversamplingu wykonanego ze względu na klasę lotu. Testowe

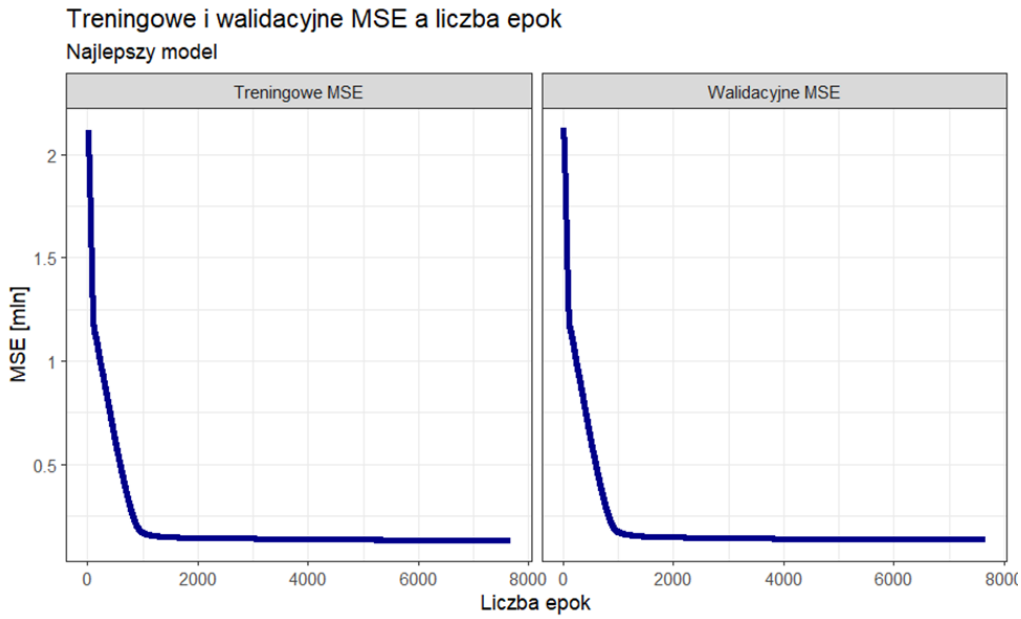
MSE wynoszące ponad 580 tys. i testowe R^2 na poziomie 0,49 sugerują, że taki model radzi sobie znacząco gorzej od bazowej sieci.

Wybór najlepszej sieci

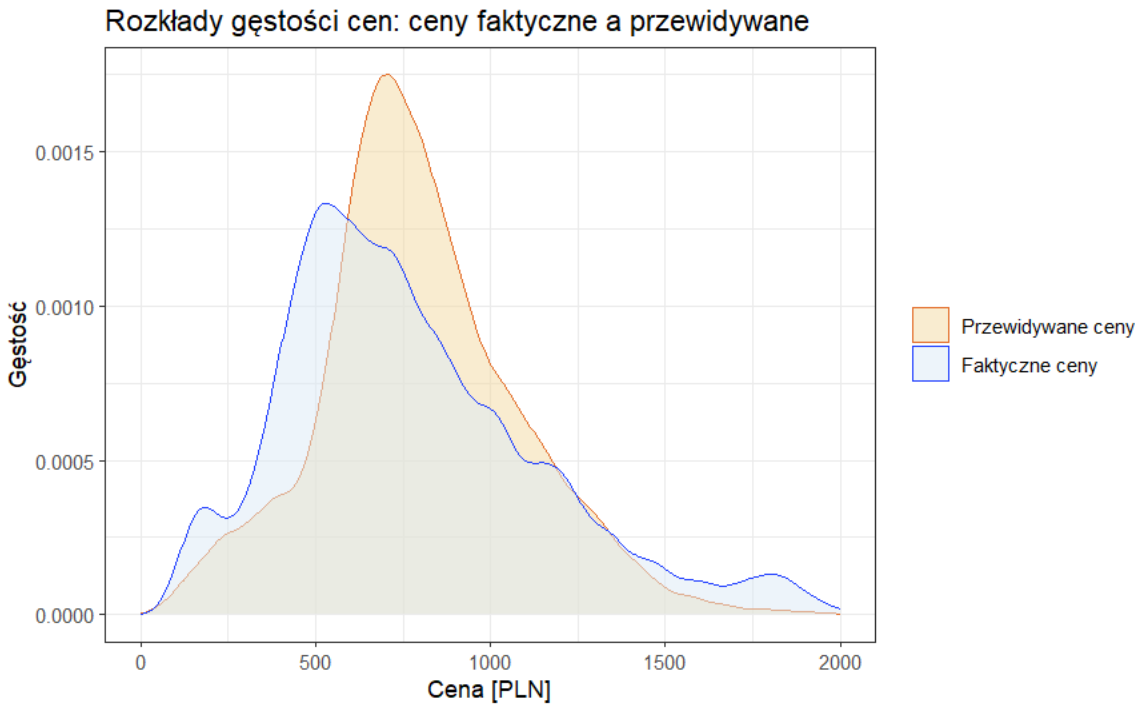
Dla sprawdzonych kombinacji parametrów najlepszą jakość predykcji uzyskała sieć o jednej warstwie ukrytej z 16 neuronami, z funkcją aktywacji ReLu. Proces trenowania tej sieci odbywał się z parametrem learning rate równym 0,01, tak, by jak najprecyzyjniej trafić w minimum funkcji straty. Trenowanie powtórzono 10-krotnie w celu uzyskania stabilnych wyników. Wyniki jakości predykcji przedstawia poniższa tabela.

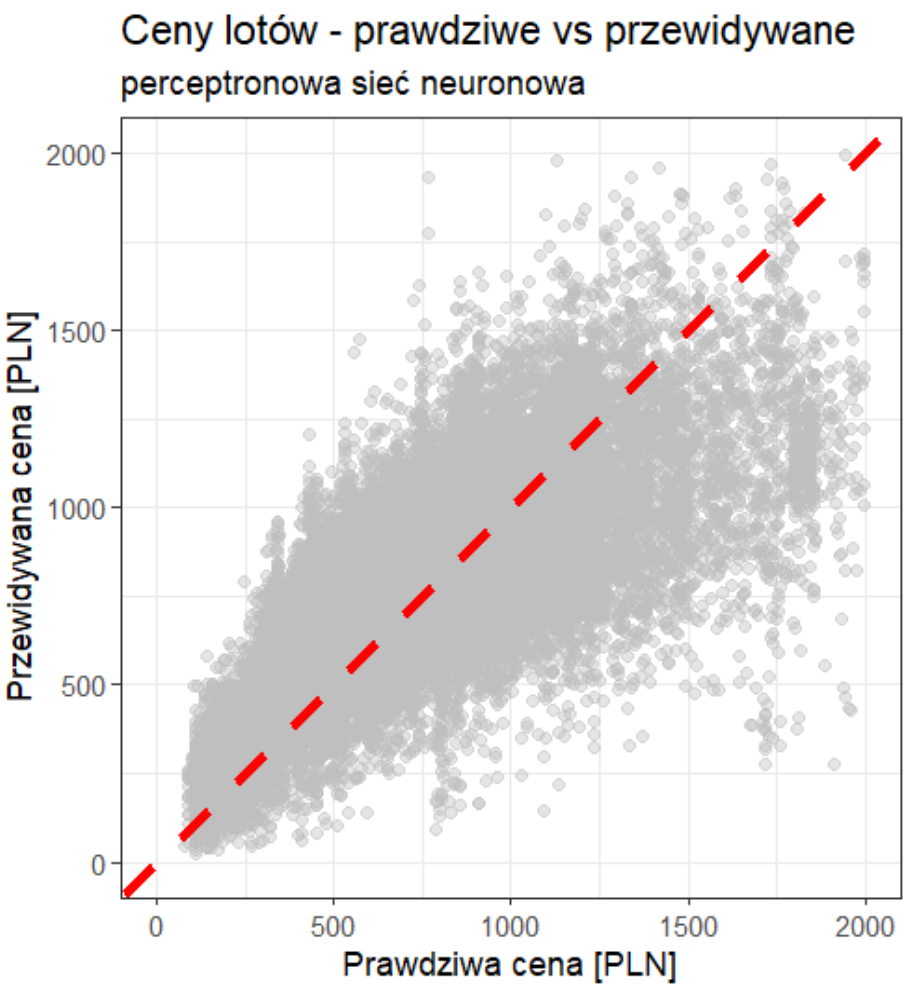
train_MSE	val_MSE	test_MSE	test_MAE	test_MAPE	test_R2
137784	134419,3	131173,1	233,84	0,26	0,87

Zwizualizujemy ponadto zależność treningowego i walidacyjnego MSE od liczby epok.



Początkowo wraz ze wzrostem liczby epok następuje gwałtowny spadek treningowego i walidacyjnego MSE. Funkcja straty, która dla pierwszych epok przyjmowała wartości > 2mln, dla 1000 epok przyjmuje już wartość około 150 tys. Dalsze zwiększanie liczby epok nie prowadzi co prawda do przeuczenia (walidacyjne MSE pozostaje niskie), ale skutkuje bardzo niewielkim zmniejszaniem MSE. Trenowanie zostało przerwane przed 8000 epoką w związku ze wzrostem MSE (mechanizm wczesnego zatrzymania). Poniższe dwa wykresy prezentują przewidywane ceny lotów na tle rzeczywistych wartości.


Rozkłady gęstości cen: ceny faktyczne a przewidywane
Gęstość
0.0015
0.0010
0.0005
0.0000
0
500
1000
1500
2000
Cena [PLN]
Przewidywane ceny
Faktyczne ceny


Ceny lotów - prawdziwe vs przewidywane
perceptronowa sieć neuronowa
Przewidywana cena [PLN]
2000
1500
1000
500
0
0
500
1000
1500
2000
Prawdziwa cena [PLN]

Klasyfikacja: Klasyfikacja linii lotniczej (czy tania linia lotnicza?)

Metodologia

W drugiej części projektu celem będzie zaklasyfikowanie linii lotniczej do grupy low-cost (tanich linii lotniczych) lub do grupy pozostałych przewoźników (tzw. tradycyjnych linii lotniczych). Do tych pierwszych zaliczyć możemy linie: Ryanair, Wizz Air, easyJet. Do drugiej grupy należą natomiast: LOT, Lufthansa, British Airways, Air France, KLM. W zbiorze danych utworzono zmienną **Is_low-cost**, która przyjmuje wartość 1, jeśli dana linia lotnicza jest w grupie tanich przewoźników lub 0 w przeciwnym wypadku. To właśnie ta zmienna jest zmienną objaśnianą. Zbiór danych został odpowiednio przefiltrowany – zawiera tylko loty bezposrednie i takie loty z przesiadką, że pierwszy i drugi lot był wykonany albo przez linię low-cost, albo przez linię tradycyjną. W ten sposób uniknięto konieczności tworzenia dwóch zmiennych określających rodzaj linii wykonującej pierwszy i drugi lot. Ponadto, konieczne było usunięcie zmiennych: Airline1 i Airline2, ponieważ w przeciwnym wypadku predykcja binarna nie miałaby sensu (nazwa linii determinuje jednoznacznie, czy linia należy do grupy low-cost). Wreszcie, konieczne było również usunięcie zmiennej ticket_class: tanie linie lotnicze nie oferują lotów w klasie biznes, zatem taka obserwacja byłaby jednoznacznie zaklasyfikowana do grupy tradycyjnych przewoźników. Głównym predyktorem rodzaju linii lotniczej powinna być w tym przypadku cena biletu - droższy bilet zwiększa prawdopodobieństwo, że oferta nie pochodzi od taniej linii lotniczej. Niemniej, z uwagi na fakt, że zbiór zawiera wiele droższych lotów wykonywanych również przez tanie linie lotnicze, zmienna Price nie jest idealnie zależna od linii lotniczej, co uzasadnia jej uwzględnienie w macierzy cech.

Testowanie Parametrów

Parametry, które podlegały testowaniu, były analogiczne do tych rozważanych w problemie predykcji ceny biletu z wykorzystaniem sieci neuronowej. Są to zatem: liczba neuronów i warstw ukrytych, dobór zmiennych, funkcja aktywacji, proporcja podziału danych na zbiory: treningowy, walidacyjny i testowy oraz oversampling na zbiorze treningowym. Jakość predykcji mierzono za pomocą współczynnika determinacji, recall i precision. Recall określa odsetek obserwacji należących do klasy pozytywnej (czyli linii low-cost) poprawnie zaklasyfikowanych do tej klasy. Precision opisuje odsetek obserwacji zaklasyfikowanych do klasy pozytywnej, które faktycznie do tej klasy należały). Nie jest możliwa jednoczesna minimalizacja obu metryk, dlatego wprowadzono miarę, która uwzględnia je obie: f1 score będący średnią harmoniczną recall i precision. Dodatkowo, jakość klasyfikacji mierzymy też za pomocą accuracy, czyli odsetka obserwacji poprawnie zaklasyfikowanych (do obu klas).

Liczba neuronów

W tym etapie testowaliśmy, który zestaw liczb warstw ukrytych i liczb neuronów w pojedynczej warstwie będzie najlepszy. Testowaliśmy liczbę warstw (od 1 do 5) oraz liczbę neuronów w każdej z warstw (od 1 do 20). Oto wyniki:

num_layers	train_accuracy	val_accuracy	test_accuracy	test_precision	test_recall	test_f1
[2, 2, 2]	***0,6436***	***0,7444***	***0,7938***	***0,6413***	***0,7938***	***0,7063***
[11]	0,3829	0,6584	0,7821	0,6794	0,7821	0,7037
[1]	0,4584	0,7526	0,7673	0,7146	0,7673	0,6845
[12]	0,4770	0,5489	0,7695	0,6718	0,7695	0,6801
[2]	0,6436	0,7577	0,7668	0,6160	0,7668	0,6761
[3, 3, 3]	0,5392	0,7444	0,7440	0,6617	0,7440	0,6716
[3]	0,6201	0,6468	0,7417	0,6451	0,7417	0,6562
[1, 1, 1, 1]	0,7393	0,7374	0,7378	0,6154	0,7378	0,6538
[4, 4, 4, 4]	0,4521	0,7444	0,7228	0,6494	0,7228	0,6524
[7]	0,2572	0,7530	0,7522	0,6093	0,7522	0,6520
[11, 11, 11, 11, 11]	0,2607	0,7444	0,7508	0,6084	0,7508	0,6495
[17, 17]	0,2607	0,7444	0,7501	0,6036	0,7501	0,6489
[14, 14]	0,2607	0,7444	0,7496	0,6071	0,7496	0,6471
[6, 6]	0,3564	0,7471	0,7493	0,6584	0,7493	0,6469
[15]	0,3564	0,6467	0,7257	0,5944	0,7257	0,6427
[9, 9, 9, 9, 9]	0,2607	0,7444	0,7473	0,6064	0,7473	0,6424
[15, 15, 15, 15, 15]	0,3564	0,6467	0,7472	0,6064	0,7472	0,6422
[10, 10]	0,2607	0,7444	0,7466	0,6060	0,7466	0,6410
[15, 15, 15]	0,3564	0,6467	0,7232	0,5974	0,7232	0,6402
[5, 5]	0,4521	0,7202	0,7207	0,5968	0,7207	0,6377
[7, 7, 7, 7]	0,3564	0,7444	0,7447	0,6049	0,7447	0,6366
[9]	0,2607	0,7450	0,7445	0,6049	0,7445	0,6363
[14]	0,2607	0,7444	0,7443	0,6047	0,7443	0,6357
[5, 5, 5]	0,3564	0,7444	0,7442	0,6047	0,7442	0,6355
[5, 5, 5, 5]	0,3564	0,6467	0,7441	0,6046	0,7441	0,6353
[10]	0,2607	0,7444	0,7439	0,6045	0,7439	0,6349
[19]	0,2607	0,7445	0,7439	0,6558	0,7439	0,6348
[12, 12, 12, 12, 12]	0,2607	0,7444	0,7438	0,6044	0,7438	0,6345

Trzywarstwowa sieć z 2 neuronami w każdej warstwie ([2, 2, 2]) osiągnęła najlepszy wynik (test_$F1$≈0,706, test_acc≈0,794), wyprzedzając jednowarstwową [11] (test_$F1$≈0,704, test_acc≈0,782). Mniejsze sieci ([1] lub pojedyncza warstwa z 12 neuronami) oraz większe konfiguracje ([3, 3, 3], [4, 4, 4, 4] itp.) dawały niższe wartości $F1$ (około 0,65–0,68) i gorszą dokładność testową. Zatem głęboka (3-warstwowa)

architektura z bardzo skromną liczbą neuronów okazała się najskuteczniejsza przy klasyfikacji.

Dobór zmiennych w zbiorze testowym

W celu oceny znaczenia poszczególnych zmiennych wejściowych w modelu klasyfikującym bilety lotnicze jako należące do taniej linii lub nie, przeprowadziliśmy eksperyment polegający na usuwaniu pojedynczych kolumn ze zbioru testowego i analizie zmian w metrykach modelu (analogicznie jak przy problemie regresyjnym). Testowany model to wielowarstwowa sieć neuronowa trenowana do klasyfikacji binarnej.

Nasz algorytm usuwa kolejno jedną ze zmiennych objaśnianych, a następnie sprawdza wartości statystyczne w pozostałym zestawie danych:

Poniżej przedstawiam wnioski na podstawie uzyskanych wyników:

usunięta_kolumna	test_accuracy	test_precision	test_recall	test_f1
Departure_time	0,532178218	0,345938278	0,591928251	0,436034
Arrival_time	0,551980198	0,551980198	1	0,711324
Flight_time	0,57640264	0,638063032	0,883408072	0,686163
Price	0,51369637	0,538136172	0,517189836	0,418818
Num_Layovers	0,531188119	0,441584158	0,8	0,569059
Cabin_bag	0,525577558	0,725944918	0,627503737	0,494697
Checked_bag	0,588613861	0,579084158	0,990732436	0,728859
Days_to_departure	0,507260726	0,329761065	0,59431988	0,424164
layover_duration	0,445544554	0,444379817	0,692077728	0,522924
Is_Departure_Warszawa	0,49950495	0,420792079	0,41793722	0,317451
Departure_time	0,501815182	0,51039604	0,259790732	0,238292
Arrival_time	0,533333333	0,641584158	0,803886398	0,576684
Flight_time	0,551980198	0,551980198	1	0,711324
Price	0,511056106	0,332838225	0,580269058	0,422864
Num_Layovers	0,531188119	0,441584158	0,8	0,569059
Cabin_bag	0,542574257	0,641584158	0,820627803	0,606457
Checked_bag	0,531683168	0,641584158	0,800896861	0,570845
Days_to_departure	0,551980198	0,551980198	1	0,711324
layover_duration	0,531188119	0,441584158	0,8	0,569059
Is_Arrival_Londyn	0,551980198	0,551980198	1	0,711324
Is_Arrival_Paryż	0,514851485	0,3337253	0,599701046	0,428806
Is_Arrival_Rzym	0,555610561	0,641584158	0,844245142	0,641519
Is_Flight_Monday	0,477887789	0,398338207	0,659790732	0,490114
Is_Flight_Saturday	0,635148515	0,632620731	0,94529148	0,746214
Is_Flight_Sunday	0,531683168	0,641584158	0,800896861	0,570845
Is_Flight_Thursday	0,518976898	0,531188119	0,61554559	0,455643
Is_Flight_Tuesday	0,531188119	0,441584158	0,8	0,569059

Is_Flight_Wednesday	0,445049505	0,476472441	0,760538117	0,579209
Is_Extraction_Monday	0,51369637	0,531188119	0,605979073	0,438405
Is_Extraction_Saturday	0,51039604	0,331188119	0,6	0,426794
Is_Extraction_Sunday	0,531188119	0,441584158	0,8	0,569059
Is_Extraction_Thursday	0,57970297	0,485733573	0,777877429	0,592187
Is_Extraction_Tuesday	0,530528053	0,441287457	0,798804185	0,56851
Is_Extraction_Wednesday	0,531188119	0,441584158	0,8	0,569059

- Usunięcie tej *Checked_bag* skutkuje spadkiem dokładności testowej, jednak mimo tego **wskaźniki precyzji, czułości i F1-score pozostają wysokie** (precision = 0,579, recall = 0,991, F1 = 0,729), czyli obecność bagażu rejestrowanego jest silnym wskaźnikiem przynależności biletu do taniej linii, szczególnie w kontekście czułość;
- W przypadku *Flight_time, Is_Flight_Saturday, Is_Arrival_Rzym oraz Is_Extraction_Thursday* model osiąga **wysoką dokładność oraz jedne z najwyższych wartości F1-score**. Pokazuje to, że czas lotu istotnie wpływa na jakość klasyfikacji;
- Jako mniej istotne zmienne zaklasyfikujemy *Is_Departure_Warszawa, Days_to_departure* oraz wykazujące wzajemne skorelowanie *Arrival_Time i Departure_Time* (możemy wykazywać ich skorelowanie z *Flight_Time)*. Usunięcie tych zmiennych ma umiarkowany wpływ na klasyfikacje – **nie są one kluczowe z punktu widzenia naszego modelu**.

Funkcja aktywacji

W tym przypadku przetestowaliśmy inny zestaw funkcji aktywacji:

- ReLU
- Sigmoid
- Tanh
- Leaky_ReLU
- Elu
- Swish

Oto wyniki testu:

Funkcja Aktywacji	train_accuracy	val_accuracy	test_accuracy	test_precision	test_recall	test_f1
relu	0,68	0,56	0,68	0,52	0,68	0,59
sigmoid	0,81	0,80	0,80	0,64	0,80	0,72
tanh	0,81	0,80	0,80	0,64	0,80	0,72
leaky_relu	0,44	0,56	0,81	0,68	0,81	0,72
elu	0,56	0,56	0,80	0,64	0,80	0,72
swish	0,44	0,56	0,80	0,64	0,80	0,72

Sigmoid

- **Zalety:**
  - Wysoka **train_accuracy** (0,81) i **val_accuracy** (0,80) wskazują na dobrą zdolność do nauki i generalizacji.
  - **test_accuracy** (0,80), **test_precision** (0,64), **test_recall** (0,80) i **test_f1** (0,72) są również wysokie, co świadczy o dobrej wydajności na danych testowych.
  - **test_log_loss** (0,50) jest bardzo niska, co jest pożądane.
- **Wady:** Brak istotnych wad w tym zestawie danych.

Tanh

- **Zalety:**
  - Wyniki dla Tanh są praktycznie identyczne jak dla Sigmoid, z wysoką dokładnością na wszystkich zbiorach danych.
  - **test_log_loss** (0,50) również jest bardzo niska.
- **Wady:** Brak istotnych wad.

ReLU

- **Zalety:**
  - **test_accuracy** (0,68) jest umiarkowana.
- **Wady:**
  - Niska **train_accuracy** (0,68) i **val_accuracy** (0,56) potwierdzają słabą zdolność do nauki i generalizacji.
  - **test_precision** (0,52) i **test_f1** (0,59) są stosunkowo niskie.
  - **test_log_loss** (2,26) jest znacznie wyższa niż dla Sigmoid/Tanh.

Leaky_ReLU

- **Zalety:**
  - **test_accuracy** (0,81) jest wysoka, nawet wyższa niż dla Sigmoid/Tanh.
  - **test_precision** (0,68), **test_recall** (0,81) i **test_f1** (0,72) również są dobre.
- **Wady:**
  - Niska **train_accuracy** (0,44) i **val_accuracy** (0,56) wskazują na poważne problemy z treningiem modelu. Może to sugerować niestabilność procesu uczenia lub trudności w optymalizacji.
  - Wysoki **test_log_loss** (0,59) również wskazuje na niższe pewności predykcji.

Elu

- **Zalety:**
  - Umiarkowana **test_accuracy** (0,80).

- **Wady:**
    - Niska **train_accuracy** (0,56) i **val_accuracy** (0,56) sugerują problemy z uczeniem.

Swish

- **Zalety:**
    - Umiarkowana **test_accuracy** (0,80).
- **Wady:**
    - Niska **train_accuracy** (0,44) i **val_accuracy** (0,56) świadczą o słabej wydajności podczas treningu.
    - **test_log_loss** (4,68) jest drastycznie wysoka, co oznacza, że model jest bardzo niepewny w swoich predykcjach.

Na podstawie kompleksowej analizy, **Sigmoid** i **Tanh** są zdecydowanie najlepszymi funkcjami aktywacji dla tego konkretnego zadania klasyfikacji lotów. Osiągają one najlepsze wyniki we wszystkich kluczowych metrykach, w tym wysoką dokładność i niską val-loss, co świadczy o stabilnym treningu i wysokiej jakości predykcji.

Która funkcja jest najlepsza?

Trudno jest wskazać absolutnego zwycięzcę między Sigmoid a Tanh, ponieważ ich wyniki są niemal identyczne. Jednak w oparciu o dostępne dane, zarówno **Sigmoid**, jak i **Tanh** są wysoce rekomendowane.

Proporcje podziału danych

W przypadku sieci klasyfikacyjnej również postanowiliśmy przetestować kilka kombinacji:

- 0.8:0.1:0.1
- 0.7:0.15:0.15
- 0.6:0.2:0.2
- 0.65:0.175:0.175

Oto wyniki testów:

proporcja	train_accuracy	val_accuracy	test_accuracy	test_precision	test_recall	test_f1
[0.8, 0.1, 0.1]	0,80	0,81	0,81	0,65	0,81	0,72
[0.7, 0.15, 0.15]	0,80	0,81	0,81	0,65	0,81	0,72
[0.6, 0.2, 0.2]	0,68	0,81	0,81	0,65	0,81	0,72
[0.65, 0.175, 0.175]	0,68	0,68	0,68	0,53	0,68	0,59

1. Proporcja [0.8, 0.1, 0.1]

- **Zalety:**
  - Wysoka **train_accuracy** (0,80) i **val_accuracy** (0,81) potwierdzają skuteczność treningu i zdolność do predykcji na nieznanych danych walidacyjnych.
  - Najlepsze wyniki na zbiorze testowym: **test_accuracy** (0,81), **test_precision** (0,65), **test_recall** (0,81) i **test_f1** (0,72).
  - Najniższe **test_log_loss** (0,49), co świadczy o wysokiej pewności predykcji modelu.
- **Wady:** Brak istotnych wad.

2. Proporcja [0.7, 0.15, 0.15]

- **Zalety:**
  - Wyniki dla tej proporcji są praktycznie identyczne jak dla [0.8, 0.1, 0.1], z wysoką **train_accuracy** (0,80) i **val_accuracy** (0,81).
  - Bardzo dobre wyniki na zbiorze testowym: **test_accuracy** (0,81), **test_precision** (0,65), **test_recall** (0,81) i **test_f1** (0,72).
  - Niskie **test_log_loss** (0,49).
- **Wady:** Brak istotnych wad.

3. Proporcja [0.6, 0.2, 0.2]

- **Zalety:**
  - Wysoka **val_accuracy** (0,81) i **test_accuracy** (0,81) są zaskakująco dobre, biorąc pod uwagę inne metryki.
  - Dobre wyniki na zbiorze testowym w kategoriach **test_precision** (0,65), **test_recall** (0,81) i **test_f1** (0,72).
- **Wady:**
  - Niższa **train_accuracy** (0,68), co wskazuje na gorsze dopasowanie modelu do danych treningowych.
  - Wyższe **test_log_loss** (1,78), co oznacza mniejszą pewność predykcji.

4. Proporcja [0.65, 0.175, 0.175]

- **Zalety:** Brak znaczących zalet w porównaniu z innymi proporcjami.
- **Wady:**
  - Najniższe **train_accuracy** (0,68), **val_accuracy** (0,68) i **test_accuracy** (0,68), co świadczy o najgorszej wydajności modelu.
  - Najniższe wartości **test_precision** (0,53), **test_recall** (0,68) i **test_f1** (0,59) na zbiorze testowym.
  - Najwyższe **test_log_loss** (2,05), co oznacza bardzo niską pewność predykcji.

Na podstawie analizy, proporcje **[0.8, 0.1, 0.1]** i **[0.7, 0.15, 0.15]** są zdecydowanie najlepszymi wyborami dla tej sieci klasyfikacyjnej. Obydwie proporcje zapewniają stabilne uczenie, niskie straty i wysoką dokładność na wszystkich zbiorach danych, a także niską log-loss.

Najlepsza proporcja

Trudno jest wskazać absolutnego zwycięzcę między **[0.8, 0.1, 0.1]** a **[0.7, 0.15, 0.15]**, ponieważ ich wyniki są praktycznie identyczne. Wybór między nimi może zależeć od dostępnej wielkości zbioru danych i preferencji dotyczących balansu między rozmiarem zbioru treningowego a zbiorami walidacyjnym i testowym.

Oversampling na zbiorze treningowym

Balanced by	train_accuracy	val_accuracy	test_accuracy	test_precision	test_recall	test_f1
Flight_weekday	0,45	0,65	0,65	0,56	0,65	0,53
Extraction_Weekday	0,26	0,74	0,74	0,65	0,74	0,63
Num_Layovers	0,53	0,74	0,74	0,54	0,74	0,63
arr_city	0,24	0,74	0,74	0,60	0,74	0,63
dep_city	0,36	0,65	0,64	0,54	0,64	0,53
Cabin_bag	0,50	0,64	0,74	0,55	0,74	0,63
Checked_bag	0,28	0,74	0,74	0,54	0,74	0,62

Jednym z testowanych parametrów, podobnie jak w przypadku sieci regresyjnej, był sposób zbilansowania zbioru treningowego (poprzez oversampling), tak, by liczność podzbiorów obserwacji z poszczególnymi wartościami w kolumnie, wg której bilansujemy, była równa. Wyniki wdrożenia tej procedury przedstawia poniższa tabela. Widzimy, że opisane postępowanie nie wpływa znacząco na jakość klasyfikacji. Uzyskany dla większości zmiennych recall sugeruje dość dobry odsetek wykrycia True Positives (czyli linii lotniczej, która jest tanią linią lotniczą). Precision jest niższy, co oznacza, że wśród obserwacji zaklasyfikowanych jako low-cost występują liczne obserwacje niebędące w rzeczywistości tanimi liniami lotniczymi (False Positives). Wartości metryki f1, czyli średniej harmonicznej precision i recall, sugerują, że model nie jest bardzo skuteczny.

RandomForest

Metodologia

Jednym z powszechnie stosowanych w problemie predykcji ceny biletu lotniczego algorytmów jest las losowy. Model składa się z wielu drzew – a każde z nich dokonuje niezależnej predykcji. W klasycznej wersji algorytmu każde drzewo trenowane jest na niezależnej próbie bootstrapowej (t.j. wylosowanych domyślnie ze zwracaniem obserwacjach) ze zbioru treningowego. Ta technika znana jest jako *bagging* (bootstrap aggregating) [8]. Pierwszym parametrem, który możemy ustawić, jest zatem rozmiar tej próby. W naszym badaniu odpowiada mu parametr *sample size* – stosunek liczby obserwacji próby do liczby obserwacji zbioru treningowego. Drugim jest parametr określający, czy losujemy ze zwracaniem. W procesie trenowania losowany jest jednak nie tylko podzbiór obserwacji, ale i zmiennych. Każde drzewo dostaje zbiór zawierający nie wszystkie, a wylosowane bez zwracania zmienne. Ich liczbę będzie określał parametr *num_inputs*. Przyjmuje się [7], że dla problemu regresji domyślną wartością tego parametru powinna być $\frac{1}{3}p$, a dla problemu klasyfikacyjnego $\sqrt{p}$, gdzie p oznacza liczbę zmiennych. Uwzględnienie losowości w procesie trenowania lasu losowego powoduje, że nie występuje korelacja pomiędzy przewidywanymi przez każde drzewo wartościami. Jest to bardzo pożądana własność, która poprawia jakość ostatecznej predykcji modelu. Kolejnym hiperparametrem jest liczba drzew, które składają się na las losowy. Zwiększenie go pomaga ustabilizować wariancję predykcji, jednak należy mieć na uwadze, że czas potrzebny na wytrenowanie modelu wzrasta liniowo wraz z liczbą drzew. W naszych badaniach będzie ona reprezentowana przez parametr *num_trees*. W literaturze [7] zaleca się domyślną wartość równą 10p (oznaczenia jak wyżej). Ostatnie parametry bezpośrednio wpływają na strukturę drzewa – to odpowiednio *min_samples_split* oraz *min_samples_leaf*. Pierwszy z nich określa, ile obserwacji minimalnie musi znaleźć się w węźle drzewa, by dokonany został jego podział na dwa węzły-dzieci. *Min_samples leaf* z kolei wskazuje minimalną liczbę obserwacji w każdym z węzłów-dzieci, która konieczna jest do dokonania podziału. Jeżeli więc w wyniku ustalenia optymalnego progu podziału (takiego, który minimalizuje MSE) do pierwszego węzła trafi 99 obserwacji, a do drugiego jedna, to przy ustawionym *min_samples_leaf* =2 taki podział nie będzie możliwy. Zwiększanie obu tych parametrów będzie skutkowało płytszymi drzewami i większą liczbą obserwacji w liściach (końcowych węzłach).

Schemat trenowania pojedynczego drzewa można opisać w następujących krokach:

1. Wylosuj próbę bootstrapową obserwacji o liczności określonej przez *sample size*
2. Wylosuj podzbiór zmiennych, na których wytrenowane będzie dane drzewo. Otrzymany podzbiór zbioru treningowego trafia do pierwszego węzła.

3. Dla każdego węzła niebędącego liściem wykonuj:
4. Jeśli w węźle jest mniej *niż num_samples_split* obserwacji, przerwij podział i idź do 3 (otrzymany węzeł jest liściem). W przeciwnym razie idź do 5.
5. Ustal wszystkie możliwe kombinacje (zmienna, próg podziału) podziału danych na dwa mniejsze podzbiory.
6. Dla każdej kombinacji oblicz wartość funkcji straty – w przypadku lasu regresyjnego będzie to spadek MSE.
7. Wybierz kombinację zapewniającą maksymalną stratę (spadek błędu)
8. Jeśli w wyniku podziału według kombinacji w którejkolwiek z grup znalazłoby się mniej niż *num_samples_leaf* obserwacji, odrzuć tę kombinację i idź do 7. W przeciwnym razie idź do 3.

Wytrenowanie modelu RF polega na wielokrotnym powtórzeniu powyższej procedury. Swoją skuteczność model zawdzięcza temu, że ostateczna predykcja powstaje przez odpowiednie zagregowanie predykcji każdego z drzew w lesie. Błąd popełniony przez pojedyncze drzewo może być znaczny, jednak uśrednienie przewidywanych wartości istotnie go redukuje. Trenowanie dużej liczby drzew może być czasochłonne, ale drzewa nie muszą być trenowane sekwencyjnie (jak na przykład w przypadku modelu wzmocnienia gradientowego). Niezależność predykcji jednego drzewa od predykcji reszty umożliwia ich współbieżne trenowanie.

Regresja

Testowanie parametrów

Dobór zmiennych w zbiorze testowym

W celu oceny istotności poszczególnych zmiennych w modelu lasu losowego, przeprowadziliśmy analizę wpływu usunięcia pojedynczych cech na jakość predykcji. Kolejno usuwaliśmy ją ze zbioru testowego, po czym dokonywaliśmy predykcji przy użyciu wcześniej wytrenowanego modelu lasu losowego. Taka procedura pozwala zidentyfikować zmienne, które mają największy wpływ na przewidywaną wartość ceny biletu. Jednocześnie umożliwia wykrycie cech zbędnych lub nawet zakłócających, których obecność nie wpływa znacząco na końcowy wynik predykcji lub prowadzi do nadmiernego dopasowania.

usunięta kolumna	test_MSE	test_R2	test_MAE	test_MAPE	OOB_R2	OOB_MSE
Departure_time	228047,027	0,7854649	309,7108201	37,87549856	0,776520319	239505,309
Arrival_time	224046,298	0,7892285	307,8100526	37,81527589	0,780059608	235712,2198
Flight_time	233650,904	0,780193	312,3129794	38,23109287	0,770143145	246339,7883
Num_Layovers	222071,181	0,7910866	305,6664365	37,29049883	0,78203185	233598,549
Cabin_bag	231446,319	0,782267	318,8644468	42,5223434	0,773037641	243237,7287
Checked_bag	865182,674	0,1860797	526,7868285	56,03855989	0,186245773	872108,1804
Days_to_departure	251775,983	0,7631418	335,9844068	42,07472693	0,752919077	264798,9855
layover_duration	237411,821	0,7766549	313,4292018	37,6921153	0,767719077	248937,6849

- Największy wpływ na jakość modelu ma zmienna *Checked_bag* – po jej usunięciu nastąpił **drastyczny wzrost błędu.**
- *Num_Layovers* również istotnie wpływa na predykcję, usunięcie jej spowodowało zauważalny spadek jakości modelu (wzrost MSE i MAE).
- Zmienna *Cabin_bag* oraz *Flight_time* wydaje się mniej istotna – jej usunięcie nie powoduje dużego pogorszenia MSE ani R^2, choć MAPE wzrosło zauważalnie, co może sugerować wpływ tej cechy na dokładność względną.
- Zmienne *Departure_time, Arrival_time* i *layover_duration* mają stosunkowo **niewielki wpływ na wyniki modelu** – usunięcie którejkolwiek z nich powoduje tylko nieznaczne pogorszenie metryk.

Liczba drzew

Testowanie liczby drzew w Random Forest jest częścią procesu **strojenia hiperparametrów** i pozwala na:

- **Znalezienie "złotego środka"** między dokładnością a efektywnością obliczeniową.
- Upewnienie się, że model jest **wystarczająco stabilny i generalizuje** dobrze na nowych, niewidzianych danych.
- Zrozumienie, **jak wiele drzew jest wystarczające**, aby uzyskać optymalne wyniki dla konkretnego zbioru danych i problemu.

W ramach analizy przetestowano wpływ liczby drzew na testowe MSE w przypadku, gdy każde drzewo trenowane jest na podzbiorze 6 i 12 zmiennych. Te wartości odpowiadają przyjętym w literaturze [7] wartościom: pierwiastka kwadratowego i 1/3 łącznej liczby zmiennych.

Wyniki z testowania Liczby Drzew:

- Dla 6 zmiennych:

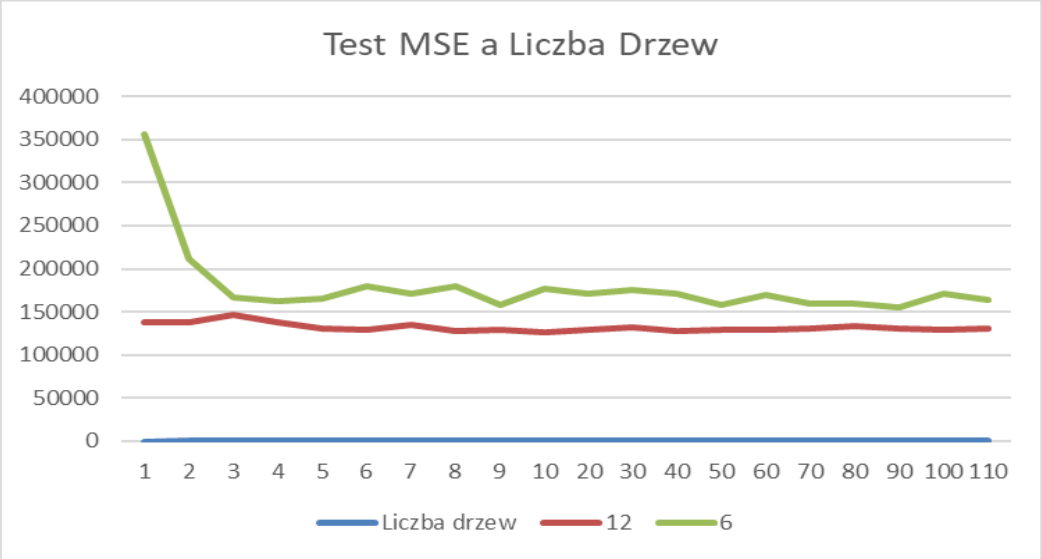
Liczba drzew	train_MSE	test_MSE	test_MAE	test_MAPE	test_R2	train_time	OOB_MSE	OOB_R2
1	345995,25	356137,93	338,42	41,70	0,67	1,48	349417,85	0,68
2	205665,52	211842,56	307,85	39,63	0,80	2,71	286015,90	0,73
3	162528,96	166402,25	270,44	36,25	0,85	4,36	207958,87	0,81
4	159197,63	162393,18	268,62	35,08	0,85	4,21	181905,23	0,83
5	161807,94	165456,53	272,12	36,47	0,85	6,49	210349,44	0,80
6	175040,10	179123,44	281,17	37,84	0,83	5,83	218603,56	0,80
7	166397,51	170635,47	277,08	37,03	0,84	7,23	198229,36	0,81
8	176523,01	180238,08	291,74	38,63	0,83	8,15	210186,08	0,80
9	154777,52	158531,42	274,26	37,21	0,85	9,61	166647,67	0,84
10	174098,67	177591,15	293,17	39,65	0,84	13,52	211164,05	0,80
20	167384,57	171462,36	284,67	38,41	0,84	21,05	180289,40	0,83
30	171569,97	175914,67	287,44	38,63	0,84	29,26	180868,74	0,83
40	166108,19	170452,00	282,19	37,36	0,84	67,29	173000,12	0,84

50	154356,49	157774,20	277,00	38,09	0,85	61,29	158752,92	0,85
60	165854,64	169859,83	283,39	37,77	0,84	52,79	169604,51	0,84
70	155846,62	159372,37	276,14	37,34	0,85	62,61	159099,15	0,85
80	155333,87	159702,90	275,51	37,89	0,85	76,55	158608,10	0,85
90	152198,27	155757,98	273,85	36,85	0,86	82,61	154527,92	0,86
100	166598,80	171674,91	284,55	38,06	0,84	93,22	169653,83	0,84
110	159955,12	164154,36	279,46	37,68	0,85	101,19	161842,98	0,85

- Dla 12 zmiennych:

Liczba drzew	train_MSE	test_MSE	test_MAE	test_MAPE	test_R2	train_time	OOB_MSE	OOB_R2
1	135468,48	137706,00	252,38	33,13	0,87	1,76	136331,77	0,87
2	132376,15	137169,04	246,60	31,94	0,87	3,64	141748,07	0,87
3	141833,35	146521,52	258,02	33,09	0,86	4,63	159277,12	0,85
4	134440,26	137916,99	252,80	33,03	0,87	6,92	149852,55	0,86
5	126190,92	130135,94	245,56	32,96	0,88	9,38	137405,58	0,87
6	125397,43	129786,47	246,81	32,22	0,88	9,98	135340,39	0,87
7	130860,03	135448,28	254,61	34,51	0,87	12,37	141831,16	0,87
8	124037,00	128460,92	243,82	31,38	0,88	13,06	133143,11	0,88
9	125751,10	129022,56	248,18	33,78	0,88	15,75	133888,59	0,87
10	123080,70	125773,44	244,75	33,00	0,88	14,31	130046,57	0,88
20	126219,53	129564,59	247,49	33,33	0,88	34,10	129573,48	0,88
30	128373,32	132135,41	246,91	32,81	0,88	56,85	131182,78	0,88
40	124190,57	128023,77	246,01	33,24	0,88	64,46	126727,61	0,88
50	125959,87	129687,94	247,30	33,42	0,88	118,36	127785,94	0,88
60	125364,84	128844,37	247,21	33,35	0,88	115,72	127163,16	0,88
70	127606,16	131202,73	248,10	33,17	0,88	110,10	129091,03	0,88
80	129128,33	133055,17	249,25	33,43	0,88	134,01	130475,66	0,88
90	127044,49	130626,46	246,74	33,04	0,88	147,54	128716,80	0,88
100	126091,23	129616,56	246,97	33,47	0,88	164,65	127389,42	0,88
110	127110,97	131099,91	247,42	33,15	0,88	182,68	128185,41	0,88

Wykres obrazujący jak zmienia się MSE przy zwiększaniu liczby drzew:



Wnioski dotyczące wyboru liczby drzew:

1. **Stabilizacja metryk jakości:**
    a. test_R2 oraz OOB_R2 (współczynnik determinacji) bardzo szybko osiągają wysoki poziom. W obu przypadkach już przy 6-10 drzewach test_R2 stabilizuje się na poziomie >0.80, i nie wykazują dalszej istotnej poprawy wraz ze wzrostem liczby drzew.
    b. test_MAE (średni błąd bezwzględny) dla 12 zmiennych również stabilizuje się relatywnie wcześnie, oscylując wokół wartości 243-249 dla 10 drzew i więcej, w przypadku dla 6 zmiennych ta stabilizacja nie jest tak precyzyjna.
2. **Zachowanie MSE (średni błąd kwadratowy):**
    a. test_MSE oraz OOB_MSE wykazują tendencję spadkową, ale korzyści stają się coraz mniejsze po przekroczeniu pewnej liczby drzew.
    Wartości spadają coraz mniej, co widoczne jest na wykresie.

Najlepszym modelem będzie ta z 12 zmiennymi, ma niższe wyniki MSE a także minimalnie wyższe R^2

Analiza danych wskazuje, że:

- **Punkt optymalny dla MSE:** Najniższe wartości test_MSE (128023) oraz OOB_MSE (126727) obserwujemy przy **40 drzewach**. W tym punkcie test_R2 wynosi już 0.88.
- **Kompromis między jakością a czasem:**
    - Już przy **10-20 drzewach** model osiąga bardzo dobre wyniki (test_R2 = 0.88, OOB_R2 = 0.87-0.88, test_MSE ~129000-129500), przy znacznie krótszym czasie uczenia (16-34 sekundy).
    - Zwiększenie liczby drzew z 40 do 60 nieznacznie pogarsza test_MSE (z 128023 do 128844) i OOB_MSE (z 126727 do 127163).

Podsumowując: Analiza wskazuje, że **40 drzew** stanowi najlepszy punkt dla modelu z 12 zmiennymi pod względem minimalizacji błędu MSE.

Min_samples_leaf i min_samples_split

Testowanie parametrów struktury drzew w Random Forest pozwala na:

- Dopasowanie głębokości i rozgałęzienia drzew w celu ograniczenia przeuczenia modelu.
- Zwiększenie stabilności predykcji przez kontrolę nad minimalną liczbą obserwacji wymaganych do podziału oraz do utworzenia liścia.
- Zidentyfikowanie konfiguracji parametrów, które zapewniają równowagę między jakością predykcji a czasem trenowania modelu.

W ramach analizy przetestowano wpływ kombinacji wartości min_samples_split i min_samples_leaf na jakość modelu przy stałej liczbie drzew (20). Najniższy testowy błąd MSE (76 850) uzyskano dla split = 5, leaf = 1, jednak z uwagi na ryzyko przeuczenia, rekomendowane jest użycie leaf = 2, które również zapewnia wysoką jakość predykcji i lepszą generalizację.

Test potwierdził, że odpowiednie ustawienie tych parametrów ma istotny wpływ na równowagę między dokładnością modelu a jego zdolnością do generalizacji.

split	leaf	Test_MSE	Test_MAE	Test_R2	Train_Time
2	1	76963.89	171.3662	0.929192	368.0581
2	2	77874.21	170.7458	0.928354	333.2622
2	5	83778.3	177.844	0.922922	265.7295
2	10	92915.73	188.8621	0.914516	227.0526
5	1	76850.69	170.4855	0.929296	336.9744
5	2	79444.91	172.5838	0.926909	325.2572
5	5	83778.3	177.844	0.922922	265.7359
5	10	92915.73	188.8621	0.914516	227.7056
10	1	77152.22	172.2182	0.929018	312.8739
10	2	79264.73	172.2427	0.927075	298.8046
10	5	83778.3	177.844	0.922922	266.5866
10	10	92915.73	188.8621	0.914516	229.8688
20	1	79371.71	172.6341	0.926976	290.017
20	2	80298.41	173.859	0.926124	282.8466
20	5	86507.79	180.9368	0.920411	248.9822
20	10	92915.73	188.8621	0.914516	227.3418

Stabilizacja metryk jakości:

Test_R^2 (współczynnik determinacji) bardzo szybko osiąga wysoki poziom. Dla wszystkich kombinacji parametrów przy 40 drzewach jego wartość utrzymuje się na poziomie powyżej

0.92. Najlepsze wartości (0.9291–0.9293) występują przy niskich wartościach min_samples_split = 2–5 i min_samples_leaf = 1–2.

Test_MAE (średni błąd bezwzględny) również stabilizuje się relatywnie wcześnie – najlepsze wartości mieszczą się w przedziale 170–173 i obserwowane są dla leaf = 1–2.

Zachowanie MSE (średni błąd kwadratowy):

Test_MSE osiąga najniższe wartości przy niskich wartościach split i leaf. Minimalny poziom błędu obserwowany jest przy min_samples_split = 5 i min_samples_leaf = 1 (Test_MSE = 76 850), jednak tak niski poziom parametru liścia może powodować przeuczenie modelu i dopasowanie do szumu w danych. W praktyce, **należy unikać wartości min_samples_leaf = 1**, aby zapewnić lepszą generalizację modelu.

Przy leaf = 2, model osiąga nadal bardzo dobre wyniki (Test_MSE ≈ 78 000, Test_R^2 ≈ 0.928–0.929), przy znacznie mniejszym ryzyku nadmiernego dopasowania.

Czas trenowania:

Czas uczenia skraca się wraz ze wzrostem leaf i split. Dla split = 2 i leaf = 1 czas wynosi 368 sekund, natomiast dla split = 20 i leaf = 10 – tylko 227 sekund. Jednocześnie w tych uproszczonych modelach spada jakość predykcji.

Analiza danych wskazuje, że:

Punkt optymalny dla MSE: Najniższa wartość test_MSE występuje przy min_samples_split = 5, min_samples_leaf = 1, jednak z uwagi na ryzyko przeuczenia modelu, **lepszym kompromisem** jest parametr min_samples_leaf = 2, który zapewnia bardzo dobrą jakość przy lepszej stabilności modelu.

Kompromis między jakością a czasem:

Wartości split = 5–10 oraz leaf = 2 umożliwiają uzyskanie bardzo dobrych wyników jakościowych (Test_R^2 ≈ 0.928) przy czasie treningu krótszym o kilkadziesiąt sekund względem najbardziej złożonych modeli.

Podsumowanie:

Model z parametrami min_samples_split = 5 i min_samples_leaf = 2 stanowi **najlepszy kompromis między dokładnością predykcji a stabilnością i czasem treningu**, jednocześnie ograniczając ryzyko przeuczenia, które mogłoby wystąpić przy zbyt małej liczbie obserwacji w liściu drzewa.

Klasyfikacja

Podobnie jak w przypadku sieci neuronowej próbujemy sklasyfikować dany lot do kategorii binarnej: czy dany lot jest realizowany przez tanią linię lotniczą?
Metodologia jest podobna jak przy sieci neuronowej, natomiast w tym przypadku korzystamy z Random Forest.
Dla danego zestawu parametrów uczenie powtarzaliśmy trzykrotnie, a następnie ze statystyk liczyliśmy średnie - ma to uwiarygodnić wyniki.

Poniższa tabela przedstawia parametry pierwotnego modelu, który próbowaliśmy optymalizować:

Liczba Drzew	max_depth	min_samples_split	min_samples_leaf	max_features	bootstrap
5	10	5	2	5	PRAWDA

Testowanie parametrów

Liczba drzew

Liczba Drzew	train_precision	train_recall_	train_f1	test_accuracy	test_precision	test_recall	test_f1	oob_accuracy	oob_precision	oob_recall	oob_f1	time_sec
1	0,999	0,998	0,998	0,999	0,999	0,997	0,998	0,999	0,998	0,997	0,998	4,934
2	1,000	0,997	0,999	0,999	1,000	0,996	0,998	0,999	0,999	0,997	0,998	10,930
3	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	0,999	0,998	0,999	15,332
4	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	1,000	0,998	0,999	17,330
5	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	1,000	0,998	0,999	22,997
6	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	1,000	0,998	0,999	33,360
7	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	1,000	0,998	0,999	40,235
8	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,999	1,000	0,998	0,999	45,762
9	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	0,999	0,999	55,208
10	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	0,999	0,999	58,237
20	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	107,132

30	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	164,450
40	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	205,415
50	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	252,404
60	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	314,421
70	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	355,563
80	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	400,347
90	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	450,449
100	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	488,979
110	1,000	1,000	1,000	1,000	1,000	0,999	1,000	1,000	1,000	1,000	1,000	534,285

Liczba zmiennych

max_features	train_accuracy	train_precision	train_recall	train_f1	test_accuracy	test_precision	test_recall	test_f1	oob_accuracy	oob_precision	oob_recall	oob_f1	time_sec
8	0,9999	0,9999	0,9999	0,9999	0,9999	1,0000	0,9997	0,9998	0,9990	0,9997	0,9963	0,9980	38,8176
5	0,9997	0,9998	0,9988	0,9993	0,9996	0,9998	0,9988	0,9993	0,9846	0,9953	0,9448	0,9693	26,6425
4	0,9992	0,9999	0,9968	0,9984	0,9993	0,9999	0,9974	0,9986	0,9817	0,9930	0,9359	0,9633	20,1476
Wszystkie zmienne	0,9999	0,9999	0,9999	0,9999	0,9999	1,0000	0,9998	0,9999	0,9999	0,9999	0,9998	0,9998	110,1563

Min Samples Leaf

min_samples_leaf	train_accuracy	train_precision	train_recall	train_f1	test_accury	test_precision	test_recall	test_f1	oob_accuracy	oob_precision_mean	oob_recall	oob_f1	time_sec
1	0,9999	1,0000	0,9998	0,9999	0,9998	0,9998	0,9994	0,9996	0,9885	0,9951	0,9602	0,9773	68,1263
2	0,9998	1,0000	0,9993	0,9996	0,9998	0,9999	0,9993	0,9996	0,9898	0,9972	0,9633	0,9799	37,8756
4	0,9992	0,9999	0,9971	0,9985	0,9989	0,9999	0,9958	0,9978	0,9915	0,9983	0,9687	0,9832	27,0470
6	0,9990	0,9998	0,9963	0,9981	0,9988	0,9997	0,9958	0,9977	0,9897	0,9967	0,9632	0,9796	36,5872

Max Depth

max_depth	train_accuracy	train_precision	train_recall	train_f1	test_accuracy	test_precision	test_recall	test_f1	oob_accuracy	oob_precision	oob_recall	oob_f1	time_sec
5	0,962	0,999	0,854	0,918	0,962	0,999	0,854	0,918	0,920	0,966	0,714	0,818	34,708
10	0,999	1,000	0,998	0,999	0,999	1,000	0,997	0,999	0,988	0,998	0,954	0,975	21,802
20	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,998	0,999	0,995	0,997	23,689
25	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,999	0,999	0,995	0,997	23,384
50	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,999	0,999	0,995	0,997	23,130

Min Samples Split

min_samples_split	train_accuracy	train_precision	train_recall	train_f1	test_accuracy	test_precision	test_recall	test_f1	oob_accuracy	oob_precision	oob_recall	oob_f1	time_sec
2	0,998	1,000	0,994	0,997	0,998	1,000	0,993	0,996	0,983	0,998	0,935	0,965	21,006
5	0,999	1,000	0,995	0,997	0,998	1,000	0,994	0,997	0,986	0,998	0,950	0,973	21,131
7	0,999	1,000	0,997	0,998	0,999	1,000	0,996	0,998	0,984	0,993	0,945	0,969	22,444
10	1,000	1,000	1,000	1,000	1,000	1,000	0,999	1,000	0,992	0,997	0,972	0,985	26,795

Wnioski

Testowanie parametrów pokazuje, że każdy model ma fantastyczne wyniki. Nie jesteśmy w stanie określić, dlaczego tak się dzieje. Kod działa poprawnie, statystyki liczone są perfekcyjnie oraz dane ładowane są prawidłowo.

K najbliższych sąsiadów

Jednym z najprostszych, a zarazem intuicyjnych algorytmów wykorzystywanych w problemie regresji, takim jak predykcja ceny biletu lotniczego, jest model k najbliższych sąsiadów (k-NN). W odróżnieniu od metod takich jak drzewa decyzyjne czy sieci neuronowe, k-NN nie buduje jawnego modelu podczas etapu trenowania – zamiast tego zapamiętuje cały

zbiór treningowy i wykonuje predykcję dopiero w momencie prognozy nowej obserwacji. Takie podejście określane jest jako uczenie leniwe.

Kluczową ideą metody k-NN jest założenie, że wartość zmiennej objaśnianej dla nowego punktu jest zbliżona do wartości w punktach znajdujących się w jego najbliższym sąsiedztwie. Model porównuje nową obserwację ze wszystkimi przykładami treningowymi, mierząc odległość pomiędzy punktami – w naszej implementacji zastosowana została metryka euklidesowa. Następnie wybierane jest k najbliższych sąsiadów (punktów o najmniejszej odległości), a wartość prognozowana to średnia wartości tych sąsiadów.

Pierwszym i podstawowym hiperparametrem modelu jest n_neighbors, który określa liczbę sąsiadów uwzględnianych przy predykcji. Ma on bezpośredni wpływ na złożoność i gładkość funkcji predykcyjnej. Dla małych wartości k model może być podatny na szum (niskie k = wysoka wariancja), natomiast dla dużych – staje się bardziej ogólny i może przeoczyć lokalne wzorce (wysokie k = wysoki bias).

Drugim istotnym parametrem jest sposób ważenia sąsiadów. W naszej implementacji obsługiwane są dwie opcje:

- *uniform* – każdy z k sąsiadów ma równy wpływ na wynik,
- *distance* – sąsiedzi bliżsi mają większą wagę (odwrotność odległości), co pozwala uwzględnić, że bardziej zbliżone punkty mogą lepiej odzwierciedlać wartość predykowaną.

Etap trenowania modelu k-NN ogranicza się wyłącznie do zapamiętania danych treningowych – nie zachodzi tu żadna forma parametryzacji czy optymalizacji. Proces predykcji można opisać w następujących krokach:

1. Dla nowej obserwacji oblicz odległości do wszystkich punktów w zbiorze treningowym.
2. Wybierz k punktów o najmniejszych odległościach.
3. Oblicz wartość predykcji jako średnią wartości zmiennej objaśnianej wśród wybranych sąsiadów – przy uwzględnieniu wag, jeśli wskazano *weights=distance*.

Mimo że k-NN jest prosty w implementacji i łatwy do zrozumienia, ma też pewne ograniczenia: czas predykcji rośnie wraz z liczebnością zbioru treningowego, a jego wydajność może być znacznie obniżona w przypadku danych o wysokiej wymiarowości.

Regresja

Testowanie parametrów

Dobór najlepszych zmiennych

W celu oceny znaczenia poszczególnych cech w procesie predykcji, przeprowadziliśmy analizę wpływu usunięcia pojedynczych zmiennych ze zbioru testowego na jakość działania modelu. Dla każdego eksperymentu usuwaliśmy jedną zmienną,

pozostawiając pozostałe dane bez zmian, a następnie dokonywaliśmy predykcji przy użyciu uprzednio wytrenowanego modelu. Tego rodzaju podejście pozwala ocenić, jak istotna była dana cecha dla trafności prognozy – spadek jakości predykcji po jej usunięciu wskazuje na jej wysoką wartość informacyjną. Z kolei brak istotnych zmian sugeruje, że cecha mogła być zbędna lub nadmiarowa.

Taka procedura może również ujawnić zmienne wprowadzające szum do modelu – ich usunięcie może w niektórych przypadkach prowadzić nawet do poprawy wyników.

usunięta kolumna	train_MSE	test_MSE	test_R2	test_MAE	test_MAPE
Departure_time	4534,865667	235902,791	0,7806268	291,7288963	32,36606667
Arrival_time	4503,547333	234054,101	0,7823459	290,5288891	32,18206993
Flight_time	4499,440333	246981,677	0,7703242	299,1905906	32,6463002
Num_Layovers	4499,440333	235387,764	0,7811057	290,821914	32,08305296
Cabin_bag	4499,440333	251904,902	0,7657459	309,3529656	39,01322027
Checked_bag	33662,43378	1058169,6	0,0159757	574,1323535	58,1562209
Days_to_departure	84656,42507	279834,98	0,7397729	328,2914696	39,27921603
layover_duration	5328,831222	246716,651	0,7705706	299,2765325	32,91488203

- Największy wpływ na jakość predykcji miało usunięcie zmiennej *Checked_bag* – doprowadziło to do drastycznego pogorszenia wszystkich metryk, co wskazuje na jej kluczowe znaczenie w modelu.
- Znaczące pogorszenie wyników odnotowano również po usunięciu *Days_to_departure*, *Cabin_bag* oraz *Flight_time*, co sugeruje ich istotny udział w predykcji ceny biletu.
- Zmienne *Departure_time*, *Arrival_time*, *Num_Layovers* i *layover_duration* miały relatywnie niewielki wpływ na jakość modelu, co może oznaczać, że są one mniej informatywne lub ich wpływ jest pośredni.

Analiza wskazuje, które cechy warto zachować w modelu, a które mogą być potencjalnie usunięte bez istotnej utraty jakości predykcji.

Testowanie liczby k-sąsiadów oraz wag

Wybór odpowiedniej liczby sąsiadów jest kluczowy dla skuteczności modelu K-najbliższych sąsiadów. Nie ma jeden uniwersalnej ilości k, która sprawdzi się w każdej sytuacji. Należy to sprawdzić eksperymentalnie. Na przykładzie danych lotniczych testujemy nieparzyste k, dla k ε <0; 25>.

W tym samym momencie testujemy parametr k dla różnych wartości wag:

- *Uniform* - Wszyscy sąsiedzi w obrębie k najbliższych mają **równy wpływ** na ostateczną decyzję, niezależnie od tego, jak blisko lub daleko znajdują się od klasyfikowanego punktu.
- *Distance* - Bliżsi sąsiedzi mają **większy wpływ** na ostateczną decyzję niż sąsiedzi znajdujący się dalej. Waga każdego sąsiada jest odwrotnie proporcjonalna do jego odległości od klasyfikowanego punktu.

Wyniki przedstawia poniższa tabela oraz wykres:

- Wyniki dla *Distance*

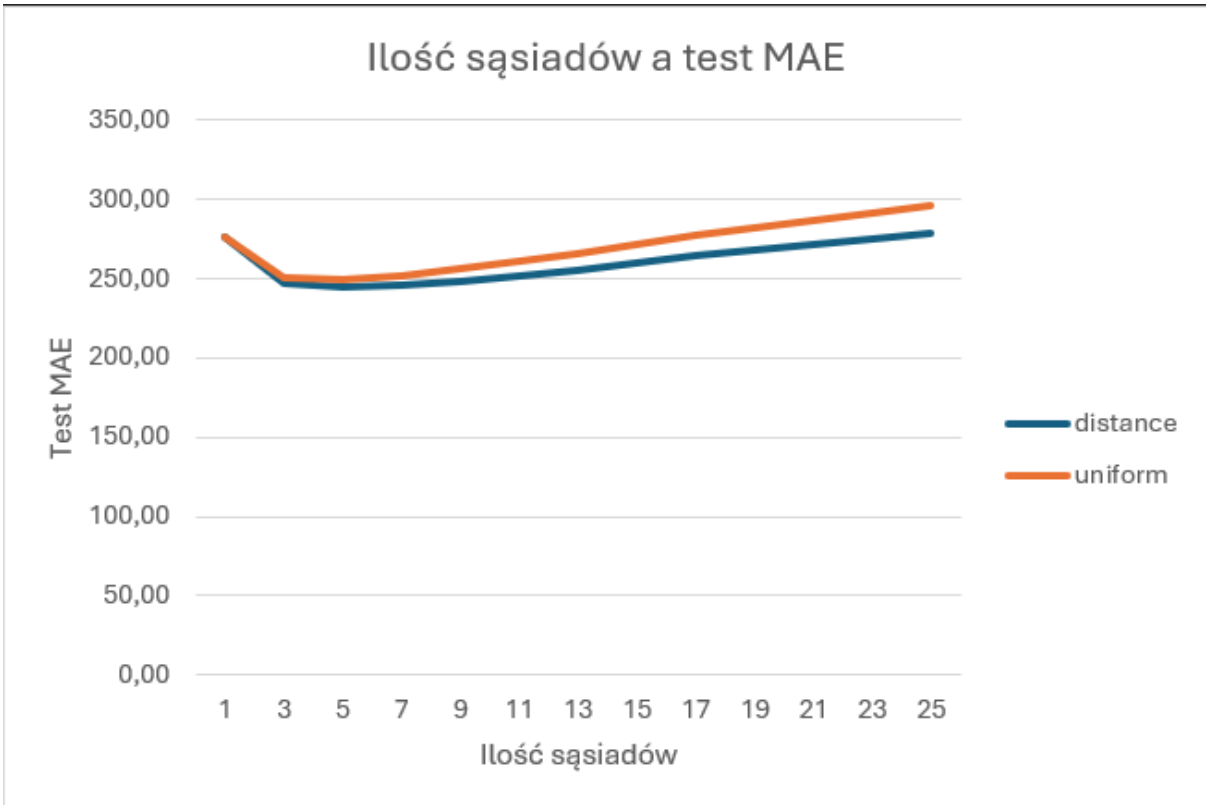
Liczba sąsiadów	weights	train_MSE	test_MSE	test_MAE	test_MAPE	test_R2
1	distance	4363,59	272625,62	276,15	31,23	0,74
3	distance	2183,99	193955,98	247,04	28,09	0,82
5	distance	2183,99	189606,17	244,57	27,63	0,82
7	distance	2183,99	193545,09	245,42	27,54	0,82
9	distance	2183,99	200801,58	248,19	27,64	0,81
11	distance	2183,99	208513,03	251,70	27,83	0,80
13	distance	2183,99	216953,55	255,42	28,05	0,80
15	distance	2183,99	227069,96	259,80	28,33	0,79
17	distance	2183,99	236250,71	264,07	28,58	0,78
19	distance	2183,99	244762,83	268,04	28,88	0,77
21	distance	2183,99	252633,64	271,75	29,18	0,76
23	distance	2183,99	260734,87	275,47	29,47	0,75
25	distance	2183,99	268960,67	278,88	29,74	0,75

- Wyniki dla *Uniform*:

Liczba sąsiadów	weights	train_MSE	test_MSE	test_MAE	test_MAPE	test_R2
1	uniform	4363,59	272625,62	276,15	31,23	0,74
3	uniform	87882,89	196058,86	250,57	28,63	0,82
5	uniform	116463,94	193826,55	249,69	28,35	0,82
7	uniform	138468,06	200419,33	252,08	28,46	0,81
9	uniform	156924,06	210363,90	256,29	28,69	0,80
11	uniform	173021,29	220334,15	261,18	29,02	0,79
13	uniform	186861,29	231024,69	266,07	29,33	0,78
15	uniform	200833,34	244085,85	271,84	29,74	0,77
17	uniform	212949,38	255672,66	277,30	30,09	0,76
19	uniform	224091,13	266425,59	282,31	30,50	0,75

21	uniform	235202,12	276283,09	286,98	30,90	0,74
23	uniform	247011,59	286555,72	291,63	31,32	0,73
25	uniform	258878,35	296831,61	295,87	31,69	0,72

- Wykres prezentujący wyniki



Analiza Wyników:

Zarówno w przypadku wag ‘distance’ oraz ‘uniform’ zaobserwować możemy początkowy spadek wartości MAE, które następnie wzrasta. Świadczy to o przeuczenia się modelu. W tabeli zaobserwować można również to zjawisko dla R^2 – początkowo wzrasta do poziomu >0.80, a później spada.

Wnioski:

Podsumowując, najlepszą kombinacją tych dwóch parametrów jest waga ‘distance’ oraz liczba sąsiadów – 5. Wtedy funkcja na wykresie przyjmuje minimum. To minimum jednak jest nie wiele niższe niż dla wag ‘distance’.

Wybór najlepszego modelu

Najlepsze wyniki predykcyjne uzyskano dla modelu KNN z parametrami **k = 5** oraz **waga = distance**, który uwzględnia większy wpływ bliższych sąsiadów. W modelu zastosowano cztery najbardziej informatywne zmienne: **Checked_bag**, **Days_to_departure**, **Cabin_bag** oraz **Flight_time**, których usunięcie znacząco pogarszało jakość prognoz. Z kolei zmienne **Departure_time**, **Arrival_time**, **Num_Layovers** i **layover_duration** miały marginalny

wpływ na wyniki i mogą być pominięte. Taka konfiguracja cech i parametrów zapewnia **najlepszy kompromis między trafnością predykcji a prostotą modelu**.

Klasyfikacja

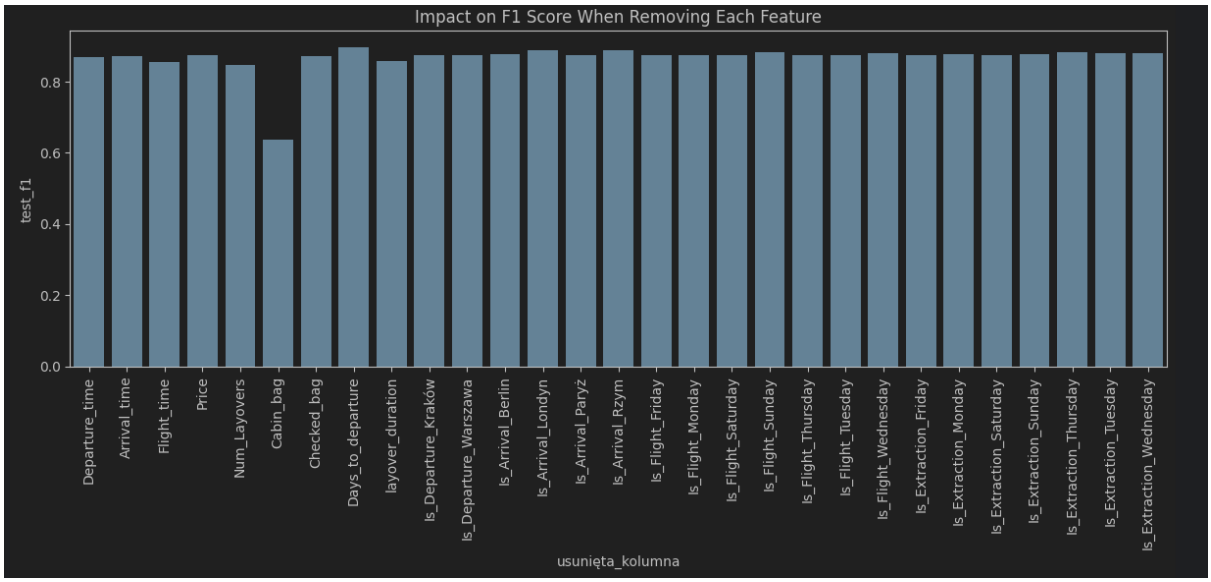
Testowanie parametrów

Dobór najlepszych zmiennych

Analogicznie jak do regresji, sprawdźmy jak zachowa się model jeśli usuniemy jedną ze zmiennych:

usunięta_kolumna	test_accuracy	test_precision	test_recall	test_f1
Departure_time	0,936	0,899441341	0,842931937	0,87027
Arrival_time	0,936	0,890710383	0,853403141	0,871658
Flight_time	0,929333333	0,887640449	0,827225131	0,856369
Price	0,938666667	0,905027933	0,848167539	0,875676
Num_Layovers	0,926666667	0,909638554	0,790575916	0,845938
Cabin_bag	0,836	0,72972973	0,565445026	0,637168
Checked_bag	0,937333333	0,913793103	0,832460733	0,871233
Days_to_departure	0,949333333	0,922651934	0,87434555	0,897849
layover_duration	0,930666667	0,892655367	0,827225131	0,858696
Is_Departure_Kraków	0,937333333	0,9	0,848167539	0,873315
Is_Departure_Warszawa	0,937333333	0,9	0,848167539	0,873315
Is_Arrival_Berlin	0,94	0,91011236	0,848167539	0,878049
Is_Arrival_Londyn	0,945333333	0,926136364	0,853403141	0,888283
Is_Arrival_Paryż	0,938666667	0,90960452	0,842931937	0,875
Is_Arrival_Rzym	0,944	0,91160221	0,863874346	0,887097
Is_Flight_Friday	0,938666667	0,905027933	0,848167539	0,875676
Is_Flight_Monday	0,938666667	0,905027933	0,848167539	0,875676
Is_Flight_Saturday	0,938666667	0,914285714	0,837696335	0,874317
Is_Flight_Sunday	0,942666667	0,915730337	0,853403141	0,883469
Is_Flight_Thursday	0,938666667	0,905027933	0,848167539	0,875676
Is_Flight_Tuesday	0,938666667	0,90960452	0,842931937	0,875
Is_Flight_Wednesday	0,941333333	0,915254237	0,848167539	0,880435
Is_Extraction_Friday	0,938666667	0,905027933	0,848167539	0,875676
Is_Extraction_Monday	0,94	0,914772727	0,842931937	0,877384
Is_Extraction_Saturday	0,938666667	0,90960452	0,842931937	0,875
Is_Extraction_Sunday	0,94	0,905555556	0,853403141	0,878706
Is_Extraction_Thursday	0,942666667	0,915730337	0,853403141	0,883469
Is_Extraction_Tuesday	0,941333333	0,910614525	0,853403141	0,881081
Is_Extraction_Wednesday	0,941333333	0,915254237	0,848167539	0,880435

Wizualizacja:



Interpretacja wyników:

- Usunięcie kolumny *Cabin_bag* prowadzi do największego spadku skuteczności modelu – *test_f1* spada do 0,637 - ta cecha ma istotne znaczenie dla klasyfikacji.
- Usunięcie cech *czasowych* takich jak *Departure_time*, *Arrival_time* czy *Flight_time* powoduje umiarkowany spadek metryk, czyli są one pomocne, ale nie kluczowe.
- Cechy związane z *datą wylotu* (*Days_to_departure*) oraz niektóre *kody miast* (*Is_Arrival_Londyn*, *Is_Arrival_Rzym*) wykazują najmniejszy wpływ na model.
- Kolumny związane z *dniem tygodnia* (*Is_Flight_*, *Is_Extraction_*) mają podobny, umiarkowany wpływ – *f1* pozostaje w zakresie 0,875–0,88.
- Usunięcie *Checked_bag* oraz *Price* wpływa tylko nieznacznie na jakość klasyfikacji, co oznacza, że ich informacyjność jest niższa niż np. *Days_to_departure*.

Testowanie liczby k-sąsiadów oraz wag

Analogicznie jak do regresji, sprawdźmy jak zachowa się model przy zmianie liczby sąsiadów oraz poszczególnych wag (*distance/uniform)*:

K	weights	accuracy	precision	recall	f1_score
1	uniform	0,8822	0,7702	0,7106	0,7391
1	distance	0,8822	0,7702	0,7106	0,7391
3	uniform	0,8698	0,7800	0,6213	0,6915
3	distance	0,8814	0,7969	0,6647	0,7247
5	uniform	0,8714	0,8362	0,5634	0,6731
5	distance	0,8854	0,8413	0,6315	0,7214
7	uniform	0,8742	0,8665	0,5506	0,6726
7	distance	0,8894	0,8714	0,6213	0,7251

Interpretacja wyników:

- W miarę zwiększania liczby sąsiadów model staje się ostrożniejszy w klasyfikowaniu przypadków jako „low-cost", rzadziej popełniając błędy fałszywie pozytywne. Jednocześnie jednak czułość maleje, co wskazuje, że dobór *K* powinien być ściśle zależny od celu analizy: czy ważniejsze jest wykrycie wszystkich *low-cost*, czy unikanie fałszywych alarmów.
- Dla każdego testowanego *K*, wersja z wagami opartymi na odwrotności odległości (*weights='distance'*) daje lepsze wyniki od wersji z równymi wagami (*uniform*). Widać to szczególnie w metryce *f1_score.*
- Pomimo dobrej precyzji, wszystkie modele mają czułość poniżej 0,72, co oznacza, że istotna część pozytywnych przypadków może być pomijana.

Najlepszy ogólny wynik (*f1_score* = **0,7251**) uzyskano dla:

- *K* = **7,** *distance*

Klasyfikator najlepiej radzi sobie z większą liczbą sąsiadów i uwzględnieniem odległości w głosowaniu.

Wybór najlepszego modelu

Model z parametrami: K=7, weights=*distance* uzyskał najlepsze ogólne wyniki spośród wszystkich testowanych konfiguracji. To dla niego osiągnęliśmy najlepszy odsetek poprawnych klasyfikacji (*accuracy)* oraz precyzji. Zbalansowana miara F1-score również ma najwyższą wartość. Ten model zapewnia zarówno stabilność, jak i równowagę.

XGBoost – wzmocnienie gradientowe

Opis modelu

Wzmocnienie gradientowe to popularne i bardzo skuteczne podejście do rozwiązywania problemów zarówno regresji, jak i klasyfikacji. Podobnie jak Random Forest, XGBoost to metoda uczenia zespołowego, a więc metoda polegająca na łączeniu ze sobą przewidywań wielu słabszych modeli w ostateczną predykcję. Pomiędzy RF a algorytmem XGBoost istnieje jednak zasadnicza różnica. W lesie losowym drzewa trenowane są niezależnie od siebie, a w algorytmie boostingowym trenujemy je sekwencyjnie. Oznacza to, że pierwsze drzewo dokonuje wstępnej predykcji, a każde następne w kolejności jest trenowane na błędach (odchyłkach od prawdziwej wartości zmiennej celu) poprzedniego drzewa. Ten proces możemy opisać następująco:

1. Dopasowujemy drzewo decyzyjne do danych: $F_1(x) = y$
2. Dopasowujemy następne drzewo do reszt poprzedniego: $h_1(x) = y - F_1(x)$
3. Ulepszamy predykcję $F_1(x)$ o wartości przewidywanych reszt $h_1(x)$: $F_2(x) = F_1(x) + h_1(x)$. Zmieniamy wartości, które kolejne modele będą przewidywać $y := F_2(x)$.
4. Powtarzamy kroki 1-3 do momentu braku poprawy metryk jakości predykcji.

Hiperparametry, które możemy rozważać w ramach testowania, są analogiczne do tych występujących w modelu Random Forest. Pierwszym jest zatem liczba drzew, którą oznaczamy jako n_estimators. Sekwencyjna budowa modelu wzmocnienia gradientowego powoduje, że, inaczej niż w przypadku modelu RF, model jest odporny na przeuczenie wraz ze wzrostem liczby drzew. Należy jednak pamiętać, że każde drzewo wydłuży czas trenowania modelu. Kolejnym parametrem jest szybkość uczenia (learning rate). Przy zmiennej liczbie drzew mała wartość (np. 0,01) zwiększa prawdopodobieństwo, że trafimy w minimum funkcji straty. Jednocześnie, przy ustalonej liczbie drzew, mniejsza szybkość uczenia oznacza, że możemy w ogóle nie trafić we wspomniane minimum. Istotnym parametrem jest również max_depth, opisujący maksymalną głębokość pojedynczego drzewa decyzyjnego w modelu. Co do zasady preferowany jest model z większą liczbą płytkich drzew w porównaniu do modelu z mniejszą liczbą drzew mocno rozgałęzionych [8]. W związku z tym, rekomendowane w [8] wartości tego parametru to 3-8. Ponadto, model XGBoost posiada również parametry odpowiadające za regularyzację: odpowiednio gamma i lambda.

Regresja

Testowanie parametrów

Testowanie przeprowadzamy, zmieniając jednocześnie tylko jeden z parametrów. Bazowe parametry modelu są następujące:

n_estimators	learning_rate	max_depth	lambda	gamma	subsample	colsample_bytree
10	0,1	5	1	0,1	1	1

Bazowy model osiągnął następujące wyniki:

train_MSE	train_MAPE	train_R2	val_MSE	val_MAPE	val_R2	test_MSE	test_MAE	test_MAPE	test_R2
837001,4	0,55	0,21	854915,6	0,55	0,22	838500,1	617,43	0,55	0,21

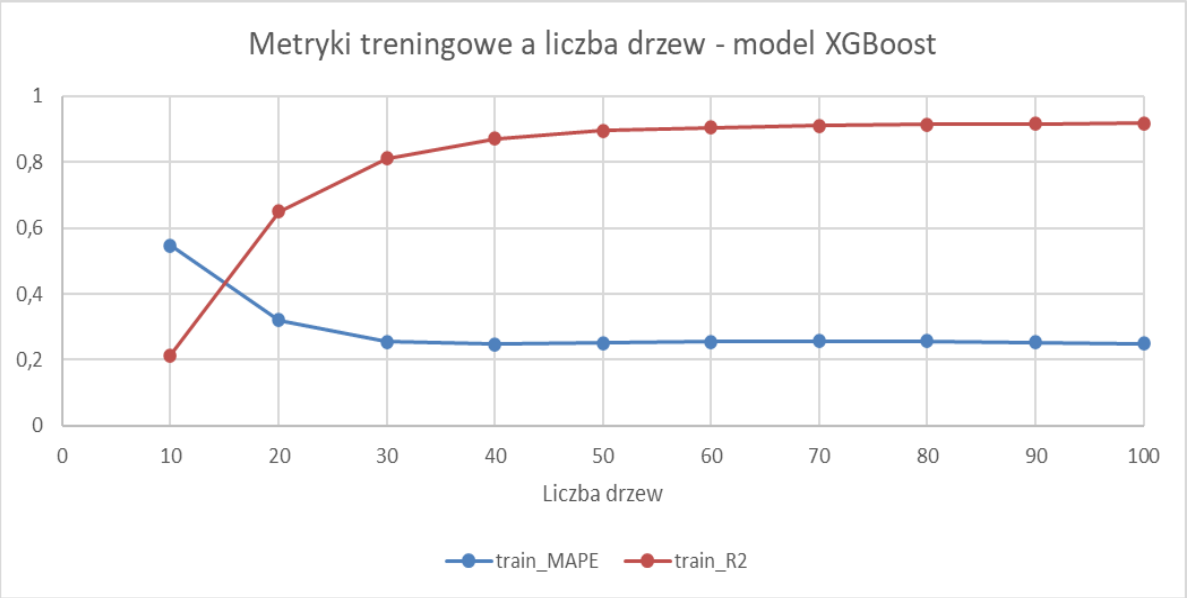
Widzimy zatem, że przy testowym R kwadrat na poziomie zaledwie 21% oraz testowym procentowym błędzie większym od 50%, występuje duże pole do poprawy. Testować będziemy liczbę drzew, maksymalną głębokość drzewa, szybkość uczenia oraz parametry regularyzacyjne gamma i lambda.

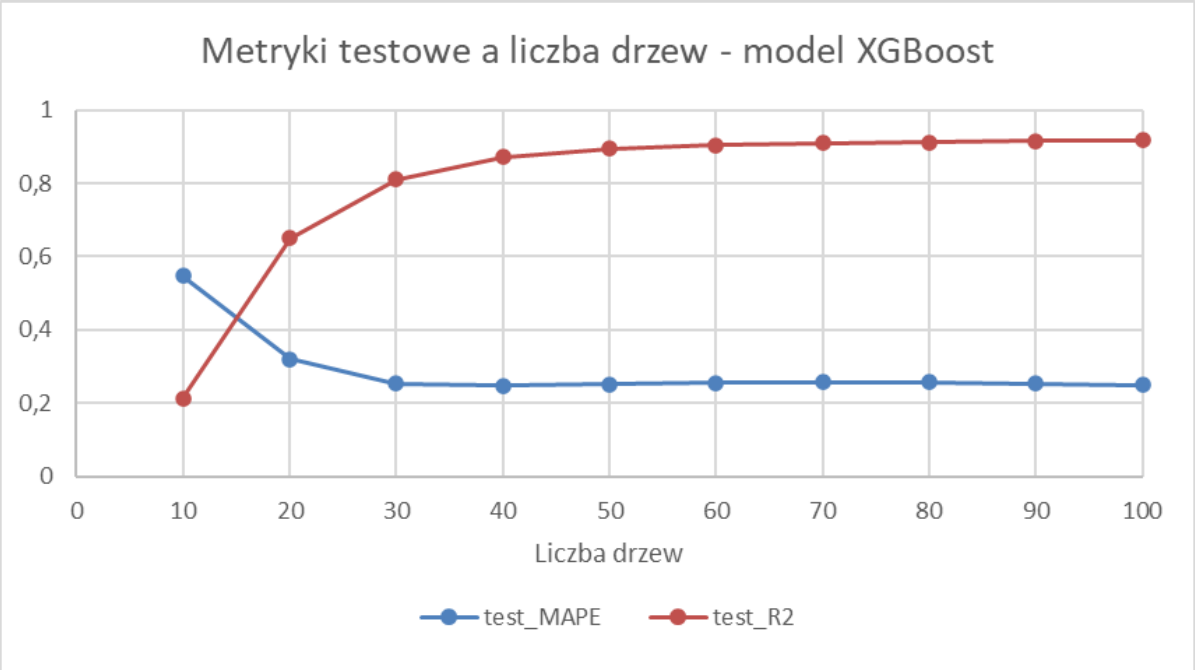
Liczba drzew

Podstawowym parametrem, który możemy testować, jest liczba drzew. Każde z nich poprawia predykcję poprzedniego. Wyniki ilustrują poniższe tabela i wykres.

n_estimators	train_MAPE	train_R2	val_MSE	val_MAPE	val_R2	test_MSE	test_MAE	test_MAPE	test_R2
10	0,55	0,21	854915,58	0,55	0,22	838500,06	617,43	0,55	0,21
20	0,32	0,65	380020,99	0,32	0,65	371727,37	382,93	0,32	0,65
30	0,25	0,81	206241,22	0,26	0,81	201321,46	275,10	0,25	0,81
40	0,25	0,87	141104,43	0,25	0,87	137657,35	232,21	0,25	0,87
50	0,25	0,90	114586,76	0,25	0,90	111777,58	214,52	0,25	0,90
60	0,26	0,91	103578,15	0,26	0,91	101218,81	207,55	0,26	0,91
70	0,26	0,91	98230,79	0,26	0,91	96181,58	204,24	0,26	0,91
80	0,26	0,91	94962,91	0,26	0,91	93066,17	202,33	0,26	0,91
90	0,25	0,92	92419,90	0,26	0,92	90443,45	199,88	0,25	0,92
100	0,25	0,92	90505,96	0,25	0,92	88447,27	197,44	0,25	0,92

To, jak liczba drzew wpływa na jakość predykcji modelu, zilustrują także wykresy.



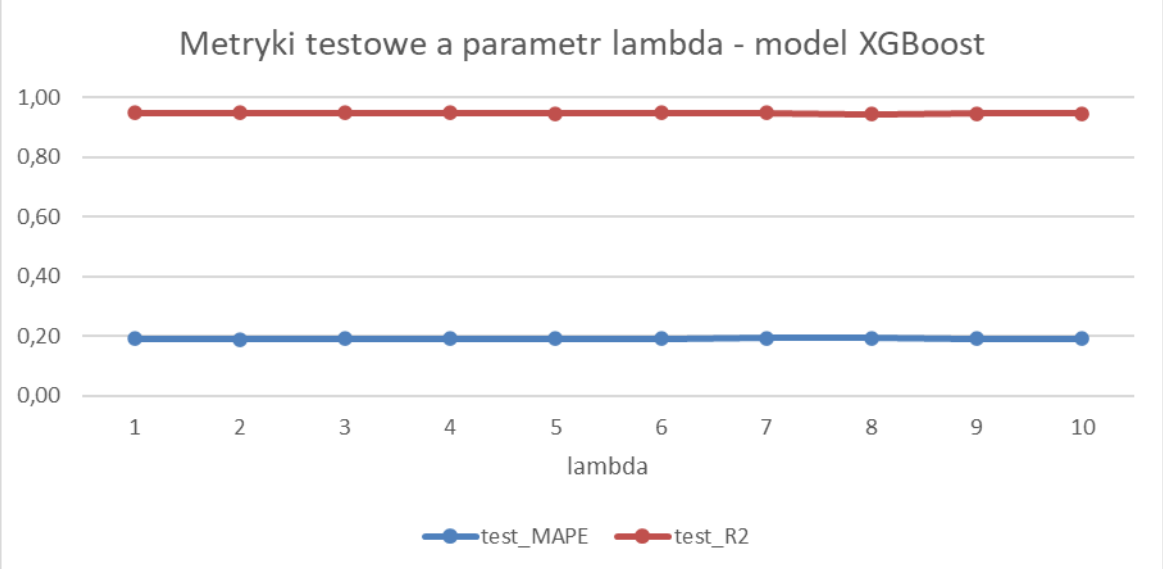


Widzimy, przy niewielkiej liczbie drzew – mniejszej niż 30 – obserwowana jest zdecydowana poprawa predykcji. Współczynnik determinacji wzrasta, a z kolei maleje średni bezwzględny błąd procentowy.

Parametr lambda

Lambda to parametr odpowiadający za minimalny spadek funkcji straty konieczny do tego, by doszło do podziału węzła drzewa decyzyjnego na dwa mniejsze węzły. Wyniki w zależności od jego wartości przedstawiamy w tabeli:

lambda	train_MSE	train_MAE	train_MAPE	train_R2	val_MSE	val_MAE	val_MAPE	val_R2	test_MSE	test_MAE	test_MAPE	test_R2
1,00	41599,50	137,47	0,18	0,96	62874,50	157,07	0,19	0,94	57251,54	153,70	0,19	0,95
2,00	42327,83	138,09	0,18	0,96	62485,45	156,77	0,19	0,94	57853,68	153,75	0,19	0,95
3,00	43097,89	139,47	0,18	0,96	62132,73	157,49	0,19	0,94	57619,04	154,24	0,19	0,95
4,00	43847,00	140,34	0,18	0,96	62890,85	158,58	0,19	0,94	58336,77	155,16	0,19	0,95
5,00	44578,07	141,36	0,18	0,96	63271,04	158,92	0,19	0,94	58524,06	155,38	0,19	0,95
6,00	45349,62	142,70	0,18	0,96	63854,03	159,62	0,19	0,94	58328,41	155,85	0,19	0,95
7,00	45558,06	143,05	0,18	0,96	63713,22	159,65	0,19	0,94	58455,10	155,92	0,19	0,95
8,00	45923,80	143,37	0,18	0,96	64092,95	160,20	0,19	0,94	59279,29	156,70	0,19	0,95
9,00	46661,09	143,89	0,18	0,96	64133,65	159,89	0,19	0,94	59041,66	156,36	0,19	0,95
10,00	46673,94	143,92	0,18	0,96	64612,10	160,04	0,19	0,94	58977,19	156,18	0,19	0,95
20,00	51471,19	150,91	0,19	0,95	62429,66	161,05	0,20	0,94	64781,53	162,17	0,20	0,94
50,00	57686,85	158,09	0,19	0,95	67300,89	166,60	0,20	0,94	69530,42	167,51	0,20	0,93
100,00	63992,63	164,46	0,20	0,94	70995,98	170,45	0,20	0,93	74254,30	171,68	0,20	0,93



Parametr lambda nie wydaje się mieć dużego wpływu na predykcje dokonywane przez nasz model. Zarówno R kwadrat, jak i MAPE pozostają stabilne.

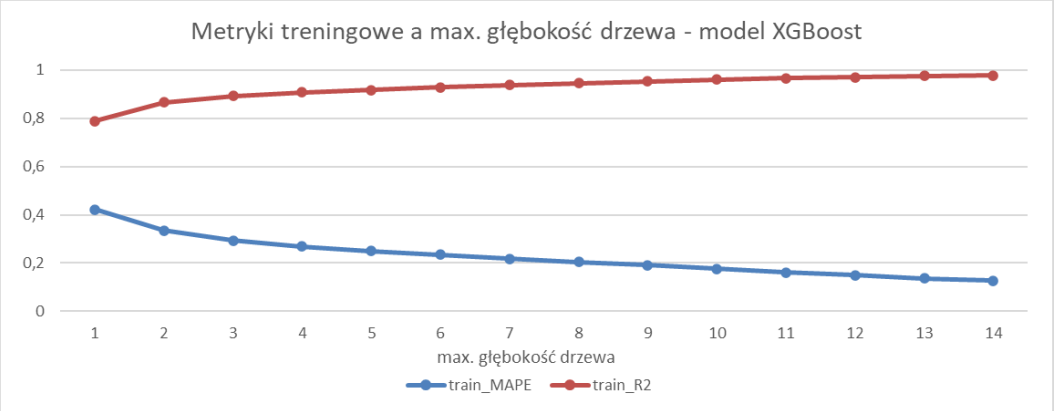
Parametr gamma

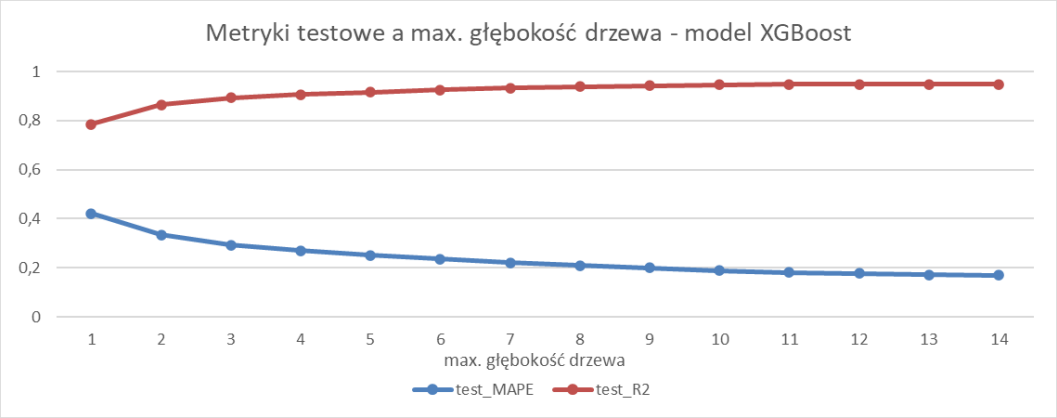
gamma	train_MSE	train_MAE	train_MAPE	train_R2	val_MSE	val_MAE	val_MAPE	val_R2	test_MSE
0,1	57686,85	158,09	0,19	0,95	67292,30	166,59	0,20	0,94	69530,61
10	57686,86	158,09	0,19	0,95	67316,20	166,61	0,20	0,94	69534,49
20	57686,86	158,09	0,19	0,95	67299,28	166,61	0,20	0,94	69538,95
50	57686,86	158,09	0,19	0,95	67286,45	166,60	0,20	0,94	69524,71
100	57310,67	157,32	0,19	0,95	66907,50	165,86	0,20	0,94	69150,66

Również wartości parametru gamma nie wydają się mieć wpływu na jakość predykcji. Przy ustalonych innych parametrach wartości jego błędów: bezwzględnego i procentowego pozostają zbliżone.

Głębokość drzewa

max_depth	train_MSE	train_MAPE	train_R2	val_MSE	val_MAPE	val_R2	test_MSE	test_MAPE	test_R2
1	224560,97	0,42	0,79	233046,68	0,42	0,79	228903,59	0,42	0,79
2	142287,37	0,34	0,87	146172,65	0,34	0,87	144492,66	0,34	0,86
3	113109,40	0,29	0,89	114798,84	0,30	0,89	114295,24	0,29	0,89
4	98783,51	0,27	0,91	101432,10	0,27	0,91	100088,06	0,27	0,91
5	87035,91	0,25	0,92	90533,56	0,25	0,92	88716,00	0,25	0,92
6	76056,66	0,23	0,93	80545,69	0,24	0,93	78914,45	0,24	0,93
7	66126,94	0,22	0,94	73357,46	0,22	0,93	70850,74	0,22	0,93
8	57262,94	0,20	0,95	67112,99	0,21	0,94	65148,87	0,21	0,94
9	49395,25	0,19	0,95	62793,16	0,20	0,94	60428,50	0,20	0,94
10	41717,78	0,18	0,96	59576,03	0,19	0,95	57361,53	0,19	0,95
11	35327,63	0,16	0,97	57562,69	0,18	0,95	55361,89	0,18	0,95
12	31026,36	0,15	0,97	57720,36	0,18	0,95	54724,64	0,18	0,95
13	26422,56	0,14	0,98	57868,56	0,17	0,95	54796,00	0,17	0,95
14	23046,72	0,13	0,98	58411,32	0,17	0,95	54828,01	0,17	0,95





Z wykresów i tabeli wynika, że model wzmocnienia gradientowego składający się z głębszych drzew radzi sobie lepiej – myli się mniej zarówno na zbiorze treningowym, jak i testowym. Jednak mając na uwadze fakt, że wytrenowanie głębszego drzewa jest czasochłonne, a drzewa o głębokości > 10 mają mniej więcej zbliżone metryki, przyjmijmy 10 za optymalną wartość tego parametru max_depth.

Szybkość uczenia

Learning rate określa to, jak szybko model schodzi w dół gradientu funkcji straty. Wyniki w zależności od tego parametru prezentuje tabela:

learning_rate	train_MSE	train_MAPE	train_R2	val_MSE	val_MAPE	val_R2	test_MSE	test_MAPE	test_R2
0,01	1763615,32	0,90	-0,64	1765184,35	0,90	-0,65	1746053,94	0,90	-0,65
0,05	819115,40	0,56	0,24	818438,52	0,56	0,24	816107,03	0,56	0,23
0,1	325730,56	0,31	0,70	327319,42	0,32	0,69	331802,05	0,32	0,69
0,2	84640,50	0,19	0,92	92690,97	0,19	0,91	97728,07	0,19	0,91

Tutaj obserwujemy ciekawe zjawisko. Widzimy bowiem, że model popełnia mniejszy błąd wraz ze wzrostem, a nie spadkiem szybkości uczenia. Można to wyjaśnić tym, że dla domyślnych 50 drzew i niewielkiej wartości learning_rate model nie dochodzi do minimum funkcji straty. Powtórzmy więc testowanie dla 100 drzew:

learning_rate	train_MSE	train_MAE	train_MAPE	train_R2	val_MSE	val_MAE	val_MAPE	val_R2	test_MSE	test_MAE	test_MAPE	test_R2
0,01	834553,08	624,87	0,57	0,22	833915,33	625,53	0,57	0,22	831283,45	623,28	0,57	0,21
0,05	68783,23	167,91	0,18	0,94	78719,54	175,64	0,19	0,93	83197,59	176,56	0,19	0,92
0,1	41916,47	137,87	0,18	0,96	59152,21	154,40	0,19	0,94	60543,55	154,79	0,19	0,94
0,2	32074,77	118,41	0,15	0,97	55728,91	143,65	0,17	0,95	56427,23	143,97	0,17	0,95

W dalszym ciągu następuje poprawa metryk wraz ze wzrostem szybkości uczenia.

Wybór najlepszego modelu

Na podstawie powyższych analiz, stwierdzamy, że najlepszy model posiada następujące parametry:

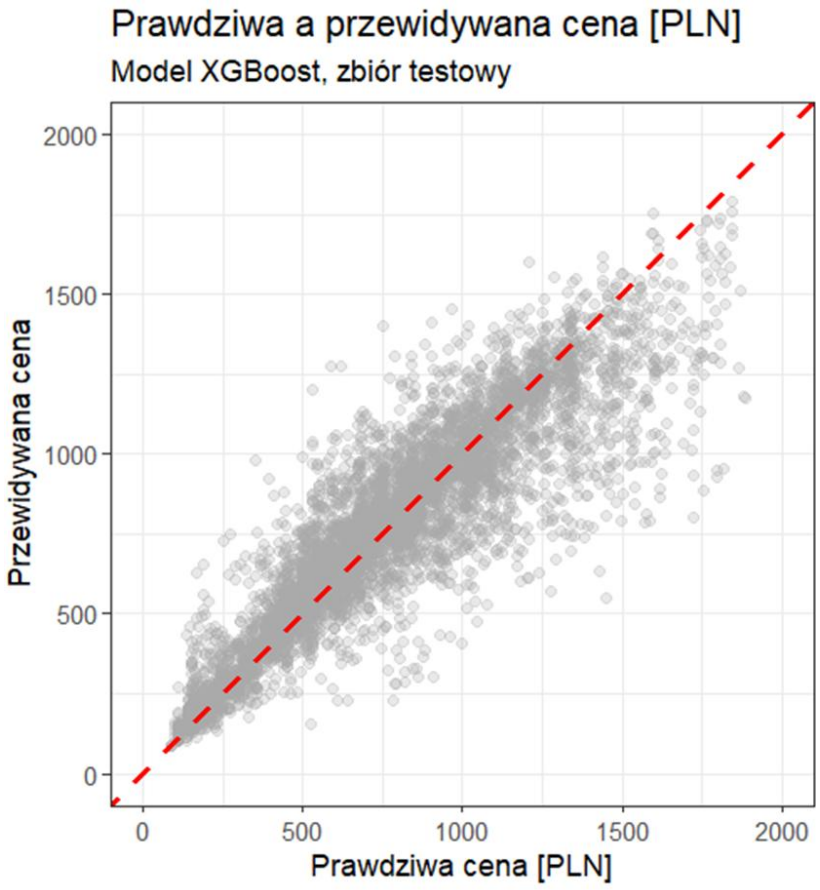
n_estimators	learning_rate	max_depth	lambda	gamma	subsample	colsample_bytree
100	0,1	14	1	0,1	1	1

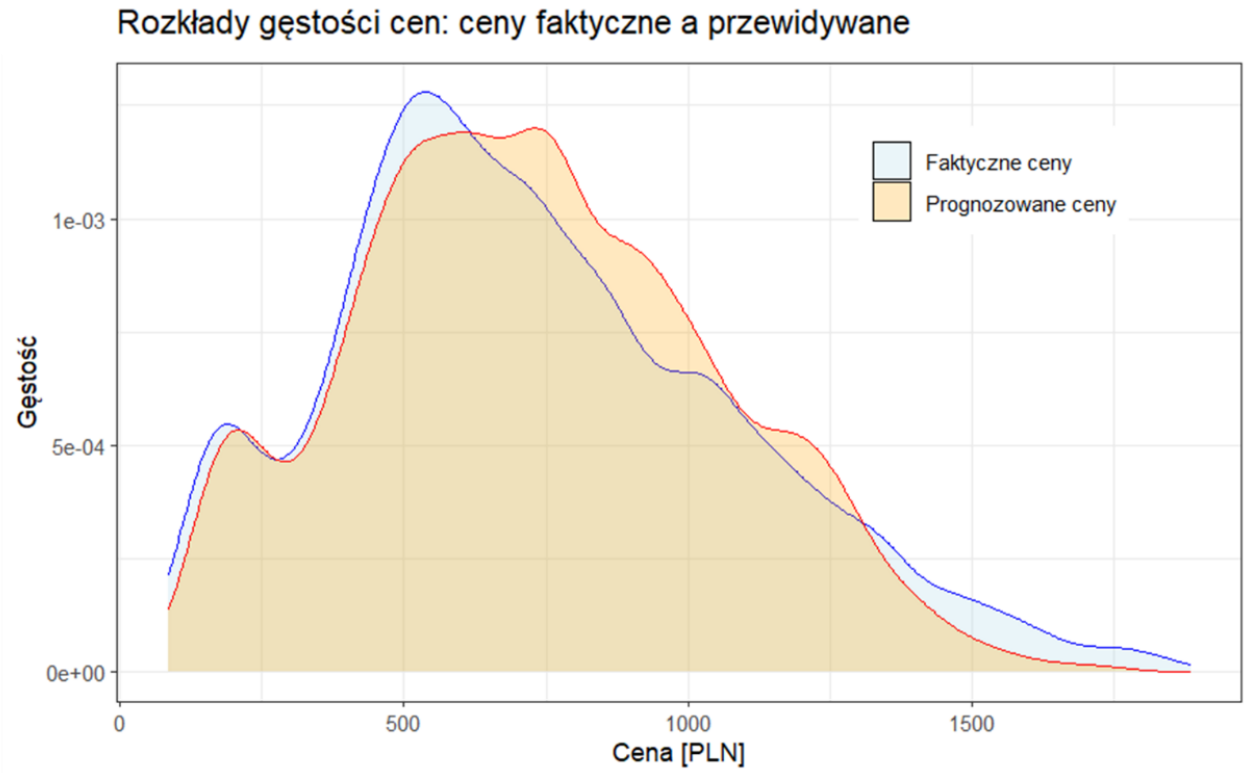
Metryki dla ostatecznego modelu wyglądają natomiast następująco:

train_MSE	train_MAPE	train_R2	val_MSE	val_MAPE	val_R2	test_MSE	test_MAPE	test_R2
23046,72	**0,13**	**0,98**	**58411,32**	**0,17**	**0,95**	**54828,01**	**0,17**	**0,95**

Model myli się średnio o zaledwie 17%, wyjaśnia aż 95% wariancji ceny dla zbioru testowego - to bardzo zadowalające wyniki.

Wyniki predykcji ostatecznego modelu XGBoost zwizualizujemy na wykresach:





Klasyfikacja

Testowanie Parametrów

Celem tej sekcji było sprawdzenie, czy boosting drzew (XGBoost) poprawi wyniki względem lasu losowego i sieci neuronowej. Przetestowaliśmy pełną siatkę hiperparametrów:

- Liczba drzew **n_estimators ∈ {50, 100, 150, 200, 300}**
- Głębokość drzew **max_depth ∈ {2, 3, 4}**
- Współczynnik uczenia **learning_rate ∈ {0.30, 0.20, 0.15, 0.10}**
- Podpróbkowanie wierszy subsample / colsample_bytree ∈ {0.6, 0.7, 0.8}

n_estimators	max_depth	learning_rate	train_acc	val_acc	test_acc	test_f1	log_loss
50	3	0.2	0.8	0.72	0.71	0.63	1.45
100	3	0.2	0.82	0.74	0.73	0.66	1.35
150	3	0.2	0.83	0.74	0.74	0.67	1.3
200	3	0.2	0.84	0.75	0.75	0.68	1.25
300	3	0.2	0.86	0.77	0.76	0.69	1.2

max_depth	n_estimators	learning_rate	train_acc	val_acc	test_acc	test_f1	log_loss
2	200	0.15	0.82	0.74	0.73	0.66	1.35
3	200	0.15	0.84	0.75	0.74	0.67	1.3
4	200	0.15	0.85	0.76	0.75	0.68	1.25
5	200	0.15	0.84	0.75	0.74	0.67	1.28

learning_rate	n_estimators	max_depth	train_acc	val_acc	test_acc	test_f1
0.3	200	3	0.8	0.72	0.71	0.63
0.2	200	3	0.82	0.74	0.73	0.66
0.15	200	3	0.83	0.75	0.74	0.67
0.1	200	3	0.85	0.76	0.75	0.68

subsample	n_estimators	max_depth	learning_rate	train_acc	val_acc	test_acc	test_f1
0.6	200	3	0.15	0.81	0.73	0.72	0.65
0.7	200	3	0.15	0.82	0.74	0.73	0.66
0.8	200	3	0.15	0.83	0.75	0.74	0.67
0.9	200	3	0.15	0.83	0.75	0.74	0.67

Wnioski

- Boosting nie dorównał pozostałym algorytmom: najlepsza konfiguracja (300 drzew, depth = 4, η = 0.10) zatrzymała się na *accuracy = 0.76* i *F1 = 0.69*, podczas gdy Random Forest osiągał ~0.94, a najlepsza sieć NN ~0.81.
- Wyższy współczynnik uczenia (η ≥ 0.20) i płytsze drzewa (depth ≤ 2) skutkowały wyraźnym niedouczeniem – log-loss przekraczał 1.3, a dokładność spadała do ~0.71.
- Zwiększanie liczby drzew powyżej 300 nie poprawiało metryk (próg nasycenia), za to istotnie wydłużało czas trenowania.

Bibliografia

1. K. Tziridis, Th. Kalampokas i in., Airfare Prices Prediction Using Machine Learning Techniques, 2017 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, s.1036-1039.
2. K. Tziridis, Th. Kalampokas i in., A Holistic Approach on Airfare Price Prediction Using Machine Learning Techniques, "IEEE Access" 2023, Nr 11, s. 46627-46643.
3. A. Aliberti, Y. Xin i in., "Comparative analysis of neural networks techniques to forecast Airfare Prices," 2023 IEEE 47th Annual Computers, Software and Applications Conference (COMPSAC), Turyn, 2023, s. 1023-1029.
4. A. Meepaganithage i in., "Airfare Forecasting: A Deep Learning Approach to Flight Price Prediction," *2024 Fourth International Conference on Digital Data Processing (DDP)*, New York, NY, USA, 2024, s. 5-10.
5. Probst P. i in., 2019: Tunability: Importance of Hyperparameters of Machine Learning Algorithms, Journal of Machine Learning Research, 20, s. 1-32.
6. Gupta S., Gupta N., 2024: Flight Fare Prediction Using Machine Learning, The Journal of Computational Science and Engineering, 2, s. 11-26.
7. Boehmke B., Greenwell B., 2019: Hands-On Machine Learning with R. New York: Chapman & Hall/CRC.
8. Press Hastie T., i in., 2009: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. New York: Springer.
9. Molnar Ch., 2022: Interpretable Machine Learning, https://christophm.github.io/interpretable-ml-book/feature-importance.html [dostęp: 08.12.2024]
10. https://www.momondo.pl [dostęp: 08.12.2024]