DEBIUTY STUDENCKIE

2023

INFORMATYKA W BIZNESIE

pod redakcją
Heleny Dudycz

Wydawnictwo Uniwersytetu Ekonomicznego we Wrocławiu
Wrocław 2023

Recenzja naukowa
Marcin Hernes

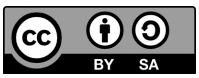
Redakcja wydawnicza
Małgorzata Tadrzak-Mazurek

Korekta
Aleksandra Śliwka

Skład i łamanie
Małgorzata Myszkowska

Projekt okładki
Beata Dębska

Na okładce wykorzystano zdjęcie z zasobów Adobe Stock



ISBN 978-83-67400-80-0 (wersja papierowa)
ISBN 978-83-67400-81-7 (wersja elektroniczna)

DOI: 10.15611/2023.81.7

Druk i oprawa: TOTEM

Mateusz Frącz

e-mail: mateuszf.poczta@gmail.com

ORCID: 0009-0002-1245-8409

Uniwersytet Ekonomiczny we Wrocławiu

Prognoza cen nieruchomości mieszkalnych z wykorzystaniem uczenia maszynowego

DOI: 10.15611/2023.81.7.05
JEL Classification: Y80

Streszczenie: Uczenie maszynowe odgrywa w dzisiejszym świecie coraz większą rolę. Ma wiele zastosowań, które mogą być wykorzystywane w nauce i przedsiębiorstwach. Pozwala na dokonywanie prognoz wartości ciągłych z wykorzystaniem estymatora przeprowadzającego regresję. Celem badania jest stworzenie modelu prognozującego ceny nieruchomości mieszkalnych i weryfikacja jego skuteczności z wykorzystaniem różnych algorytmów w języku Python 3, biblioteki Pandas, LazyPredict, Scikit-learn w środowisku Jupyter Notebook. Zastosowane metody badawcze to: analiza literatury, analiza dokumentacji, studium porównawcze wykorzystanie biblioteki LazyPredict do wyznaczenia najwydajniejszego estymatora zgodnie z wartością współczynnika determinacji, wykorzystanie GridSearchCV do znalezienia najlepszej kombinacji hiperparametrów modelu lasu losowego. Dokonana analiza umożliwiła wskazanie najlepszego algorytmu spośród wybranych, którym okazał się regresor lasu losowego, do przeprowadzenia procesu uczenia na danych.

Słowa kluczowe: prognoza cen nieruchomości mieszkalnych, uczenie maszynowe, Lazy Predict, GridSearchCV, RandomForestRegressor

1. Wstęp

Oszacowanie wartości nieruchomości mieszkalnej wymaga wiedzy dotyczącej sposobu ich wyceny. W sytuacji braku odpowiednich kwalifikacji w tej dziedzinie zalecana jest konsultacja z rzeczoznawcą majątkowym. Jednak niniejsze badanie nie skupia się na aspektach związanych z rynkiem mieszkań i sposobach, w jaki dochodzi do ustalania ich cen. Badanie nie uwzględnia tendencji występujących na rynku nieruchomości wpływających na wartość mieszkań w czasie.

Przedmiotem niniejszego badania są modele uczenia maszynowego prognozujące ceny nieruchomości mieszkalnych w wybranych polskich miastach uzyskane za pomocą różnych algorytmów. Badanie polega na przeprowadzeniu procesu trenowania modeli uczenia maszynowego na wybranych danych. Dane ograniczają się do wybranych lokalizacji na terenie Krakowa, Poznania i Warszawy. W badaniu wykorzystywane są dane, w których istnieją pewne zależności między konkretnymi cechami opisującymi daną nieruchomość a jej ceną. W związku z tym celem badania jest przeprowadzenie prognoz cen nieruchomości mieszkalnych przy uwzględnieniu ich lokalizacji (współrzędne geograficzne, ulica, miasto), numeru piętra, powierzchni, liczby pokoi i roku nieruchomości z wykorzystaniem wytrenowanych modeli uczenia

maszynowego i wybranie algorytmu umożliwiającego uzyskanie najwydajniejszego modelu. Można w związku z tym postawić następujące pytania badawcze:

- Który algorytm pozwoli uzyskać najwydajniejszy model uczenia maszynowego?
- Czy można przeprowadzić prognozę cen nieruchomości mieszkalnych z względnie akceptowalnymi wartościami błędów, z wykorzystaniem uczenia maszynowego, mimo nieznanego charakteru zależności występujących między cechami a celem?

Niniejszy artykuł jest podzielony na siedem sekcji. Pierwsza zapoznaje z przedmiotem badania, opisuje jego cel i zawiera postawione pytania badawcze. W drugiej sekcji artykułu opisano istotę przeprowadzonego badania i wykorzystane metody badawcze. Trzecia sekcja opisuje proces przygotowania danych do utworzenia modeli uczenia maszynowego z wykorzystaniem wybranych algorytmów, czwarta zawiera analizę występujących zależności między cechami a celem, kolejna opisuje proces wykorzystania biblioteki Lazy Predict do utworzenia rankingu uzyskanych modeli uczenia maszynowego, w którym kryterium porównawczym wydajności modeli jest współczynnik determinacji i pierwiastek błędu średniokwadratowego. Następna sekcja opisuje wyszukiwanie najlepszych hiperparametrów z podanej przestrzeni wyszukiwania z wykorzystaniem GridSearchCV. W ostatniej natomiast podsumowano wyniki badania i wskazano, w jaki sposób można rozszerzyć jego zakres.

2. Istota i metody badania

Wybór właściwego estymatora bez przeprowadzenia badania wydajności modeli na konkretnych danych może nie być łatwy. W związku z tym podczas badania postanowiono przeprowadzić analizę porównawczą (studium porównawcze) wartości współczynników determinacji i pierwiastków błędu średniokwadratowego różnych regresorów z wykorzystaniem wygenerowanego przez bibliotekę Lazy Predict rankingu wybranych modeli uczenia maszynowego.

Po wybraniu odpowiedniego estymatora zgodnie z wynikami działania Lazy Predict lub opierając się na intuicji, wiedzy i doświadczeniu, można zadbać również o właściwy dobór hiperparametrów danego modelu – to znaczy wyregulować ten model w celu zwiększenia jego wydajności. W związku z tym postanowiono przeprowadzić proces dokonujący automatycznego porównania wyników (studium porównawcze). W tym celu wykorzystano kolejne narzędzie o nazwie GridSearchCV. Jest to metoda przeszukiwania siatki. Wymaga ona zdefiniowania nazw hiperparametrów i przyjmowanych przez nie wartości w słowniku. Klucz stanowi nazwę danego hiperparametru, dla którego chcemy znaleźć najlepszą wartość. Wartości zostają wybrane spośród podanych przez użytkownika w postaci listy. W przypadku przeszukiwania siatkowego sprawdzane są wszystkie kombinacje wartości dla podanych hiperparametrów w celu znalezienia najlepszych kombinacji. O słuszności wyboru konkretnie tych, a nie innych hiperparametrów decyduje osoba przygotowująca model uczenia maszynowego. Należy mieć na uwadze, iż przeszukiwanie wszystkich

kombinacji wymaga stosunkowo dużej mocy obliczeniowej i może zająć dużo czasu. Jest również możliwość przekazania listy na przykład z dwoma słownikami. W takim przypadku sprawdzane są wszystkie możliwości konfiguracji hiperparametrów modelu dla każdego słownika osobno. Liczba kombinacji wówczas jest równa sumie kombinacji dla obu słowników.

Badanie dotyczy regresji wielorakiej. Regresja jest wieloraka, ponieważ istnieje wiele cech, na podstawie których dokonywana jest prognoza ceny nieruchomości mieszkalnej. Prawdopodobnie zależności mają charakter nieliniowy, co zostanie wyjaśnione w punkcie opisującym analizę zależności cech i przewidywanej wartości (celu). Zdecydowano się na wybór lasu losowego, ponieważ uzyskał on najwyższy wynik w rankingu wybranych modelów wygenerowanym przez Lazy Predict.

Pozostałymi zastosowanymi metodami badawczymi były analiza literatury i analiza dokumentacji biblioteki Pandas (*API Reference Pandas*, 2022) i Scikit-learn (*API Reference Scikit-learn*, 2023).

3. Proces przygotowania danych do badania

Poniżej zamieszczono opis poszczególnych kroków, które umożliwiły utworzenie różnych modelów uczenia maszynowego z Lazy Predict i ostatecznego modelu z wykorzystaniem regresora lasów losowych z wyszukanymi odpowiednimi hiperparametrami za pomocą GridSearchCV. Oczywiście w artykule nie zamieszczono wszystkich kroków przygotowania danych, tylko informacje uznane za najistotniejsze z punktu widzenia przeprowadzonego badania.

3.1. Opis danych

Wykorzystany zbiór danych dotyczący nieruchomości mieszkalnych pochodził z serwisu Kaggle (Cegielski, 2021). Dokładna informacja dotycząca źródła danych nie została przez autora zamieszczona. Wiadomo, iż pochodzą one z serwisu z ogłoszeniami sprzedaży mieszkań, z lutego 2021 roku. Zostały wstępnie przygotowane, mimo to wymagały odpowiedniego nakładu pracy w celu wykorzystania ich do utworzenia modelu uczenia maszynowego. Opublikowano je w pliku o rozszerzeniu .csv (ang. *comma separated values* – wartości oddzielane przecinkami). Do wczytania i przygotowania danych została użyta biblioteka Pandas, której zastosowanie podczas przypisywania danych do zmiennej *data* jest widoczne na rysunku 1.

```
data = pd.read_csv('Houses.csv', encoding='cp1250')
```

Rys. 1. Wczytanie danych z pliku .csv do ramki danych z ręcznie wybranym kodowaniem

Źródło: opracowanie własne.

Do pracy nad danymi wykorzystano specjalną strukturę nazywaną ramką danych (ang. *data frame*). Zawierała ona 11 kolumn i 23 764 wiersze. Ramka danych miała o jeden wiersz mniej, niż wynosi linii w oryginalnym pliku, ponieważ pierwszy wiersz wykorzystano jako nagłówek tabeli. Niestety, bibliotece Pandas nie udało się automatycznie wybrać właściwego kodowania znaków. Zaistniała więc konieczność przypisania do parametru *encoding* wartości w postaci tekstu *'cp1250'.*

3.2. Wstępna ocena danych

Na początku dokonano oceny wczytanych danych. W związku z obecnością kolumny *Unnamed*: 0, czyli bez nazwy, dokonano weryfikacji poprawności zawartych danych z wykorzystaniem metody head(15). Ze względu na duży rozmiar rysunku postanowiono umieścić pięć pierwszych wierszy ramki danych – jako efekt wywołania metody head bez parametru. Rezultat tego działania jest widoczny na rysunku 2.

```
data.head()
```

	Unnamed: 0	address	city	floor	id	latitude	longitude	price	rooms	sq	year
0	0	Podgórze Zabłocie Stanisława Klimeckiego	Kraków	2.00	23918.00	50.05	19.97	749000.00	3.00	74.05	2021.00
1	1	Praga-Południe Grochowska	Warszawa	3.00	17828.00	52.25	21.11	240548.00	1.00	24.38	2021.00
2	2	Krowodrza Czarnowiejska	Kraków	2.00	22784.00	50.07	19.92	427000.00	2.00	37.00	1970.00
3	3	Grunwald	Poznań	2.00	4315.00	52.40	16.88	1290000.00	5.00	166.00	1935.00
4	4	Ochota Gotowy budynek. Stan deweloperski. Osta...	Warszawa	1.00	11770.00	52.21	20.97	996000.00	5.00	105.00	2020.00

Rys. 2. Zawartość pierwszych pięciu wierszy ramki danych po wczytaniu z pliku

Źródło: opracowanie własne.

Kolumna *Unnamed: 0* prawdopodobnie jest identyfikatorem danego rekordu, więc może się okazać niepotrzebna do budowy modelu. Poza tym identyfikator automatycznie został nadany przez Pandas. To samo dotyczy kolumny *id*. Jednak ostateczną decyzję o usunięciu postanowiono podjąć po głębszej analizie danych.

Następnym krokiem było wykonanie podstawowych statystyk opisowych, które są widoczne na rysunku 3. Z macierzy wynika, iż wartości dotyczące numeru piętra mogą być prawdziwe – najwyższym piętrem może być piętro numer 10. Kolumny *Unnamed: 0* i *id* różnią się od siebie, jednak nie wydają się interesujące. Najprawdopodobniej nie wnoszą żadnych istotnych danych, które mogą być skorelowane z ceną nieruchomości, dlatego uznano je za nieistotne. Natomiast największy niepokój budzi minimalny rok (70), w którym została wybudowana lub sprzedana nieruchomość mieszkalna (informacja precyzująca, czego dotyczy kolumna „rok", nie została umieszczona na stronie internetowej z danymi). Niemożliwe jest też, żeby mieszkanie zostało wybudowane lub sprzedane w 2980 roku, biorąc pod uwagę fakt, iż dane pochodzą z 2021 roku, a badanie zostało przeprowadzone w roku 2023.

```
data.describe()
```

	Unnamed: 0	floor	id	latitude	longitude	price	rooms	sq	year
count	23764.00	23764.00	23764.00	23764.00	23764.00	23764.00	23764.00	23764.00	23764.00
mean	11881.50	2.81	15621.96	51.37	19.86	649353.65	2.62	102.72	2000.55
std	6860.22	2.46	8617.29	1.10	1.45	532696.99	1.00	6533.69	48.31
min	0.00	0.00	1.00	49.93	4.20	5000.00	1.00	8.80	70.00
25%	5940.75	1.00	8420.75	50.07	19.92	411546.12	2.00	42.00	1985.00
50%	11881.50	2.00	15637.50	52.19	20.00	520000.00	3.00	53.89	2019.00
75%	17822.25	4.00	23111.25	52.27	21.00	699999.00	3.00	68.91	2021.00
max	23763.00	10.00	30308.00	54.44	30.32	15000000.00	10.00	1007185.00	2980.00

Rys. 3. Podstawowe statystyki opisowe dla nieoczyszczonego zbioru danych

Źródło: opracowanie własne.

Pozostałe kwestie, wynikające prawdopodobnie z nieprawidłowych danych, zostały pominięte ze względu na ograniczenia formalne artykułu.

3.3. Oczyszczanie danych

Dane nieoczyszczone zawierające cechy niemające zauważalnego związku z celem należy poddać analizie, a następnie odpowiednio przygotować lub usunąć.

Działaniami, które zostały zaplanowane po obejrzeniu danych, było usunięcie kolumn *Unnamed: 0* i *id*, ponieważ wysunięto wniosek, iż prawdopodobnie nie zawierają one danych istotnych z punktu widzenia przeprowadzenia procesu uczenia maszynowego – nie są to cechy (zmienne niezależne), od których zależy cel (zmienna zależna). Jednak lokalizacja nieruchomości może mieć istotną zależność z jej ceną z punktu widzenia prognozowania ich wartości. Po obejrzeniu zawartości kolumny *address* postanowiono, że zbadania jej wpływu na wydajność modelu dokona się po przeprowadzeniu kodowania „gorącojedynkowego" (ang. *one-hot encoding*) w celu zamiany danych kategorycznych (w postaci tekstu) na wartości numeryczne, które potrafią przetwarzać algorytmy uczenia maszynowego (Géron, 2020).

3.4. Przygotowywanie danych

Pierwszą operacją przygotowującą zbiór danych do trenowania modelu było kodowanie „gorącojedynkowe". Przed jakąkolwiek modyfikacją zbioru, poza wcześniej dokonaną zamianą wartości kategorycznych na numeryczne (w celu umożliwienia pracy algorytmom na zbiorach z wartościami tekstowymi), bardzo ważne jest, aby dokonać podziału na zbiór treningowy i testowy. Operacja ta umożliwia przeprowa-

dzanie procesu weryfikacji uzyskanego modelu za pomocą wybranych metryk mierzących błędy wartości przewidywanych.

Kolejnym krokiem podjętym w procesie przygotowywania danych było rzutowanie na typ całkowitoliczbowy kolumn: *rooms*, *floor* i *year*. W zbiorze danych nie zostały wykryte wartości puste, natomiast wykryto wartości odstające, które częściowo zostały usunięte. Nie zdecydowano się na eliminację wszystkich, ponieważ po dokonaniu tego zabiegu pozostałe dane również zawierały wartości odstające. Drugi powód to obniżanie się współczynnika determinacji. W związku z ograniczonym czasem na przeprowadzenie badania pozostawiono zbiór danych w stanie, który pozwolił uzyskać względnie akceptowalne wyniki.

Lasy losowe nie wymagają skalowania cech. Ze względu na poszukiwanie najlepszego modelu z wykorzystaniem różnych estymatorów mogłaby zaistnieć konieczność dokonania skalowania wartości numerycznych, jednak operację tę wykonuje automatycznie Lazy Predict. Zauważono jednak, iż po wykorzystaniu skalowania cech podczas bezpośredniego wykorzystania Scikit-learn do trenowania wynik R^2 nieznacznie się poprawił dla lasu losowego w początkowych etapach pracy. W związku z tym postanowiono pozostawić ten krok. Do skalowania wykorzystano transformator StandardScaler, a wybór był podyktowany większą odpornością na dane odstające niż w przypadku transformatora *min-max* ograniczającego dane do określonego zakresu (Géron, 2020).

Na rysunku 4 przedstawiono liczbę pól kategorycznych w kolumnie *address* uzyskaną za pomocą metody value_counts.

```
data2['address'].value_counts()
```

```
Mokotów                                 426
Wola                                    384
Nowe Miasto Malta ul. Katowicka         377
Śródmieście                             352
Białołęka                               248
                                       ...
Mokotów Dolny Mokotów ul. Konstancińska   1
Mokotów Sadyba Bernardyńska               1
Podgórze Płaszów Wielicka                 1
Mokotów Służewiec Bełdan                  1
Bemowo ul. Antoniego Kocjana              1
Name: address, Length: 5419, dtype: int64
```

Rys. 4. Liczba wystąpień konkretnych wartości kategorycznych w wierszach dla kolumny z adresem

Źródło: opracowanie własne.

Widoczne jest zróżnicowanie między najliczniejszymi wartościami tekstowymi adresu a najmniej licznymi.

Procesu *one-hot encoding* dokonano za pomocą funkcji get_dummies. W rezultacie uzyskano ramkę danych o liczbie kolumn wynoszącej 5321.

4. Analiza zależności między cechami a celem

Eksploracyjna analiza danych wymaga odpowiedniego przygotowania danych do przeprowadzenia procesu uczenia modelu. Niepodjęcie działań związanych z analizą zbioru danych może skutkować niską wydajnością modelu – jego prognozy mogą mieć ogromne wartości błędów, co uczyni model bezużytecznym efektem pracy.

W badaniu postanowiono wykorzystać gotową metodę dostarczaną przez bibliotekę Pandas, zwracającą jako rezultat swojego działania ramkę danych, w której znajdują się współczynniki korelacji między każdą zmienną występującą w zbiorze danych. Podczas analizy zbioru danych użyto specjalnego parametru, do którego podano wartość tekstową sprawiającą, iż współczynniki korelacji zostały obliczone za pomocą metody rho Spearmana. Dokonano tego ze względu na fakt, iż po pierwsze, nie był znany charakter zbioru danych (czy istnieją w nim między danymi zależności liniowe – może są one nieliniowe), po drugie, nie było wiadomo podczas pierwszych etapów prac, czy dane zawierają wartości odstające. W przypadku regresji liniowej każda wartość, która jest uznawana za odstającą, sprawia, iż współczynniki utworzonego modelu wpływają negatywnie na wyniki prognoz. Model regresji wielorakiej liniowej jest płaszczyzną w przestrzeni wielowymiarowej (Bruce, Bruce i Gedeck, 2021). Na początku w badaniu postanowiono sprawdzić, czy uzyskany model ma taki charakter. Wraz z uzyskiwanymi gorszymi przewidywaniami osiąga się większe wartości RMSE. Jeżeli dokonano by wyliczeń standaryzowanych kowariancji, czyli współczynników korelacji Pearsona, nie uzyskano by odporności na dane odstające. Na podstawie uzyskanych wartości współczynników korelacji względem ceny nieruchomości mieszkalnej, które w większości nie są zadowalające, postanowiono w badaniu obserwować zmiany współczynnika determinacji po dokonywaniu operacji usuwania poszczególnych cech. Zaobserwowano następujące zjawisko: pozostawienie dwóch najlepiej skorelowanych cech z celem o wartościach współczynnika korelacji wynoszącej: dla cechy *sq* – czyli powierzchni mieszkania – 0,77, a dla cechy *rooms* – czyli liczby pokoi – 0,61, spowodowało uzyskanie niższej wartości współczynnika determinacji niż w przypadku usunięcia tylko cechy *id* o współczynniku korelacji 0,07 i *Unnamed: 0*, którego współczynnik wyniósł 0,00. Najwyższy współczynnik korelacji po cesze *rooms* uzyskał *longitude* (długość geograficzna) z wartością 0,24. Jak można się domyślić, jest to bardzo niska wartość. Następnie dla kolumny *floor* (piętro) jest równy 0,09, dla *latitude* (szerokość geograficzna) jest już wartością ujemną –0,10, a dla *year* (rok) wynosi –0,11.

Należy się zastanowić, dlaczego po usunięciu słabo skorelowanych danych wydajność modelu się pogarsza. Pozostawienie tylko cech *sq* i *rooms* z uwzględnieniem takich samych wszystkich pozostałych etapów przygotowania danych, które zostały ponownie wykonane (to znaczy, że powtórnie dokonano podziału zbioru na testowy, treningowy i tak dalej – był powtórzony cały proces od wczytania danych do uruchomienia Lazy Predict i GridSearchCV), sprawiło, iż Lazy Predict dla lasu losowego uzyskał współczynnik determinacji równy 0,54 i pierwiastek błędu średniokwadra-

towego wynoszący 341 061,44. Natomiast wytrenowany model z wykorzystaniem najlepszych kombinacji znalezionych przez GridSearchCV hiperparametrów (spośród ręcznie wprowadzonych dla nich wartości – tzn. że przestrzeń wyszukiwania hiperparametrów była taka sama jak podczas podejścia z usunięciem tylko *Unnamed: 0* i *id*) dał R^2 wynoszący 0,5468390951957882, a RMSE równy 336985,9140325337. Wybranymi przez GridSearchCV parametrami były: *bootstrap* = True, *max_depth* = 16 i *n_estimators* = 70 również dla tej samej wartości *random_state* = 42. Więcej informacji dotyczących wykorzystania w badaniu tego narzędzia znajduje się w punkcie poświęconym wyszukiwaniu hiperparametrów. Należy oczywiście zaznaczyć, iż zaliczenie cech na podstawie wartości korelacji do grupy danych, które są nieistotne, średnio istotne lub istotne, jest zależne od kontekstu (Szeliga, 2017). Konieczne jest zwrócenie uwagi na pewien fakt dotyczący współczynnika korelacji. Wcześniej zaznaczono, iż bada on zależności liniowe między podanymi zmiennymi z wyjątkiem rho Spearmana, który – jak twierdzi autor książki – jest przeznaczony również dla danych cechujących się pewnymi rodzajami nieliniowości (Bruce i in., 2021). W związku z tym można wywnioskować, iż związek między cechami a celem może być nieliniowy. Poza tym faktem sprawdzono również, iż jest mnóstwo cen odstających w zmiennej zależnej, jednak usuwanie dużej liczby odstających wartości, które mają być prognozowane, może mijać się z celem badania. Uzasadnia się to chęcią nieograniczania zbioru danych, poza tym postanowiono w badaniu sprawdzić, czy przy wartościach odstających można uzyskać sensowne wyniki prognoz. Przypuszczenie o braku liniowych zależności potwierdza tabela 1 umieszczona w punkcie dotyczącym Lazy Predict, którą na wyjściu zwraca metoda *fit*, do której przekazuje się dane wywołane na instancji klasy LazyRegressor. Model LinearRegression, na potrzeby którego usunięto tylko cechy *Unnamed: 0* i *id,* uzyskał R^2 wynoszący wartość ujemną równą –8 116 402 552 911 139 470 049 280,00. RMSE wyniósł 1 426 156 939 746 814 208,00. Dla *RandomForestRegressor* R^2 był równy 0,77, a RMSE 241 035,29. Jest to wynik niewyobrażalnie lepszy od *LinearRegression*.

Na rysunku 5 jest widoczne wywołanie metody *corr* z parametrem *method = 'spearman'* na obiekcie *DataFrame* o nazwie *data*. Efektem tego jest wygenerowanie macierzy korelacji. Następnie zamieszczono na tym rysunku fragment kodu odpowiadający za wypisanie wartości posortowanych malejąco współczynników korelacji dla każdej cechy względem ceny – celu. Warto zwrócić uwagę na wartość współczynnika korelacji dla kolumny *id* i *Unnamed*: 0.

Następnie zamieszczono macierz z wartościami współczynnika korelacji dla każdej zmiennej względem wszystkich pozostałych, która jest widoczna na rysunku 6. Wartość współczynnika korelacji ceny z tą samą ceną należy oczywiście pominąć. Jest to przykład całkowitej dodatniej korelacji. Ta informacja nic nie wnosi do interpretacji zależności między zmiennymi. Natomiast ukazuje brak wyraźnej liniowej zależności. Widoczne jest to po niskich współczynnikach. Po liczbie pokoi pozostałe zależności są z pozoru nieistotne. Ale pozory mogą mylić, tym bardziej iż nie

```
correlation = data.corr(method = 'spearman')
correlation['price'].sort_values(ascending=False)
```

```
price          1.00
sq             0.77
rooms          0.61
longitude      0.24
floor          0.09
id             0.07
Unnamed: 0     0.00
latitude      -0.10
year          -0.11
Name: price, dtype: float64
```

Rys. 5. Przypisanie do zmiennej *correlation* zwróconej przez metodę *corr* macierzy korelacji i wyświetlenie posortowanej listy współczynników korelacji dla ceny w porządku malejącym

Źródło: opracowanie własne.

	Unnamed: 0	floor	id	latitude	longitude	price	rooms	sq	year
Unnamed: 0	1.00	0.02	-0.00	-0.00	0.00	0.00	0.00	0.00	0.01
floor	0.02	1.00	-0.02	0.02	0.12	0.09	0.00	-0.01	-0.05
id	-0.00	-0.02	1.00	-0.85	0.01	0.07	-0.01	-0.03	0.05
latitude	-0.00	0.02	-0.85	1.00	-0.02	-0.10	0.00	0.02	-0.05
longitude	0.00	0.12	0.01	-0.02	1.00	0.24	0.00	0.01	-0.16
price	0.00	0.09	0.07	-0.10	0.24	1.00	0.61	0.77	-0.11
rooms	0.00	0.00	-0.01	0.00	0.00	0.61	1.00	0.85	0.04
sq	0.00	-0.01	-0.03	0.02	0.01	0.77	0.85	1.00	-0.01
year	0.01	-0.05	0.05	-0.05	-0.16	-0.11	0.04	-0.01	1.00

Rys. 6. Macierz korelacji przedstawiająca zależności dla każdej cechy względem wszystkich pozostałych przed przygotowywaniem danych

Źródło: opracowanie własne.

weryfikuje się w tej analizie jednocześnie zależności między więcej niż dwiema zmiennymi. Każdy współczynnik ukazuje zależność między konkretną cechą a celem. Jest to analiza dwuczynnikowa (ang. *bivariate analysis*). Można przeprowadzać także analizę jednoczynnikową i wieloczynnikową (Bruce i in., 2021). Również nie ma podstaw do stwierdzenia, iż między zmiennymi nie istnieją zależności nieliniowe. Dlatego postanowiono pozostawić pozostałe cechy. Poza tym wcześniej opisano negatywny wpływ na zmianę wydajności modelu po usunięciu wszystkich cech oprócz *sq* i *rooms*.

5. Wykorzystanie Lazy Predict do wyszukania najwydajniejszego modelu

Liczba dostępnych estymatorów przeznaczonych do regresji jest większa, niż wymagało tego badanie. W związku z tym koniecznie było podjęcie dodatkowych działań umożliwiających wykonanie procesu z wykorzystaniem wybranych algorytmów zgodnie z instrukcją zamieszczoną na stronie użytkownika RAMA serwisu Kaggle (Lazy predict: Choose Classifier or Regressor Models, 2021). Widoczna poniżej lista *regressors* po dodaniu poszczególnych elementów zawierała tylko niezbędne w badaniu estymatory, co zostało przedstawione na rysunku 7.

```
regressors = []
regressors.append(lazypredict.Supervised.REGRESSORS[3])
regressors.append(lazypredict.Supervised.REGRESSORS[7])
regressors.append(lazypredict.Supervised.REGRESSORS[8])
regressors.append(lazypredict.Supervised.REGRESSORS[23])
regressors.append(lazypredict.Supervised.REGRESSORS[24])
regressors.append(lazypredict.Supervised.REGRESSORS[33])
```

Rys. 7. Dodanie do listy regressors wybranych algorytmów

Źródło: opracowanie własne.

Aby dowiedzieć się, jaki regresor znajduje się pod danym numerem, należy wyświetlić zawartość lazypredict.Supervised.REGRESSORS. Na rysunku 8 można zauważyć, iż następnym etapem było przypisanie zawartości utworzonej listy regressors do lazypredict.Supervised.REGRESSORS.

```
lazypredict.Supervised.REGRESSORS = regressors
```

Rys. 8. Przypisanie do lazypredict.Supervised.REGRESSORS wcześniej utworzonej listy z wybranymi algorytmami

Źródło: opracowanie własne.

Operacja ta pozwala na skrócenie czasu oczekiwania na ostateczne wyniki przy ograniczonej mocy obliczeniowej komputera.

Proces użycia narzędzia Lazy Predict należy rozpocząć od utworzenia obiektu klasy LazyRegressor, co jest widoczne na rysunku 9.

```
reg = LazyRegressor(verbose = 0, ignore_warnings = False, custom_metric = None, predictions = True)
```

Rys. 9. Utworzenie instancji klasy LazyRegressor z przekazaniem parametrów

Źródło: opracowanie własne.

W badaniu podczas tej operacji postanowiono przekazać dodatkowo parametr *predictions = True*. Skutkuje to zwróceniem prognoz wybranych modeli po przeprowadzonym procesie uczenia, jednak ze względu na ograniczony rozmiar artykułu nie zostaną one przedstawione. Następnie na utworzonej instancji klasy LazyRegressor należy wywołać metodę *fit*, przekazując do niej dane treningowe i testowe. Wywołanie metody szkolącej modele i zwracającej ich ranking widoczne jest na rysunku 10.

```
models, predictions = reg.fit(X_train, X_test, y_train, y_test)
```

Rys. 10. Wywołanie metody *fit* na obiekcie klasy LazyRegressor przeprowadzającej trenowanie modeli i utworzenie tabeli z metryką mierzącą ich wydajność i wartość błędu

Źródło: opracowanie własne.

Metoda ta pozwoliła na dokonanie procesu uczenia modeli z wykorzystaniem wybranych na początku sekcji algorytmów przeznaczonych do regresji. Generalnie wygląda to podobnie w porównaniu z przeprowadzaniem uczenia dla dowolnego estymatora w Scikit-learn.

W tabeli 1 umieszczono porównanie wydajności wybranych modelów za pomocą R^2 i RMSE. Wyniki pokazują, że faktycznie między danymi nie istnieją zależności liniowe pozwalające na dokonywanie prognoz przez LinearRegression lub algorytm ten nie jest w stanie z innego powodu utworzyć właściwego modelu.

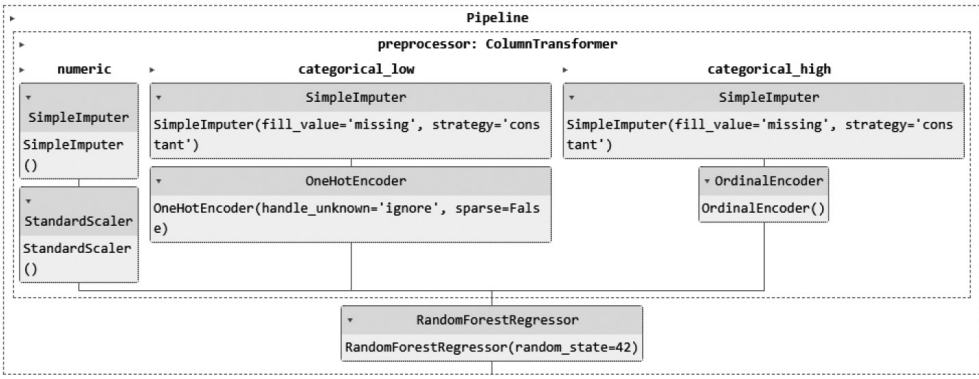
Tabela 1. Porównanie współczynnika determinacji i pierwiastka błędu średniokwadratowego wybranych algorytmów

Model	R-Squared	RMSE
RandomForestRegressor	0.768159019712789	241035.29003576894
ExtraTreesRegressor	0.752164617562571	249210.98665996664
ExtraTreeRegressor	0.6879324868795367	279646.77175657946
DecisionTreeRegressor	0.6287710542122855	305004.6399490731
LinearSVR	–1.5550899891007104	800181.7284448043
LinearRegression	–8.11640255291114e+24	1.4261569397468142e+18

Źródło: opracowanie własne na podstawie wyników działania biblioteki Lazy Predict.

Najwyższy wynik uzyskał wykorzystany w badaniu RandomForestRegressor z wartością *R-Squared* równą 0,768 i pierwiastkiem błędu średniokwadratowego wynoszącym 241 035. Zbliżone wartości metryk osiągnął ExtraTreesRegressor. Po analizie wyników osiągniętych przez wszystkie modele postanowiono wykorzystać w badaniu RandomForestRegressor. Warto odnotować, iż DecisionTreeRegressor uzyskał wynik gorszy o ok. 0,14 dla współczynnika determinacji. Zauważalna jest przewaga lasu losowego składającego się z wielu drzew w porównaniu do drzewa

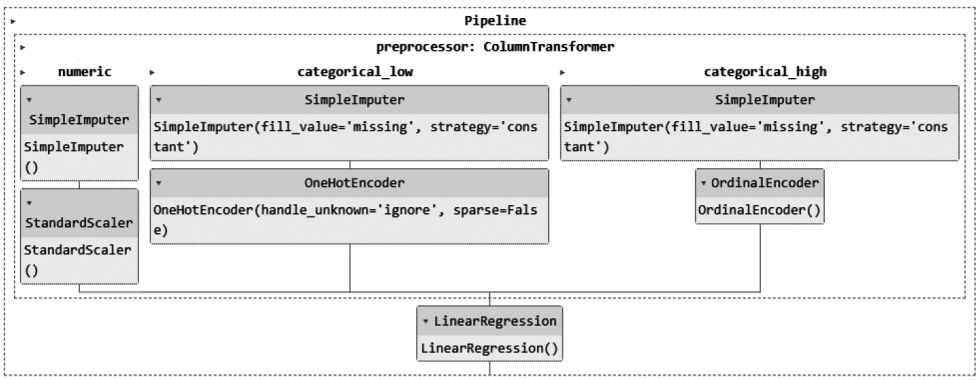
decyzyjnego. Schemat przedstawiony na rysunku 11 przedstawia potok (ang. *pipeline*) dla RandomForestRegressor.



Rys. 11. Potok LazyPredict dla RandomForestRegressor

Źródło: opracowanie własne na podstawie wyników działania biblioteki Lazy Predict.

W związku z ograniczeniem objętości artykułu postanowiono nie opisywać poszczególnych elementów schematów na rysusnkach 11 i 12. Na rysunku 12 widoczny jest potok dla LinearRegression.



Rys. 12. Potok Lazy Predict dla LinearRegression

Źródło: opracowanie własne na podstawie wyników działania biblioteki Lazy Predict.

Wygenerowane graficzne formy potoków, które zostały zamieszczone na rysunkach 11 i 12, można uzyskać przez wywołanie na obiekcie Lazy Regressor metody provide_models, przekazując tej metodzie wszystkie wcześniej wykorzystane zmienne z danymi i przypisując rezultaty działania metody provide_models do zmiennej. Zmienna ta zawierała słownik, w którym można było przez podanie w kluczu nazwy algorytmu wyświetlić widoczny schemat na ekranie. Mimo niewykorzystania w ba-

daniu imputacji danych za pomocą SimpleImputer, wykorzystania tylko OneHotEncoder dla danych kategorycznych, wynik nieznacznie się różnił. Zastanawiające jest, dlaczego Lazy Predict dokonał imputacji, jeżeli nie odnotowano żadnych pustych próbek. Mimo wcześniejszego przeskalowania danych Lazy Predict postanowił ponownie je przeskalować.

6. Wyszukiwanie najlepszych wartości hiperparametrów z wykorzystaniem GridSearchCV

Proces należy rozpocząć od utworzenia obiektu klasy RandomForestRegressor. W tablicy parameters zawarty jest słownik zawierający podane jako klucze nazwy hiperparametrów, dla których chce się znaleźć jak najlepsze wartości (spośród ręcznie wpisanych). Dana weryfikowana wartość hiperparametru jest zapisywana jako wartość dla danego klucza. W naszym przypadku kluczem jest *n_estimators* – liczba drzew w lesie, *max_depth* – wysokość drzewa , *bootstrap* – wzmocnione tworzenie drzew i *random_state* – losowa wartość ziarna

Następnie tworzony jest obiekt klasy GridSearchCV, do którego przekazuje się wybrany obiekt regresora, wybraną tablicę z hiperparametrami i liczbę 3 dla parametru verbose, co odpowiada za wyświetlanie wszystkich dostępnych do wyboru komunikatów na ekranie dotyczących przeprowadzanego procesu. Na samym końcu należy przeprowadzić trenowanie, przekazując cechy i cel ze zbioru treningowego. Wszystko to przedstawiono na rysunku 13.

```
parameters=[
   {'n_estimators': [70,80,90],
    'max_depth':[16,20,25],
    'bootstrap':[True,False],
    'random_state':[42]}
]
rgr = GridSearchCV(model, parameters, verbose=3)
rgr.fit(X_train,y_train)
```

Rys. 13. Wykorzystanie GridSearchCV do wyszukania najlepszej wartości dla hiperparametru n_estimators, max_depth i bootstrap dla random_state = 42

Źródło: opracowanie własne.

Powyższa konfiguracja sprawiła, iż model był trenowany 90 razy. Najlepsza kombinacja wartości dla hiperparametrów z podanej przestrzeni wyszukiwania jest zawarta w atrybucie *best_params_* obiektu klasy GridSearchCV zawierającym słownik. Przedstawia się ona następująco: dla hiperparametru bootstrap jest to True, dla max_depth 25, a dla n_estimators – 90. Wynik uzyskany w walidacji krzyżowej na zbiorze treningowym zawarty w *best_score_* wynosi 0,8321072982000401.

Wytrenowanie modelu z bezpośrednim użyciem Scikit-learn z przekazaniem uzyskanych wartości hiperparametrów i weryfikacją jego wydajności na zbiorze testowym dało wynik R^2 wynoszący 0,7709571696543107. RMSE był równy 239 576,31523881434. Jest to nieznacznie wydajniejszy model niż ten, który uzyskał Lazy Predict. Natomiast MAE (ang. *Mean Absolute Error*) dla uzyskanego modelu wyniósł 77 557,08444888794. Jest to metryka odporniejsza na wartości odstające niż pierwiastek błędu średniokwadratowego.

7. Zakończenie

Celem badania było przeprowadzenie procesu uczenia maszynowego na danych dotyczących ograniczonej liczby nieruchomości mieszkaniowych z wybranych miast Polski przez różne algorytmy i wybranie najwydajniejszego modelu. Badanie było ograniczone tylko do wybranej liczby cech. Nie uwzględniało sposobu, w jaki rzeczoznawca majątkowy dokonuje ustalenia ceny mieszkania.

W efekcie przeprowadzonego badania uzyskano ranking wybranych algorytmów. Najlepszym estymatorem okazał się RandomForestRegressor, czyli regresor lasu losowego. Mimo wypisanych ograniczeń badanie pokazało, iż można wytrenować model uczenia maszynowego lasu losowego mającego względnie akceptowalną wydajność w przypadku danych nieposiadających wielu liniowo skorelowanych cech z wartością prognozowaną. W przyszłości można zidentyfikować dodatkowe czynniki mające wpływ na ceny nieruchomości po konsultacji z rzeczoznawcą majątkowym. Uwzględnienie istotnych czynników, od których zależą ceny nieruchomości, z pewnością zmniejszy pierwiastek błędu średniokwadratowego i zwiększy wartość współczynnika determinacji, co przełoży się na lepszą jakość modelu dokonującego prognoz.

Wart odnotowania jest fakt, iż badanie skupiło się na tradycyjnych algorytmach uczenia maszynowego. Istnieje możliwość rozszerzenia zakresu badania. Można uwzględnić podczas wyszukiwania kombinacji z wykorzystaniem GridSearchCV, RandomizedSearchCV lub BayesSearchCV (biblioteka Scikit-Optimize) hiperparametr *max_features*, czyli liczbę cech przeznaczonych do podziału w lesie losowym. Istnieje możliwość porównania wydajności otrzymanych modeli tradycyjnego uczenia maszynowego z modelami głębokiego uczenia (ang. DL – *Deep Learning*). W tym celu można wybrać odpowiednią do tego sieć neuronową i na przykład wykorzystać poznane już w niniejszym badaniu narzędzie GridSearchCV w celu wyszukania najlepszych wartości hiperparametrów z podanej przestrzeni wyszukiwania. Istnieje również możliwość skorzystania z dedykowanego narzędzia do dostrajania hiperparametrów dla modeli Keras o nazwie KerasTuner (KerasTuner API, 2023).

Literatura

API Reference Pandas. (2022). Pandas.pydata.org. https://pandas.pydata.org/pandas-docs/version/1.5/reference/index.html

API Reference Scikit-learn. (2023). Scikit-learn.org. https://scikit-learn.org/stable/modules/classes.html

Bruce, P., Bruce, A. i Gedeck, P. (2021). *Statystyka praktyczna w data science. 50 kluczowych zagadnień w językach R i Python.* Helion SA.

Cegielski, D. (2021). *House Prices in Poland.* Kaggle.com. https://www.kaggle.com/datasets/dawidcegielski/house-prices-in-poland

Géron, A. (2020). *Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow.* Helion SA.

KerasTuner API. (2023) Keras.io. https://keras.io/api/keras_tuner/

Lazypredict: Choose Classifier or Regressor Models. (2021). Kaggle.com. https://www.kaggle.com/code/rahulgoel1106/lazypredict-choose-classifier-or-regressor-models/notebook

Szeliga, M. (2017). *Data science i uczenie maszynowe.* Wydawnictwo Naukowe PWN SA.

Residential Real Estate Price Forecast Using Machine Learning

Abstract: Machine learning plays an increasingly important role in today's world. It has many applications that can be used in science and enterprises. It allows making forecasts of continuous values with the use of a regression estimator. The aim of the study is to carry out a forecast of residential real estate prices and verify its effectiveness using various algorithms in Python 3, the Pandas library, LazyPredict, Scikit-learn in the Jupyter Notebook environment. The research methods used were: literature analysis, documentation analysis, comparative study – using the LazyPredict library to determine the most efficient estimator according to the value of the determination coefficient, using GridSearchCV to find the best hyperparameters of the random forest model. The analysis made it possible to choose the best algorithm from among the selected ones, which turned out to be the random forest regressor to carry out the learning process on the data.

Keywords: residential real estate price forecast, machine learning, LazyPredict, GridSearchCV, RandomForestRegressor