Praca Dyplomowa Inżynierska

Cezary Czarnogłowski
200796

Zastosowanie techniki sztucznej inteligencji do wyznaczenia wielkości mocy umownej w zakładach przemysłowych

Application of artificial intelligence techniques for determining the amount of contracted power in industrial facilities

Praca dyplomowa na kierunku:
Informatyka

Praca wykonana pod kierunkiem
dr. Rafika Nafkhi
Instytut Informatyki Technicznej
Katedra Systemów Informatycznych

Warszawa, rok 2023

SZKOŁA GŁÓWNA GOSPODARSTWA WIEJSKIEGO

Wydział Zastosowań Informatyki i Matematyki

Oświadczenie Promotora pracy

Oświadczam, że niniejsza praca została przygotowana pod moim kierunkiem i stwierdzam, że spełnia ona warunki do przedstawienia tej pracy w postępowaniu o nadanie tytułu zawodowego.

Data . . . . . Podpis promotora . . . . . .

Oświadczenie autora pracy

Świadom/a odpowiedzialności prawnej, w tym odpowiedzialności karnej za złożenie fałszywego oświadczenia, oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. 2019 poz. 1231 z późn. zm.)

Oświadczam, że przedstawiona praca nie była wcześniej podstawą żadnej procedury związanej z nadaniem dyplomu lub uzyskaniem tytułu zawodowego.

Oświadczam, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną. Przyjmuję do wiadomości, że praca dyplomowa poddana zostanie procedurze antyplagiatowej.

Data . . . . . Podpis autora pracy . . . . . .

Streszczenie

Zastosowanie techniki sztucznej inteligencji do wyznaczenia wielkości mocy umownej w zakładach przemysłowych

Tematem niniejszej pracy było wykorzystanie techniki sztucznej inteligencji do wyznaczenia takiej wielkości mocy umownej, aby ta wielkość była możliwie najbardziej optymalna z punktu widzenia danego zakładu przemysłowego.

Słowa kluczowe – sztuczna inteligencja, uczenie maszynowe, moc umowna, analiza regresji

Summary

Application of artificial intelligence techniques for determining the amount of contracted power in industrial facilities

The subject of this thesis was to apply artificial intelligence techniques to determine the contracted power value that is optimal from the perspective of a particular industrial facility.

Keywords – artificial intelligence, machine learning, contracted power, regression analysis

Spis treści

1 Wstęp

W dobie stale rosnącego zapotrzebowania energetycznego, efektywne zarządzanie i wykorzystanie zasobów energetycznych stało się jednym z najistotniejszych aspektów funkcjonowania zakładów przemysłowych. W ramach zawarcia rocznej umowy dotyczące energii elektrycznej między odbiorcą, a dostawcą energii na podstawie indywidualnych ustaleń odbiorca energii zobowiązany jest do ustalenia wartości mocy umownej, które będą obowiązywać w każdym miesiącu trwania umowy. Dobór odpowiedniej wartości mocy umownej jest kwestią niezwykle istotną dla optymalnego wykorzystania możliwości zakładu [1]. Przyjęcie zbyt wysokiej wartości mocy umownej naraża przedsiębiorstwo na wygenerowanie wysokich kosztów. Przyjęcie zbyt niskiej wartości mocy może prowadzić do wielokrotnego przekroczenia przyjętych limitów mocy, co z kolei skutkuje nakładaniem kar finansowych. Kara za przekroczenie wartości mocy wynosi iloczyn stałej stawki oraz dziesięć największych nadwyżek ponad moc umowną lub dziesięciokrotność maksymalnej nadwyżki.

Rozwiązaniem problemu doboru optymalnej wartości mocy umownej [2] jest zastosowanie techniki sztucznej inteligencji, która zostanie wyłoniona jako technika o najmniejszym błędzie predykcyjnym na podstawie analizy porównawczej wybranych metod uczenia maszynowego. W ramach projektu inżynierskiego do predykcji wartości mocy umownej dla przedsiębiorstwa, jakim jest elektrownia gazowo-parowa pracująca z wykorzystaniem cyklu kombinowanego, została przeprowadzona analiza porównawcza takich metod uczenia maszynowego jak: Regresja liniowa, algorytm K-najbliższych sąsiadów, Drzewo decyzyjne, Las losowy, Wzmocnienie gradientu oraz Sztuczna sieć neuronowa. Metoda, która uzyskała najmniejszy błąd predykcji została wybrana jako optymalna technika do budowy modelu. Następnie na podstawie wyselekcjonowanego algorytmu zbudowano model, który został oceniony zarówno pod względem błędu predykcji, dopasowania do rzeczywistych danych oraz ogólności.

Celem pracy jest stworzenie możliwie najbardziej dokładnego i ogólnego modelu regresji do przewidywania wartości ciągłej, jaką jest moc umowna. Taki model jest w stanie zapewnić, że przewidziane wartości mocy będą na tyle precyzyjne, aby możliwie najdokładniej zoptymalizować wydatki energetyczne danego przedsiębiorstwa.

2 Regulacje prawne dotyczące mocy umownej

Moc umowna stanowi jeden z najbardziej kluczowych elementów umów o dostawę energii elektrycznej. Wartość mocy umownej w bezpośredni sposób przekłada się bowiem na dobór taryfy oferowanej przez dostawcę energii. Dlatego jest ściśle określona w różnych dokumentach prawnych regulujących rynek energetyczny.

2.1 Pojęcie mocy umownej oraz formy karania za jej przekroczenie

Zgodnie z §2 pkt. 10 ust. a Rozporządzenia Ministra Gospodarki z dnia 4 maja 2007 r. w sprawie szczegółowych warunków funkcjonowania systemu elektroenergetycznego moc umowną [3] określa się jako: "moc czynną pobieraną lub wprowadzaną do sieci, określoną w umowie o świadczenie usług przesyłania lub dystrybucji energii elektrycznej, umowie sprzedaży energii elektrycznej albo umowie kompleksowej jako wartość nie mniejszą niż wyznaczoną jako wartość maksymalną ze średniej wartości mocy w okresie 15 minut, z uwzględnieniem współczynników odzwierciedlających specyfikę układu zasilania odbiorcy". Zatem moc umowną określa się jako maksymalną moc, jaką odbiorca może pobrać z sieci w danym momencie, jest to więc pewna umowna granica dla odbiorcy energii. Zwykle moc umowna wyrażana jest w kilowatach kW, jednak w przypadku gdy odbiorcą energii elektrycznej jest duże przedsiębiorstwo wyrażana jest również w megawatach MW.

W przypadku nakładania kar finansowych na odbiorcę energii elektrycznej za naruszenie warunków umowy i wielokrotne przekroczenie ustalonej wartości mocy umownej również występują odpowiednie regulacje prawne. Zgodnie z §48 pkt. 3 Rozporządzenia Ministra Klimatu I Środowiska z dnia 29 listopada 2022 r. w sprawie sposobu kształtowania i kalkulacji taryf oraz sposobu rozliczeń w obrocie energią elektryczną [4] za przekroczenie mocy umownej określonej w umowie nalicza się karę finansową na dwa sposoby. Pierwszy rodzaj kary stanowi iloczyn stałej stawki oraz sumy dziesięciu największych wielkości nadwyżek mocy pobranej ponad moc umowną. Drugi zaś jest określony jako iloczyn stałej stawki oraz dziesięciokrotność maksymalnej nadwyżki pobranej ponad moc umowną.

3 Pojęcie sztucznej inteligencji w kontekście uczenia maszynowego

Sztuczna inteligencja (zwana również w sposób skrótowy *SI* używając polskiego akronimu lub *AI* stosując angielski odpowiednik) jeszcze do niedawna było uznawana za bardzo tajemnicze i nieznane szerzej pojęcie, które zarezerwowane było jedynie dla wąskiego grona specjalistów i naukowców z wielu dziedzin nauki oraz specjalistów z zakresu technologii informacyjnej (*ang. IT*). We współczesnych realiach wraz z rozwojem sztucznej inteligencji i silnym wzrostem jej wykorzystywania w wielu ogólnodostępnych systemach można zaobserwować rosnące znaczenie tego rozwiązania w życiu codziennym. *SI* z powodzeniem znajduje zastosowanie w takich dziedzinach jak medycyna, handel, rolnictwo, ekonomia, astronomia i wielu innych. Właściwie bardzo trudno znaleźć dziedzinę, która w szerszym, bądź węższym zakresie nie wykorzystuje rozwiązań sztucznej inteligencji do automatyzacji, usprawnienia, czy zwiększenia efektywności procesów wykonywanych w ramach danej działalności.

Termin "sztuczna inteligencja" został sformułowany przez Profesora Johna McCarthy'ego w 1955 roku. Określał on, że *SI* to "nauka i inżynieria tworzenia inteligentnych maszyn ze specjalnym uwzględnieniem programów komputerowych" [5]. Inny z pionierów sztucznej inteligencji Profesor Marvin Minsky mówił, że "sztuczna inteligencja to nauka o maszynach realizujących zadania, które wymagają inteligencji, gdyby były realizowane przez człowieka" [6]. Można zatem ująć, że sztuczna inteligencja jest działem nauki skupiającym się na tworzeniu maszyn oraz algorytmów, które starają się odzwierciedlić zdolność do uczenia się, rozwiązywania określonych problemów na podstawie zdobytej wiedzy, dostosowywania się do zmiennych warunków, czy podejmowania decyzji.

W ramach dziedziny *SI* istnieje dział o nazwie uczenie maszynowe, które zajmuje się konstruowaniem i rozwijaniem algorytmów oraz modeli matematycznych pozwalających systemom komputerowym na rozwiązanie danego problemu w sposób samodzielny, bez konieczności ingerowania w ten proces ze strony człowieka. Proces uczenia maszynowego polega na uczeniu algorytmów lub modeli matematycznych na konkretnych danych. Pozwala na poznanie zależności i wzorców, które występują w danych, dzięki czemu model jest zdolny do podejmowania decyzji i rozwiązywania postawionych przed nim zadań [7].

3.1 Rodzaje uczenia maszynowego

3.1.1 Uczenie nadzorowane

Uczenie nadzorowane (*ang. supervised learning*) jest rodzajem uczenia maszynowego, w którym technika uczenia modelu opiera się na danych, które zawierają dane wejściowe wraz z danymi wyjściowymi tzw. etykiety lub klasy stanowiące rozwiązanie danego problemu. Oznacza to, że uczenie nadzorowane opiera się na rozpoznaniu zależności występujących między danymi wejściowymi, a danymi wyjściowymi [8, 9]. Obszarami, które wykorzystują uczenie nadzorowane są:

- **Analiza regresji** - jest procesem, który służy do prognozowania poszukiwanych wartości wyjściowych będącymi wartościami ciągłymi, na podstawie danych wejściowych. Jednym z przykładów zastosowania analizy regresji, może być przewidywania wartości nieruchomości, czy wartości samochodów na podstawie sprecyzowanych danych wejściowych.
- **Klasyfikacja** - jest procesem, który służy do przewidywania wartości dyskretnych, tzn. przewidywania przynależności do odpowiedniej klasy na podstawie danych wejściowych. Przykładem klasyfikacji może być rozpoznawanie gatunków motyli na podstawie zdjęć.

3.1.2 Uczenie nienadzorowane

Uczenie nienadzorowane (*ang. unsupervised learning*) jest rodzajem uczenia maszynowego, w którym do trenowania modelu używa się tzw. nieoznakowanych danych. Nieoznakowane dane charakteryzują się tym, że nie zawierają etykiet, nie zawierając więc wartości, które można byłoby określić jako wartości poszukiwane. Stosując modele przeznaczone dla tego typu uczenia, możliwe jest poznanie struktury analizowanych danych oraz uzyskanie użytecznych informacji [8, 10]. Uczenie nienadzorowane znajduje zastosowanie przy rozwiązywaniu następujących problemów :

- **Analiza skupień** - jest procesem, który służy do grupowania obiektów, które posiadają taki stopień podobieństwa, by mogły być zakwalifikowane do danej grupy.
- **Redukcja wymiarowości** - jest procesem, który służy do przekształcenia zbioru danych, w taki sposób, aby zredukować pewne fragmenty danych, które określa się jako wymiary lub zmienne, przy minimalizacji strat ważnych informacji, które pierwotne dane zawierał.

3.1.3 Uczenie częściowo nadzorowane

Uczenie częściowo nadzorowane (*ang. semi-supervised learning*) jest rodzajem uczenia maszynowego, które łączy w sobie zarówno cechy uczenia nadzorowanego i uczenia nienadzorowanego. Ten rodzaj uczenia często stosuje się, gdy dane przeznaczone do trenowania modelu nie mają w pełni przypisanych etykiet. W takich przypadkach algorytmy oparte na uczeniu częściowo nadzorowanym same proponują odpowiedzi. Przykładem zastosowania tego typu uczenia jest klasyfikacja obrazów, w których nie wszystkie obrazy mają nadane etykiety [8, 11].

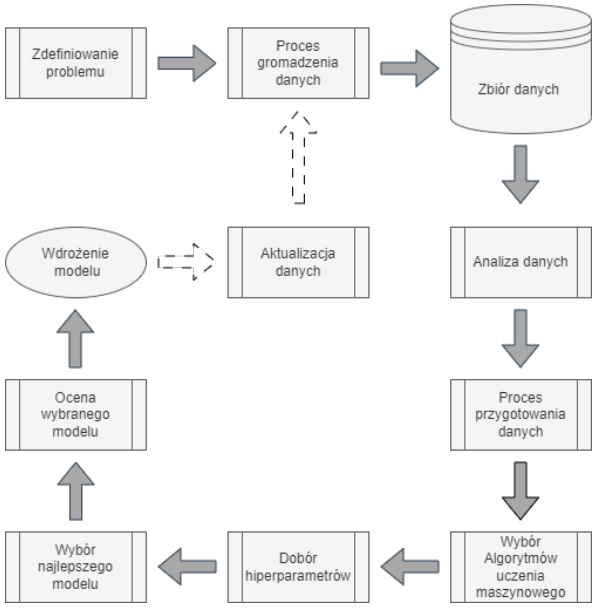
3.1.4 Uczenie przez wzmocnienie

Uczenie przez wzmocnienie (*ang. reinforcement learning*) jest rodzajem uczenia maszynowego, które jest zupełnie odmienne od wyżej wymienionych typów uczenia maszynowego. W uczeniu przez wzmocnienie nie stosuje się bowiem żadnych wcześniej przygotowanych zestawów danych. Polega ono na tworzeniu systemu, nazywanego agentem, który na podstawie podjętych działań otrzymuje nagrody oraz kary w ramach interakcji ze środowiskiem. Nagrody określają jak dobrze agent radzi sobie z wykonywanym zadaniem, natomiast w przypadku jeśli agent wykonuje niepożądane działania jest karany. Celem agenta jest maksymalizowanie nagród i minimalizowanie kar [8, 12]. Przykładem zastosowania uczenia przez wzmocnienie może być zastosowanie agenta, który poznaje zasady i uczy się osiągać jak najwyże wyniki w danej grze.

4 Projektowanie i budowa modelu uczenia maszynowego do prognozowania wartości ciągłych

Tworzenie skutecznego modelu uczenia maszynowego to złożony proces, który wymaga odpowiedniego podejścia i zachowania dbałości. Każda składowa procesu może mieć znaczący wpływ na jakość finalnego modelu.

Proces projektowania i budowy modelu można podzielić na kilka podprocesów. Pierwszym krokiem jest zdefiniowanie problemu i określenie w jaki sposób można ten problem rozwiązać. Mając już zdefiniowany problem w następnym kroku przystępuje się do gromadzenia niezbędnych zestawów danych. Dysponując odpowiednio dużym zestawem danych zwykle dokonuje się procesu ich analizy i przygotowania do trenowania modelu. Kolejnym krokiem jest dobór algorytmów, które są brane pod uwagę jako potencjalne narzędzia do wykorzystania w modelu. Dla każdego algorytmu można dokonać procesu tzw. dostrajania hiperparametrów, czyli doboru unikalnych parametrów dla każdego rodzaju algorytmu, które zmienają zdolności predykcyjne danego modelu. Następnie spośród rozważanych modeli wybiera się jeden, finalny model, który jest proponowanym rozwiązaniem do zastosowania. Cały proces prezentuje **Rys. 4.1.**



Rys. 4.1. Schemat ideowy projektowania modelu predykcyjnego.

W tym rozdziale zostały przedstawione poszczególne etapy powyższego procesu w zastosowaniu dla modeli, które prognozują wartości ciągłe, inaczej mówiąc są wykorzystywane w analizie regresji.

4.1 Zdefiniowanie analizowanego problemu

Zdefiniowanie analizowanego problemu pozwala poznać specyfikę problemu, który projektowany model ma rozwiązać. Precyzyjna analiza pozwala określić jaki typ uczenia maszynowego należy zastosować, jakie dane mogą okazać się potrzebne, jakie źródła zapewniają takie dane oraz jak je pozyskać. Wstępnie można również rozważyć, jakie algorytmy uczenia maszynowego można zastosować, aby przyspieszyć cały proces.

4.2 Gromadzenie danych

Posiadające obraz specyfiki problemu oraz wiedząc jakie dane mogą okazać się użyteczne do jego rozwiązania można przystąpić do etapu zebrania danych, na podstawie których będzie budowany model. Im wyższa jakość zebranych danych, tym dokładniej wytrenowany model będzie w stanie odzwierciedlić charakterystykę danych, a tym samym jego zdolności predykcyjne będą bardziej dokładne.

W przypadku regresji szczególny nacisk powinno się położyć w dwóch aspektach, którymi są zmienne niezależne oraz obserwacje. Odpowiednia liczba zmiennych niezależnych i obserwacji oraz ich zróżnicowanie i wysoka jakość pozwoli na dokładniejsze odwzorowanie rzeczywistego problemu przez model. W przypadku gdy danych jest mało, ich zróżnicowanie jest niewielkie lub zmienne niezależne nie są wystarczająco powiązane ze zmienną celu, jakikolwiek model, oparty o bardzo skuteczne algorytmy nie będzie w stanie rozwiązać postawionego zagadnienia w sposób skuteczny.

4.3 Analiza i przygotowanie danych do procesu uczenia modelu

Aby poznać wykorzystywane dane niezbędne jest przeprowadzenie procesu ich eksploracji. Analiza pozwala na szczegółowy wgląd do danych, które są używane w procesie budowy modelu. Do takiej analizy stosuje się dostępne techniki matematyczne oraz wizualizacyjne, które pozwolą w dokładny sposób poznać charakterystyki zmiennych.

Analiza wskazuje, czy w danych nie znajdują się błędnie uzupełnione wartości, brakujące wartości, czy też powtarzające się obserwacje. Oprócz weryfikacji poprawności danych, konkretna analiza pozwala na szczegółowe poznanie jaki zasięg wartości występuje w poszczególnych zmiennych, ich rozkładu, czy też korelacji występującej między zmiennymi. Niezwykle istotne jest również zidentyfikowanie, czy w danych występują obserwacje odstające.

Szczegółowe poznanie charakterystyk i zależności występujących w danych pozwala na zweryfikowanie wcześniej rozważanych algorytmów i przystąpienie do przygotowania danych do użycia w dalszym etapach. Na takie przygotowanie zwykle składa się:

- **obsługa brakujących elementów**
- **obsługa obserwacji odstających**
- **obsługa danych kategoryzujących**
- **skalowanie cech (zmiennych) liczbowych**

4.3.1 Obsługa brakujących danych

Brakujące dane definiuje się jako niezapisane wartości, nieznane wartości lub braki w wartościach w zmiennej w ramach danej obserwacji będącej częścią zbioru danych wykorzystywanych w procesie tworzenia modelu. Występowanie brakujących wartości jest powszechnym zjawiskiem występujących w wielu analizach, można zatem stwierdzić, że stanowi wręcz nieodłączny ich element. Oczywistym jest więc fakt, że brakujące dane w bezpośredni sposób wpływają na funkcjonowanie danego modelu.

Przed przystąpieniem do obsługi brakujących danych, pomocne może okazać się zdefiniowanie przyczyny ich występowania. Załóżmy, że zbiór danych składający się z pewnej liczby zmiennych oraz pewnej liczby obserwacji jest postaci $D = \{D_{obs}, D_{mis}\}$, gdzie D_{obs} to dane bez brakujących wartości, a D_{mis} to dane brakujące. Ponadto zbiór danych jest macierzą wartości $D = v_{ij}$. R_{ij} stanowi macierz wskaźników, taką że:

$$R_{ij} = \begin{cases} 1, & \text{jeśli dana wartość } v_{ij} \text{ wystepuje} \\ 0, & \text{jeśli dana wartość } v_{ij} \text{ nie wystepuje} \end{cases}$$

,ϕ stanowi pewien nieznany parametr, który może reprezentować czynniki zewnętrzne. Znając powyższe założenia wyróżniamy trzy główne typy wystepowania brakujących danych [13]:

- **Całkowicie przypadkowe występowanie wartości brakujących** *(ang. Missing Completely at Random)* - charakteryzują się tym, że prawdopodobieństwo Pr występowania brakujących wartości nie jest zależna zarówno od wartości występujących w danych, jak i wartości brakujących. Dopuszcza się jednak fakt, że brakujące wartości mogą być zależne od nieznanego parametru ϕ.

  $$Pr(R|D,\phi) = Pr(R|\phi) \tag{4.1}$$

  Innymi słowy braki w danych nie wynikają z żadnych zmiennych występujących w danych.
- **Przypadkowe występowanie wartości brakujących** *(ang. Missing at Random)* charakteryzują się tym, że prawdopodobieństwo Pr występowania brakujących wartości wynika z danych obserwowanych D_{obs} oraz parametru ϕ.

  $$Pr(R|D,\phi) = Pr(R|D_{obs},\phi) \tag{4.2}$$

  W przypadku tego typu nie występuje powiązanie między brakami, a wartościami zmiennej w której braki zaobserwowano.
- **Brakujące dane nielosowe** *(ang. Missing Not at Random)* - braki danych charakteryzują się tym, że prawdopodobieństwo Pr występowania brakujących wartości zależy od brakujących danych D_{mis} oraz parametru ϕ.

  $$Pr(R|D,\phi) = Pr(R|D_{mis},\phi) \tag{4.3}$$

  Oznacza to, że między brakującymi wartościami w obrębie danej zmiennej, a tą zmienną może występować jakaś zależność.

Znajomość charakterystyki brakujących wartości pozwala na dobór metody radzenia sobie z nimi. Wyróżnia się trzy podstawowe podejścia:

- **Usuwanie danych**
- **Uzupełnianie brakujących danych**
- **Pozostawienie brakujących danych**

Usunięcie danych odnosi się do usunięcia konkretnych obserwacji lub zmiennych, które zawierają brakujące wartości. W przypadku jeśli ilość rekordów zawierających brakujące wartości jest niewielka w stosunku do ilości danych, można rozważyć usunięcie tych

rekordów. Jeśli jednak brakujących danych w rekordach jest więcej ich usunięcie mogłoby zniekształcić rozkład danych, bądź spowodować utratę istotnych informacji. W sytuacji, w której jedna ze zmiennych (kolumna, cecha) zawiera wiele brakujących wartości, usunięcie całej zmiennej może być sposobem na poradzenia sobie z brakującymi danymi. Jednak podobnie jak w przypadku usuwania obserwacji, usunięcie całej zmiennej może spowodować zniekształcenie danych i utratę ważnych informacji.

Kolejną metodą radzenia sobie z brakującymi danymi, jest metoda uzupełniania brakujących wartości. Metoda ta pozwala na wyeliminowanie negatywnych skutków usunięcia z danych rekordów bądź całych zmiennych. Istnieje wiele metod uzupełniania brakujących danych, a najbardziej popularne są metody [14]:

- **Uzupełnienie na podstawie podobnych rekordów** - metoda ta polega na znalezieniu najbardziej podobnego rekordu, do tego, w którym występują brakujące dane pod względem zmiennych i uzupełnieniu komórki z brakiem taką samą wartością, jaka wystepuje w podobnym rekordzie.
- **Uzupełnienie wartościami najczęściej występującymi** - metoda polega na uzupełnieniu brakujących wartości wartościami najczęściej występującymi w danej zmiennej.
- **Uzupełnienie za pomocą regresji** - metody, które polegają na przewidywaniu brakującej wartości na podstawie innych zmiennych objaśniających. Jedną z najpopularniejszych tego typu metod jest regresja liniowa, która przybliża wartość brakującej zmiennej przy pomocy funkcji liniowej. Innym rodzajem metody jest zastosowanie algorytmu k-najbliższych sąsiadów [15], która pozwala na uzupełnienie brakującej wartości poprzez wartości “najbliższych sąsiadów”, innymi słowy punktów, które znajdują się najbliżej punktu z brakującą wartością.
- **Uzupełnienie za pomocą estymacji** - metody polegające na przewidywaniu brakującej wartości, na podstawie pozostałych danych w danej zmiennej. Jedną z najbardziej popularnych metod estymacji jest uzupełnienie brakującej wartości średnią wyliczoną na podstawie całej zmiennej.

Ostatecznie możliwe jest także pozostawienie brakujących danych, jednak należy pamiętać, że nie wszystkie algorytmy przy takich danych okażą się skuteczne.

4.3.2 Obsługa obserwacji odstających

Obserwacje odstające definiuje się jako obserwacje, które różnią się znacząco od pozostałych elementów danych. Definicja ta nie określa w bezpośredni sposób, od jakiej wartości zaliczymy daną obserwację jako odstającą, co właściwie pokazuje nam złożoność i nieoczywistość problematyki obserwacji odstających.

Występowanie obserwacji odstających w danych może wynikać z wielu przyczyn. Istnieje możliwość, że wartość odstająca jest konsekwencją błędu. Błędy te mogą wynikać z wystąpienie błędu pomiarowego w danym urządzeniu pomiarowym, np. z powodu awarii. Kolejnym źródłem błędów może być błędna interpretacja zadanego pytania lub jego złe sformułowanie, czy też błędna odpowiedź udzielone przez danego ankietera. Ostatecznie możliwe jest również wprowadzenie niepoprawnych wartości do danych. Warto jednak mieć na uwadze, że obserwacje odstające mogą być również mimo swej nietypowej wartości, jak najbardziej prawidłowe. Obserwacje odstające mogą odzwierciedlać bowiem prawdziwe zjawisko będące bardzo istotne z punktu widzenia przeprowadzanej analizy.

Istnieje wiele sposobów pozwalających wykryć obserwacje odstające. Jeśli rozpatrywana zmienna ma rozkład normalny lub zbliżony do normalnego można zastosować jedną z poniższych metod:

- **Reguła 3 Sigm** - polega na wyznaczeniu przedziału $[\mu - 3\sigma, \mu + 3\sigma]$, gdzie σ oznacza odchylenie standardowe, μ wartość średnią dla rozpatrywanych danych. Dany przedział obejmuje ok. $99,73\%$ obserwacji analizowanej zmiennej. Pozostałe dane, które znajdują się poza tym przedziałem kwalifikuje się jako obserwacje odstające [16].
- **Metoda z-score** - definiuje jak daleko od wartości średniej znajduje się dana obserwacja w zmiennej [17]. Wskaźnik z dla danej obserwacji wyznacza się ze wzoru:

  $$z_i = \frac{x_i - \mu}{\sigma} \tag{4.4}$$

  gdzie, x_i to pojedyncza obserwacja danej zmiennej, μ to wartość średnia dla wszystkich danych zmiennej, σ to odchylenie standardowe dla wszystkich danych w zmiennej. Jeżeli wyznaczona wartość $z_i \notin [-3,3]$, to taką obserwację przyjmuje się jako odstającą.

W przypadku jeśli rozkład danej zmiennej nie ma rozkładu normalnego lub nie jest do niego zbliżona, można zastosować metody, które nie są wrażliwe na rodzaj rozkładu. Takimi metodami są:

- **Zmodyfikowana metoda z-score** - jest modyfikacją standardowej metody z-score w której stosuje się medianę $\tilde{X}$ dla danej zmiennej, zamiast wartości średniej μ oraz zamiast odchylenia σ medianowe odchylenie bezwzględne (*ang. Mean absolute deviation*). Wskaźnik zmodyfikowanego z^* oblicza się ze wzoru:

$$z_i^* = \frac{x_i - \tilde{X}}{1.4826 \cdot MAD} \tag{4.5}$$

gdzie, x_i oznacza wartość dla i-tej obserwacji, $\tilde{X}$ medianę dla danej zmiennej X, MAD medianowe odchylenie bezwzględne dane wzorem $\text{MAD} = M(|x_i - \tilde{X}|)$, gdzie M oznacza medianę [18].

- **Rozstęp międzykwartylowy IQR** *(ang. interquartile range)* - jest to metoda pozwalająca na ocenę rozpiętości danych dla pojedynczej cechy. Polega na wyznaczeniu odległości między pierwszym kwartylem, który określa, że 25% obserwacji danej zmiennej znajduje się poniżej wartości kwartyla pierwszego, a kwartylem trzecim, stanowiącym wartość poniżej której znajduje się 75% obserwacji. Taką wartość określa się jako rozstęp ćwiartkowy (międzykwartylowy) *IQR*.

$$IQR = q_3 - q_1 \tag{4.6}$$

gdzie, q1 - kwartyl pierwszy, q3 - kwartyl trzeci.

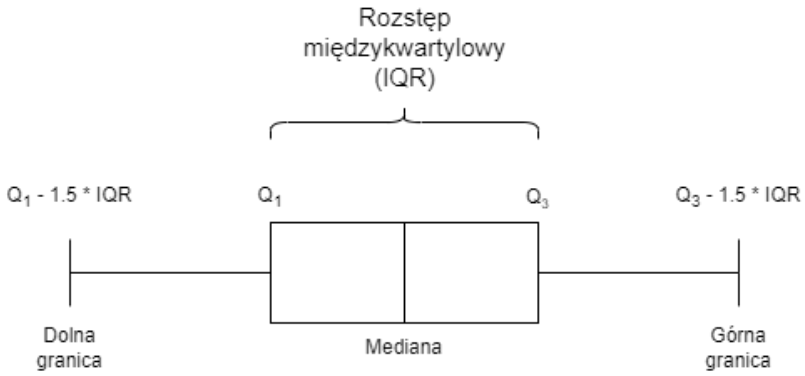
Wszystkie obserwacje, które znajdują się poniżej dolnej granicy wyrażonej wzorem:

$$\text{dolna} = q_1 - (1.5 \cdot IQR) \tag{4.7}$$

oraz powyżej górnej granicy określonej:

$$\text{gorna} = q_3 + (1.5 \cdot IQR) \tag{4.8}$$

są uznane za obserwacje odstające [19]. Wizualne przedstawienie metody *IQR* prezentuje **Rys. 4.2.**.



Rys. 4.2. Wizualizacja metody IQR

W przypadku obsługi wartości odstających wyróżnia się trzy główne podejścia:

- **zachowanie obserwacji** - w sytuacji, kiedy w toku analizy wiadomo, że te konkretne obserwacje nie są wynikiem błędu i są istotne z punktu widzenia procesu tworzenia modelu
- **poprawienie błędów** - w sytuacji, kiedy przypuszczalnie jasne jest, jaki prawidłowy zakres wartości może przyjąć dana obserwacja
- **usunięcie obserwacji** - w sytuacji, kiedy nie jest możliwe ustalenie prawidłowej wartości lub gdy dana obserwacja nie jest istotna z punktu widzenia analizy

4.3.3 Obsługa danych kategoryzujących

Oprócz prezentowania danych w sposób ilościowy, które reprezentują różne miary liczbowe wykorzystuje się typ danych przeznaczony do przedstawienia danych przy pomocy pewnych kategorii lub klas. Takie dane określa się jako dane kategoryzujące w obrębie których wyróżnia się dwa główne rodzaje. Pierwszym z nich są tzw. dane nominalne, które charakteryzują się brakiem występowania wewnętrznej hierarchii, czy też kolejności [15]. Przykładem zmiennych nominalnych są:

- **płeć** (mężczyzna, kobiera)
- **kolor włosów** (blond, brązowy, czarny)
- **typ mieszkania** (dom, mieszkanie, apartament)
- **narodowość** (polska, niemiecka, francuska)

Drugim rodzajem są cechy porządkowe, które jednoznacznie i naturalnie określają pewną kolejność. Przykładem takich cech mogą być:

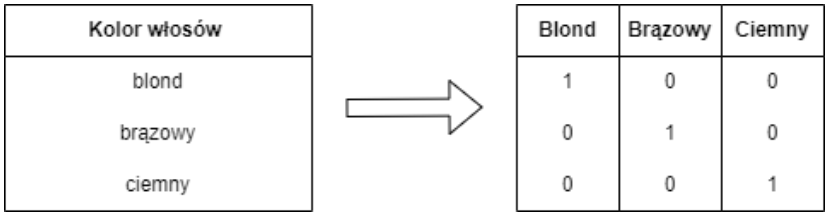
- **stopień naukowy** (licencjat, magister, doktor)

- **poziom języka** (podstawowy, średniozaawansowany, zaawansowany)
- **poziom dochodu** (niski, średni, wysoki)

Aby taki typ danych można było zastosować w analizie regresji, należy je odpowiednio przygotować. Proces ten nazywamy kodowaniem cech, a polega on na przedstawieniu etykiet takich cech, czy to nominalnych, czy też porządkowych za pomocą liczb.

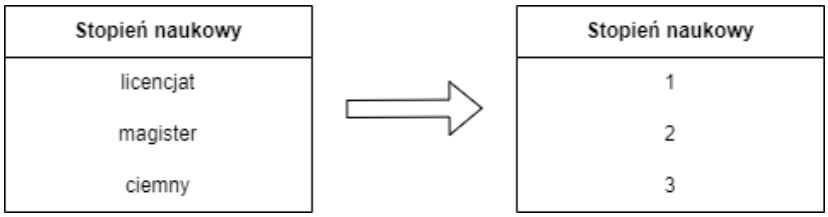
Istnieje wiele metod kodowania cech kategoryzujących, które stosuje się w zależności od konkretnego przypadku. Oto dwie metody kodowań:

- **Kodowanie "gorącojedynkowe"** *(ang. one-hot encoding)* - polega na stworzeniu dla każdej unikatowej wartości cechy nominalnej odrębnej zmiennej (kolumny). Do danej kategorii przypisuje się wartość 1 zgodnie z występowaniem wartości w rekordzie. Pozostałe kategorie przyjmują wartość 0. Przykład takiego rodzaju kodowania prezentuje **Rys. 4.3.**



Rys. 4.3. Przykład kodowania "gorącojedynkowego"

- **Kodowanie porządkowe** *(ang. ordinal encoding)* - jest to kodowanie cech porządkowych polegające na przypisaniu kolejnym etykietom kolejnych liczb zgodnie z kolejnością. Przykład takiego rodzaju kodowania prezentuje **Rys. 4.4.**



Rys. 4.4. Przykład kodowania porządkowego

4.3.4 Skalowanie zmiennych liczbowych

W przypadku danych, które służą do wytrenowania modelu powszechnie występuje zjawisko, w którym poszczególne zmienne mają inne skale liczbowe. Niektóre z cech mogą zawierać bardzo małe liczby, podczas gdy inne bardzo duże. Niezrównoważony zasięg cech może mieć poważny wpływ na jakość wytrenowanego modelu. Może to się objawiać np.

zniekształcaniem otrzymywanych wyników, kiedy jedna z cech zawierająca duże liczby dominuje nad innymi cechami, które powinny mieć istotny wpływ na wynik, ale są przez model w pewien sposób lekceważone. Innym przypadkiem może być niska jakość modelu, który napotka trudności w nauczeniu się wzorców w danych. Jednym z rozwiązań, które może okazać się pomocne w rozwiązaniu tego problemu jest skalowanie cech.

Skalowanie cech jest procesem, które opiera się na takim zmodyfikowaniu zmiennych niezależnych, aby ujednolicić i zminimalizować różnicę w zasięgu tychże zmiennych. Ze stosowanych metod do skalowania cech szerokie i częste zastosowanie mają normalizacja i standaryzacja [20].

Normalizacja jest metodą , która zmienia wartości cech na zasięg od 0 do 1. Formuła normalizacji prezentuje się następująco:

$$x_i^{'} = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{4.9}$$

gdzie, x_i to wartość oryginalna i-tej obserwacji, min(x) to wartość minimalna dla danej cechy, max(x) wartość maksymalna danej cechy.

Powyższy wzór pokazuje, że główne założenie normalizacji opiera się na skalowaniu przy użyciu wartości krańcowych tj. wartości maksymalnych i minimalnych dla danej cechy. Dlatego też spotykana jest zamienna nazwa normalizacji znana jako skalowanie min-max.

Innym podejście do skalowania cech jest zastosowanie techniki standaryzacji. Standaryzacja jest metodą, która sprawia, że przeskalowane cechy posiadają wartość średnią μ równą 0 oraz odchylenie standardowe σ równe 1. Formułą standaryzacji jest:

$$x_i^{'} = \frac{x_i - \mu}{\sigma} \tag{4.10}$$

gdzie, x_i to wartość oryginalna i-tej obserwacji, μ to wartość średnia dla danej cechy, σ to odchylenie standardowe danej cechy.

4.4 Wybór i ocena modelu

Sercem każdego problemu predykcyjnego jest wybór odpowiedniego modelu, który pozwoli na uzyskanie satysfakcjonującego rozwiązania. Ważne, aby taki model łączył w sobie zarówno dokładność i elastyczność, w celu zachowania zdolności do dokonywania możliwie najbardziej precyzyjnych przewidywań przy zmieniających się warunkach i danych wejściowych. Problem predykcji opiera się na dwóch zasadniczych filarach,

którymi są wybór modelu oraz ocena modelu. Obydwa etapy wzajemnie się uzupełniają i są równie ważne w całym procesie budowy modelu predykcyjnego.

Etap wyboru modelu to proces, w którym ocenia się wiele różnych modeli w celu wybrania najlepszego dla analizowanego problemu. W procesie tym zwykle dokonuje się wyboru spośród wielu dostępnych algorytmów oraz parametrów tych algorytmów, również zwanymi hiperparametrami. Taka złożona analiza pozwala na wyselekcjonowanie optymalnego modelu, który będzie odpowiedni dla danego problemu.

Ocena modelu po etapie wyboru polega na sprawdzeniu, jak model, który został wybrany jako najlepszy radzi sobie z nowymi danymi i pozwala ustalić, czy spełnia oczekiwania co do swojej dokładności predykcyjnej.

4.4.1 Rodzaje zestawów danych

Aby skutecznie przeprowadzić proces wyboru i oceny modelu, konieczne jest podzielenie danych w odpowiedni sposób. Trenowanie modelu oraz późniejsza jego weryfikacja na tym samym zestawie danych nie pozwalają stwierdzić, czy dany model działa prawidłowo, albo czy jego predykcje są wystarczająco dokładne. Aby rozwiązać ten problem w uczeniu maszynowym stosuje się podział danych na trzy podstawowe grupy. Są nimi [21]:

- **grupa ucząca** (grupa treningowa)
- **grupa walidacyjna**
- **grupa testująca**

Grupa ucząca, jak wskazuje na to nazwa, używana jest do trenowania i dostosowywania modelu do charakterystyki analizowanych danych. Grupa ta stanowi zazwyczaj większą część całości danych, aby podczas treningu model miał styczność z jak największą liczbą przypadków, jaka jest możliwa.

Grupa walidacyjna jest kluczowa przy procesie wyboru modelu. To właśnie na podstawie tej grupy wybiera się najlepszy algorytm wraz z rozpatrywanymi parametrami.

Grupa testowa jest podzbiorem danych, który nie jest używany w procesie wyboru modelu. Używa się tych danych dopiero przy ocenie końcowej wydajności wybranego wcześniej modelu. Testowanie modelu na danych, które nie były widziane wcześniej podczas trenowania i dopasowywania hiperparametrów pozwala na stwierdzenie, czy model spełnia swoje zadanie.

Proporcje podziału danych nie są jednoznaczne i nie ma jednej generalnej zasady, która

określa ile obserwacji powinno się znaleźć w którymś z zestawów. Proporcja zależy w dużym stopniu od ilości danych, które posiadamy i od charakterystyki analizowanego problemu. Proporcje należy dobierać w taki sposób, aby model miał wystarczająco obszerny zestaw treningowy, który pozwoli na możliwie najszersze poznanie danych, ale jednocześnie wystarczająco duże grupy walidacyjne i testujące, które pozwolą na możliwie najbardziej dokładne i obszerne testy.

Typowymi podziałami mogą być [22]:

- 50% dla zestawu uczącego, 25% dla zestawu walidacyjnego i testującego
- 60% dla zestawu uczącego, 20% dla zestawu walidacyjnego i testującego
- 70% dla zestawu uczącego, 15% dla zestawu walidacyjnego i testującego

Warto pamiętać, że do trenowania finalnego modelu dane uczące i walidacyjne, które służyły do trenowania i wyboru najlepszego modelu łączy się w jeden zbiór uczący. Natomiast wcześniejszy zestaw testujący, oczywiście testuje tenże finalny model. W praktyce więc najpierw dzieli się dane na dwa podzbiory uczący oraz testujący. Następnie do wyboru najlepszego modelu dzieli się dane uczące na dwa kolejne podzbiory uczący i walidacyjny.

4.4.2 Metody podziału danych

Metody podziału danych można scharakteryzować, jako techniki podziału danych na trzy podgrupy w celu jak najdokładniejszego podejścia do wyboru, czy też oceny modelu. Dobór konkretnej metody uzależniony jest w dużym stopniu od jakości i rozmiaru zestawu danych, jaki jest użyty w procesie budowy modelu.

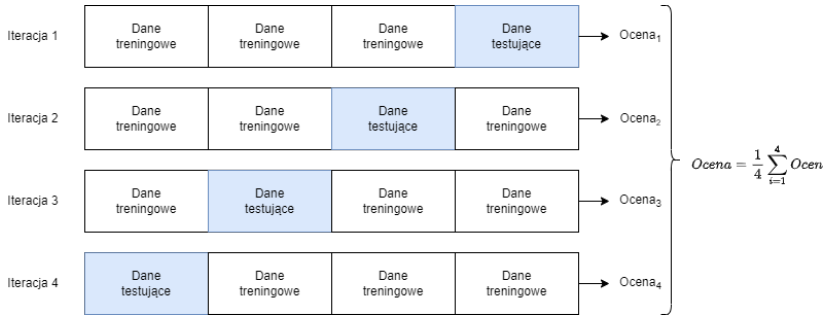
Oto zestawienie kilku najczęściej stosowanych podejść do podziału danych [22]:

- **wydzielanie** *(ang. hold-out)* - metoda ta polega na losowym podziale zestawu danych na odpowiednie podzbiory. Jest to najprostsza i najczęściej stosowana metoda podziału danych. Jej wizualizacje prezentuje **Rys. 4.5.**

Rys. 4.5. Wizualizacja metody wydzielenia

- **k-krotne sprawdzenie krzyżowa** *(ang. k-fold cross-validation)* - metoda ta polega na k-krotnym losowym podziale danych na k równe podzbiory z których jeden stanowi podzbiór testujący/walidacyjny, a pozostałe stanowią zestaw danych uczących. Proces

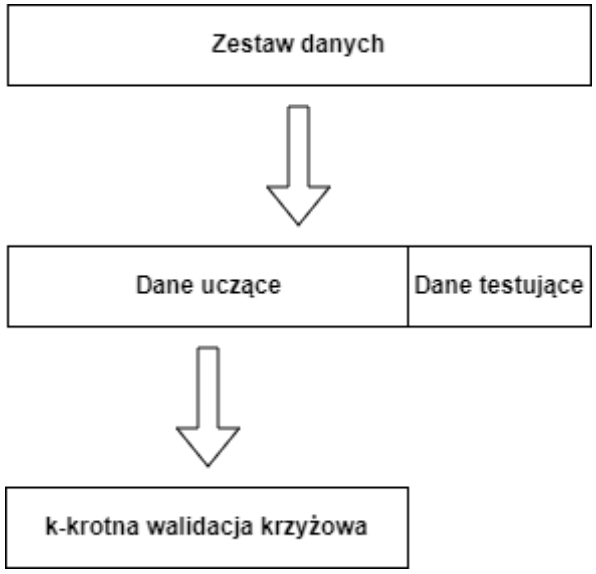
ten powtarzany jest k-krotnie, po czym skuteczność modelu określa się przy pomocy średniej ze wszystkich iteracji. Graficzne przedstawienie metody prezentuje **Rys. 4.6.**



Rys. 4.6. Wizualizacja k-krotnego sprawdzenia krzyżowego

- **walidacja krzyżowa z jednym wydzielonym przykładem** *(ang. leave-one-out cross validation)* - metoda ta jest przypadkiem k-krotnej walidacji krzyżowej, gdzie k jest równe n liczbie obserwacji w zestawie danych. Dzieli się więc dane na n podzbiorów z czego n - 1 stanowi dane uczące, a jedna obserwacja stanowi zestaw testujący.

Możliwe jest stosowanie mieszanych metody podziału danych. Bardzo częstym przykładem takiego zastosowania jest wstępne podzielenie zestawu danych na zestaw uczący i testowy, a następnie zastosowanie k-krotnej walidacji krzyżowej na grupie uczącej do etapu wyboru modelu. Prezentuje to **Rys. 4.7.**



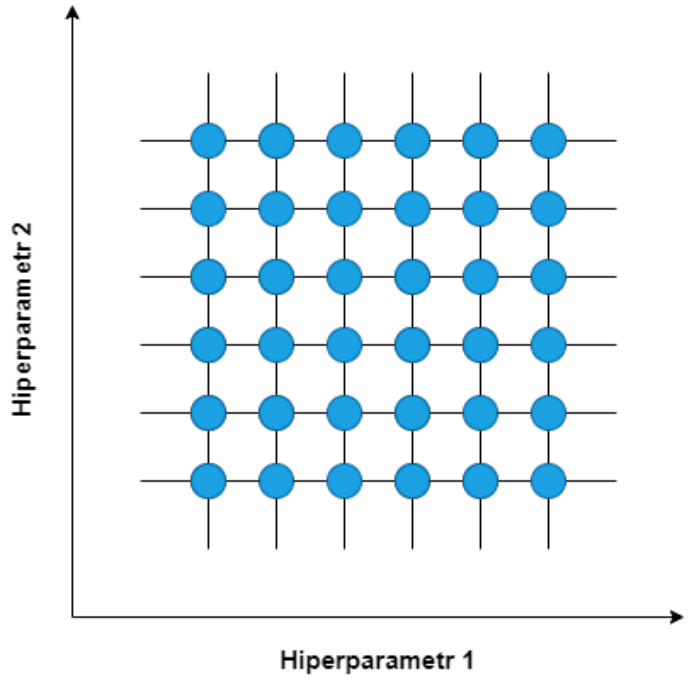
Rys. 4.7. Wizualizacja mieszanej metody wydzielenia oraz k-krotnej walidacji krzyżowej

4.4.3 Selekcja hiperparametrów

Hiperparametrami algorytmów uczenia maszynowego nazywa się parametry, które zostają wybrane przed wdrożeniem procesu uczenia danego modelu. Przykładami hiperaparametrów mogą być zarówno liczba ukrytych warstw w sieci neuronowej, jak i głębokość drzewa decyzyjnego. Wybór odpowiednich wartości tych parametrów ma bezpośredni wpływ na stworzenie efektywnego modelu predykcyjnego.

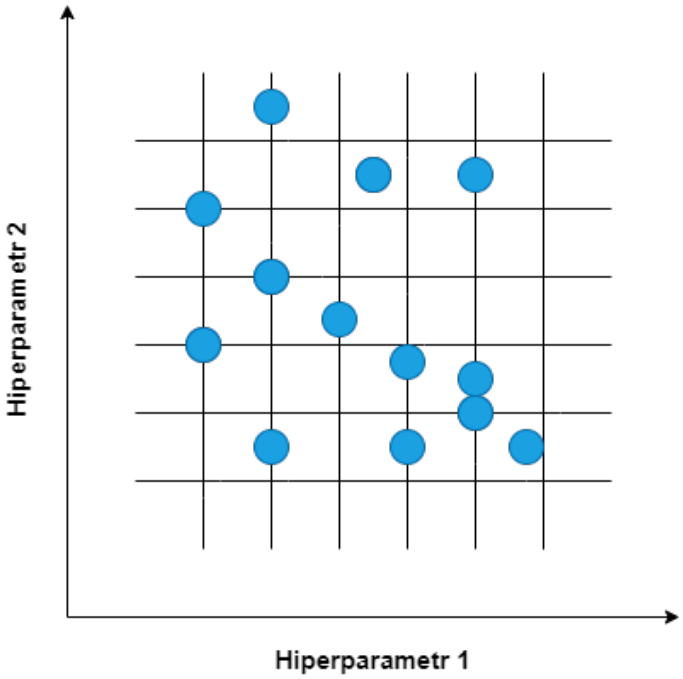
Do doboru hiperparametrów dla danego algorytmu stosuje się różne metody. Najbardziej powszechne są [23]:

- **Metoda siatki** *(ang. Grid Search)* - polega na sprawdzeniu wszystkich kombinacji zdefiniowanych wartości hiperparametrów. Wszystkie kombinacje wartości tworzą siatkę, której metoda zawdzięcza swoją nazwę. Wizualne przedstawienie metody prezentuje **Rys. 4.8.**



Rys. 4.8. Prezentacja metody siatki dla dwóch hiperparametrów

- **Metoda przeszukiwania losowego** *(ang. Random Search)* - polega na wyborze wartości hiperparametrów w sposób losowy spośród określonych zakresów. Wizualne przedstawienie metody prezentuje **Rys. 4.9.**



Rys. 4.9. Prezentacja metody przeszukiwania losowego dla dwóch hiperparametrów

- **Metoda optymalizacji bayesowskiej** *(ang. Bayesian optimization)* - polega na wykorzystaniu modelu probabilistycznego do znalezienia optymalnych wartości hiperparametrów.

4.4.4 Metryki

W analizie regresji błąd pojedynczej obserwacji to różnica między wartością przewidzianą przez model, a rzeczywistą wartością danej obserwacji i określa się go jako błąd predykcji (*ang. residual*).

$$b_i = y_i - \hat{y}_i \tag{4.11}$$

gdzie, b_i to błąd, y_i rzeczywista wartość, $\hat{y}_i$ wartość przewidywana dla i-tej obserwacji.

W celu określenia jakości modelu regresji stosuje się różne metryki, które pozwalają ocenić model na bazie błędów predykcji dla danego zbioru obserwacji. Podstawowymi technikami są suma kwadratów błędów *SSR*, błąd średniokwadratowy *MSE*, średni błąd bezwzględny *MAE*, pierwiastek średniego błędu kwadratowego *RMSE*, współczynnik determinacji R^2 [24].

Suma kwadratów błędów *(ang. Sum of Squared Residuals, SSR)* jest to suma kwadratów błędów predykcji. Im mniejsza wartość metryki, tym większa dokładność modelu.

$$SSR = \sum_{i=1}^{n} (y - \hat{y}_i)^2 \tag{4.12}$$

gdzie, n to liczba obserwacji, y_i rzeczywista wartość, $\hat{y}_i$ wartość przewidywana dla i-tej obserwacji.

Błąd średniokwadratowy *(ang. Mean Squared Error, MSE)* to średnia kwadratów różnicy błędów między wartościami rzeczywistymi, a prognozowanymi. Im większa wartość błędu, tym mniejsza skuteczność modelu predykcyjnego.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.13}$$

Średni błąd bezwzględny *(ang. Mean Absolute Error, MAE)* jest średnią bezwzględną wartością różnic między prawdziwymi wartościami, a przewidywanymi. Im wartość błędu mniejsza, tym bardziej precyzyjne predykcje modelu.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.14}$$

Pierwiastek średniego błędu kwadratowego *(ang. Root Mean Squared Error - RMSE)*

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.15}$$

Współczynnik determinacji lub określoności *(ang. R^2 score)* określa jaki procent zmienności, inaczej wariancji zmiennej zależnej jest określany poprzez zmienne niezależne. Innymi słowy współczynnik ten określa, jak dobrze model potrafi przewidzieć zmienną zależną na podstawie zmiennych niezależnych. R^2 przyjmuje wartość od 0 do 1. Im bliżej 1, tym model jest skuteczniejszy.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.16}$$

gdzie, n to liczba obserwacji, y_i to wartość rzeczywista zmiennej zależnej dla, $\hat{y}_i$ to wartość przewidywana zmiennej zależnej dla i-tej obserwacji, $\bar{y}$ to średnia wartość zmiennej zależnej.

4.4.5 Kompromis między obciążeniem, a wariancją modelu

W modelach sztucznej inteligencji wyróżnia się trzy podstawowe błędy [25, 26]:

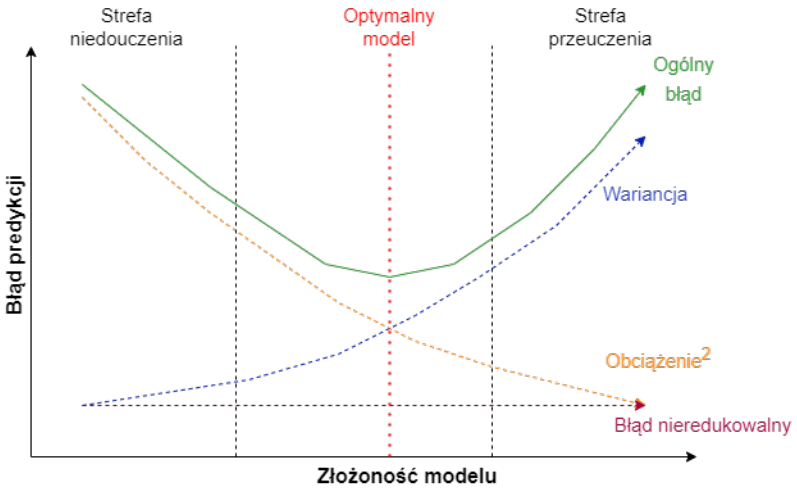
- **błąd obciążenia** *(ang. Bias)*
- **błąd wariancji** *(ang. Variance)*
- **błąd nieredukowalny** *(ang. Unreducible error)*

Obciążenia odnosi się do niezdolności modelu do uchwycenia prawdziwych zależności występujących w danych. Błąd ten odzwierciedla więc jak dokładnie dany model dopasowany jest do danych.

Wariancję modelu uczenia maszynowego definiuje się jako zdolności predykcyjne modelu w odniesieniu do różnych zestawów danych. Obrazuje więc różnicę predykcji dla różnych danych wejściowych.

Błąd nieredukowalny wynika z zaszumienia danych. Błąd ten w żaden sposób nie może być zredukowany na etapie budowy modelu. Jedynym sposobem na redukcję tego typu błędu jest odpowiednie przygotowanie danych.

Optymalny model predykcyjny powinien łączyć w sobie zarówno wysoką dokładność przewidywanych wartości, jak i odpowiedni stopień generalizacji. Problem ten określa się jako kompromis pomiędzy obciążeniem, a wariancją *(ang. Bias-Variance tradeoff)*. W przypadku zbyt dokładnego dopasowania do danych treningowych model posiada dużą złożoność, co zmniejsza zdolności uogólniania. Taki model posiadający niskie obciążenie oraz wysoką wariancję określa się mianem zjawiska przeuczenia *(ang. overfitting)*. Predykcje modelu przeuczonego są bardzo dokładne na danych treningowych, a na danych testowych dokładność bardzo wyraźnie spada. Przeciwnym zjawiskiem do przeuczenia jest tzw. niedouczenie *(ang. underfitting)*. Niedouczone modele posiadają wysoką wartość obciążenia oraz niską wariancję. Zazwyczaj są to modele proste, które nie uchwyciły zależności występującej w danych. Głównym założeniem tego kompromisu jest więc znalezienie złotego środka, który zapewni stosunkowo niskie obciążenie oraz niską wariancję. Wizualne przedstawienie kompromisu prezentuje **Rys. 4.10.**



Rys. 4.10. Wykres kompromisu między obciążeniem i wariancją poprzez zestawienie złożoności modelu z błędem predykcyjnym

Kompromis ten można zapisać za pomocą wzoru:

$$\text{Ogólny błąd predykcji modelu} = \text{Obciążenie}^2 + \text{Wariancja} + \text{Błąd nieredukowalny} \quad (4.17)$$

4.5 Algorytmy uczenia maszynowego wykorzystywane w analizie regresji

W ramach modeli regresji predykcje przedstawia się poprzez funkcja $E(Y|X)$ obrazującą warunkową wartość oczekiwaną zmiennej zależnej Y pod warunkiem zmiennych niezależnych $X = \{X_1, X_2, ..., X_p\}$ [27]. Ich głównym zadaniem więc jest możliwie najwierniejsze oddanie relacji występujących między zmiennymi. Aby jednak było to możliwe dobiera się odpowiednie metody numeryczne, a inaczej algorytmy, które w ramach analizowanego problemu są w stanie wspomniane relacje między wektorem zmiennych niezależnych X, a zmiennej zależnej Y zamodelować.

W dziedzinie analizy regresji istnieje wiele podejść, które stosuje się w zależności od danych, złożoności analizowanego problemu, czy też celu analizy. Podejścia te mogą się różnić zarówno skomplikowaniem, czy złożonością, jak i samą metodologią oraz podejściem do rozwiązania problemu. W tym rozdziale zostało zaprezentowane kilka takich zróżnicowanych pod względem złożoności i podstawowych założeń metod.

4.5.1 Regresja liniowa

Regresja liniowa jest jednym z podstawowych algorytmów wykorzystywanych do tworzenia modeli regresyjnych [28]. Celem regresji liniowej jest odnalezienie liniowej zależności między zmiennymi niezależnymi X, a zmienną zależną y. Im taka zależność jest oddana bardziej dokładnie, tym model regresji prognozuje bardziej dokładne wyniki. Formuła na uogólnioną postać regresji jest następująca:

$$\hat{\mathbf{y}} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \quad (4.18)$$

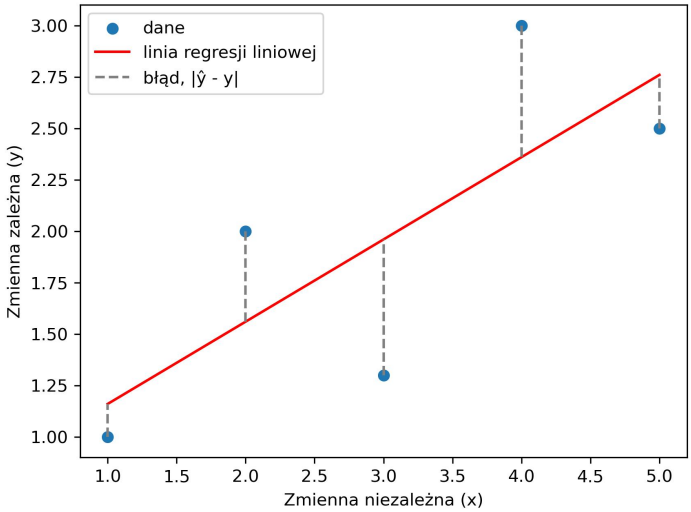
gdzie, $\hat{\mathbf{y}}$ oznacza prognozowany wynik, x_i (dla $i = 1, 2, ..., n$) oznacza kolejne zmienne niezależne, β_0 i β_i (dla $i = 1, 2, ..., n$) są współczynnikami regresji.

Założeniem Regresji liniowej jest znalezienie takiej funkcji liniowej, aby możliwie najdokładniej była dopasowana do danych. W tym celu dobiera się możliwie najlepsze współczynniki regresji. Aby zobrazować ten proces, został zaprezentowany przykład dla

danych z jedną zmienną niezależną. Formuła regresji liniowej dla takiej funkcji przedstawia się nastepująco:

$$\hat{y} = \alpha + \beta x \tag{4.19}$$

gdzie, α oznacza wyraz wolny, a β współczynnik regresji dla zmiennej x. **Rys. 4.11.** przedstawia schemat tego modelu regresji liniowej dla powyższej funkcji.



Rys. 4.11. Schemat modelu regresji liniowej dla danych zawierających jedną zmienną niezależną

Jedną z metod optymalizacji współczynników regresji jest metoda najmniejszych kwadratów *(ang. least squares method)*. Metoda ta zakłada minimalizację sumy kwadratów różnic wartości przewidzianej przez model $\hat{y}$ oraz rzeczywistej wartości y. Sumę tę określa się jako suma kwadratów błędów i przedstawia się:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \alpha - \beta x)^2 = min \tag{4.20}$$

gdzie $\hat{y}_i$ oznacza wartość prognozowana dla i-tej obserwacji, y_i oznacza rzeczywistą wartość dla i-tej obserwacji, n oznacza liczbę obserwacji.

Do wyznaczenia wartości minimalnej funkcji sum kwadratów błędów wykorzystuje się operację różniczkowania. Wyznacza się różniczkę dla każdego z parametrów funkcji i przyrównuje do 0.

$$\begin{cases} \dfrac{\partial}{\partial \alpha}\displaystyle\sum_{i=1}^{n}(y_i - \alpha - \beta x)^2 = \sum_{i=1}^{n} 2(y_i - \alpha - \beta x)(-1) = 0 \\ \dfrac{\partial}{\partial \beta}\displaystyle\sum_{i=1}^{n}(y_i - \alpha - \beta x)^2 = \sum_{i=1}^{n} 2(y_i - \alpha - \beta x)(-x) = 0 \end{cases} \tag{4.21}$$

Rozwiązując powyższy układ otrzymuje się wzory na współczynniki funkcji liniowej, które zminimalizują sumę kwadratów błędów.

4.5.2 Algorytm k-najbliższych sąsiadów

Kolejnym algorytmem znajdującym zastosowanie w problemach regresji jest metoda K-najbliższych sąsiadów *(ang. k-Nearest Neighbors, kNN)* [29]. Algorytm K-najbliższych sąsiadów przewiduje wartość zmiennej zależnej y dla nowego punktu w n-wymiarowej przestrzeni na bazie k-punktów, które znajdują się w jego najbliższym otoczeniu o znanej wartości zmiennych wyjściowych. Szacując wynik zmiennej zależnej nowego punktu jednym z podejść jest wyznaczenie średniej ze zmiennych zależnych k-rozpatrywanych sąsiadów. Taka formuła wygląda następująco:

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{4.22}$$

gdzie, k to liczba sąsiadów, a y_i to wartość zmiennej zależnej dla i-tego sąsiada.

Do określenia k-sąsiadów dla nowej obserwacji służy miara odległości. Najczęściej wykorzystywanymi metodami pomiaru odległości w algorytmie K-najbliższych sąsiadów są:

- **Odległość euklidesa**:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4.23}$$

  gdzie, x_i i y_i to dwie obserwacje między którymi liczona jest odległość, a n to liczba zmiennych (cech) dla każdej obserwacji.

- **Odległość miejska** (Manhattan):

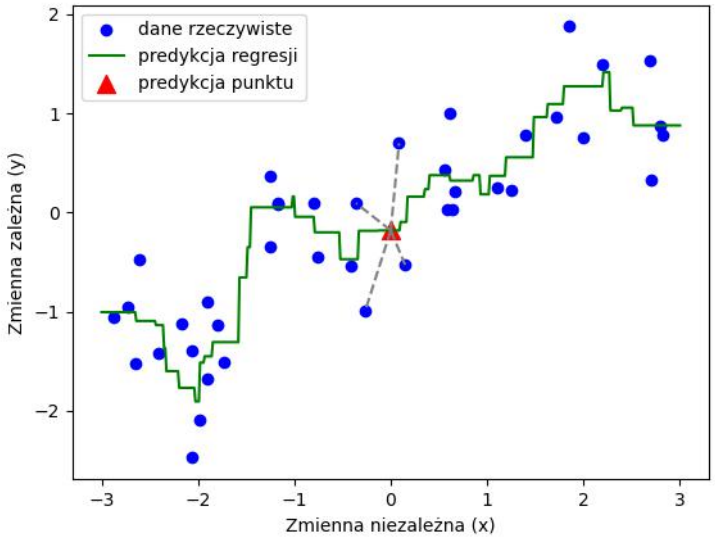
$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.24}$$

- **Odległość minkowskiego**:

$$d(x,y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{4.25}$$

  gdzie, p jest hiperparametrem. Dla $p = 2$ mamy odległość euklidesową, a przy $p = 1$ odległość miejską. Odległość minkowskiego można więc traktować jako uogólnioną miarę odległości.

Oprócz doboru metody określania najbliższych sąsiadów na podstawie pomiaru odległości ważnym elementem jest dobór ich ilości. Parametr k bezpośrednio przekłada się na skuteczność modelu, dlatego też jego odpowiedni dobór wymaga szczegółowej analizie. Jednak gdy zarówno parametr k zostanie wybrany oraz zostanie określona metoda pomiaru odległości dla najbliższych punktów można przystąpić do budowy modelu. Schemat prognozy w algorytmie K-najbliższych sąsiadów prezentuje **Rys. 4.12.**.



Rys. 4.12. Schemat modelu K-najbliższych sąsiadów dla k = 4 oraz dla jednej zmiennej niezależnej

Powyżej została zaprezentowana wersja algorytmu K-najbliższych sąsiadów w przypadku, kiedy każdy z sąsiadów jest równie istotny, tzn. każdy z sąsiadów posiada jednakową wagę równą 1. Możliwa jest jednak pewna modyfikacja. Jedna z metod polega na zastosowaniu wag dla sąsiadów, które stanowią odwrotność ich odległości od nowej obserwacji. Dodanie takich wag pozwala na zwiększenie wpływu sąsiadów, znajdujących się bliżej nowego punktu. Po zmodyfikowaniu formuła predykcyjna algorytmu K-najbliższych sąsiadów z wagami równa się [30]:

$$\hat{y} = \frac{\sum_{i=1}^{k} w_i y_i}{\sum_{i=1}^{k} w_i} \tag{4.26}$$

gdzie, w_i to waga dla i-tego najbliższego sąsiada. Przykładowo dla odległości euklidesa waga dla i-tego sąsiada może być wyznaczona na podstawie wzoru:

$$w = \frac{1}{d(x,y)} \tag{4.27}$$

gdzie, $d(x,y)$ oznacza odległość między obserwacjami x i y.

4.5.3 Drzewa decyzyjne

Algorytmy oparte na drzewach są szeroko stosowanymi metodami w uczeniu maszynowym. Podstawą tego algorytmu jest drzewo decyzyjne, które składa się z takich elementów jak węzły, gałęzie oraz liście. Na szczycie każdego drzewa znajduje się węzeł początkowy, który nazywany jest korzeniem drzewa. Zarówno korzeń, jak i kolejne węzły na podstawie reguły decyzyjnej zostają podzielone na dwa podzbiory. Wynik każdej z reguł reprezentowany jest przez gałąź, która prowadzi do kolejnych węzłów lub do liścia. Konstrukcja drzewa przebiega więc iteracyjnie i każdy z węzłów dzieli dane na kolejne dwa podzbiory. Elementami kończącymi drzewo są liście. Stanowią one węzły końcowe, które nie zawierają w sobie reguły decyzyjnej opartej na testach, ale przewidzianą wartość określaną na podstawie średniej z próbek danych znajdujących się w liściu.

Wspomniane wcześniej reguły zgodnie z którymi następuje podział danych w węzłach formułowane są na podstawie testów. Rodzaj przyjętego testu zależny jest od rodzaju zmiennej niezależnej [31].

- **Dla zmiennych nominalnych:**
  - **testy tożsamościowe**
    wyniki testu są tożsame z wartościami zmiennej
  - **testy równościowe**
    $$t(x) = \begin{cases} 1, & \text{jeśli } a(x) = v \\ 0, & \text{jeśli } a(x) \neq v \end{cases}$$
    gdzie, a oznacza daną zmienną, x wartość zmiennej, a v wartość kryterium danej zmiennej
  - **testy przynależności**
    $$t(x) = \begin{cases} 1, & \text{jeśli } a(x) \in V \\ 0, & \text{jeśli } a(x) \notin V \end{cases}$$
    gdzie, V jest podzbiorem atrybutu a
- **Dla atrybutów ciągłych:**

– **testy nierównościowe**

$$t(x) = \begin{cases} 1, & \text{jeśli } a(x) \geq v \\ 0, & \text{jeśli } a(x) < v \end{cases}$$

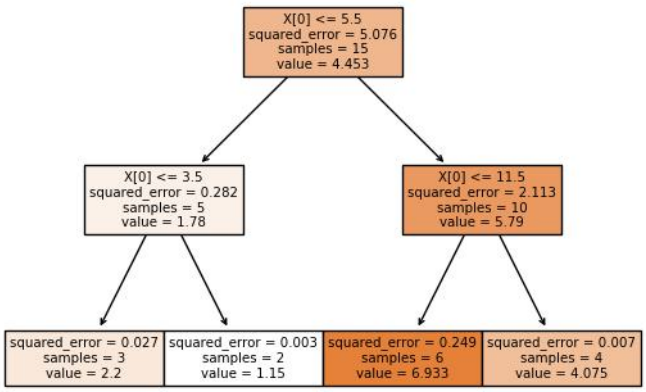
- **Dla atrybutów porządkowych:**
  - **testy nierównościowe**
  - **testy przynależnościowe**

Rozrastanie się drzewa, a inaczej mówiąc tworzenie kolejnych podziałów danych, odbywa się w taki sposób w przypadku analizy regresji, aby możliwie jak najbardziej zminimalizować błąd między przewidzianą wartością $\hat{y}$, a wartością rzeczywistą y. Błąd można wyznaczyć za pomocą kilku technik, a jedną z nich jest średni błąd kwadratowy:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.28}$$

gdzie, n to liczba próbek w zbiorze danych, y_i to rzeczywista wartość , a $\hat{y}_i$ to wartość predykcji dla i-tej próbki.

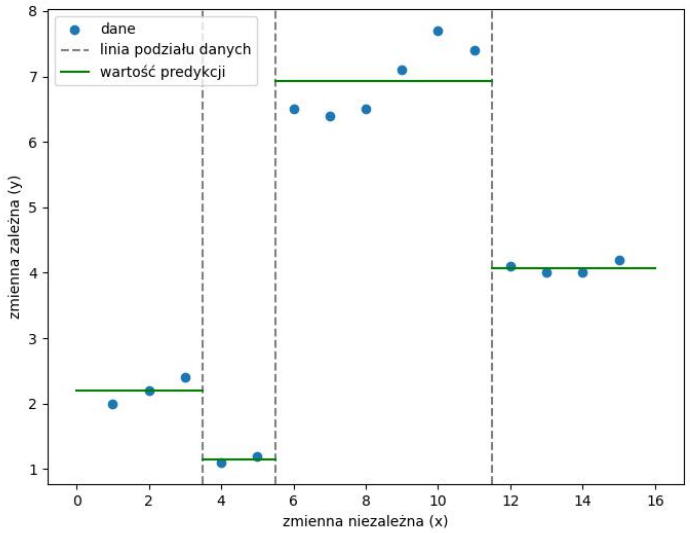
Graficzne przedstawienie drzewa decyzyjnego dla jednej zmiennej niezależnej prezentuje **Rys. 4.13.**



Rys. 4.13. Graficzne przedstawienie drzewa decyzyjnego dla jednej zmiennej niezależnej

Powyższy przykład drzewa można również przedstawić na wykresie zaprezentowanym na **Rys. 4.14.** Pionowe przerywane szare linie podziału danych reprezentują wartości reguł

decyzyjnych dla zmiennej niezależnej. Zielone poziome linie reprezentują wartości predykcji dla poszczególnych podzbiorów dla poszczególnych liści.



Rys. 4.14. Wykres drzewa decyzyjnego dla jednej zmiennej niezależnej (x) i zmiennej zależnej (y)

Rzadkim przypadkiem w drzewie decyzyjnym jest występowanie tylko jedna zmienna niezależna. W przypadku wielu zmiennych niezależnych przy konstrukcji drzewa ustala się, która ze zmiennych w danym węźle ma największy wpływ na minimalizację przyjętej techniki szacowania błędu. Tak sprawdza się każdą ze zmiennych przy kolejnych podziałach, aż zostaną utworzone liście drzewa.

W przypadku drzew decyzyjnych dostępne jest kilka technik optymalizacyjnych. Pierwszy sposób optymalizacji opiera się na ustawieniu warunków stopu, którymi mogą być:

- **maksymalna głębokość drzewa** - maksymalna liczba poziomów, które może osiągnąć drzewo. Na ostatnim z poziomów tworzone są liście.
- **minimalna liczba próbek w węźle** - jeżeli liczba próbek w danym węźle będzie mniejsza od liczby ustalonej jako granicznej nie nastąpi dalszy podział, a węzeł stanie się liściem.

Drugim podejściem jest tzw. przycinanie drzewa. Proces przycinania następuje dopiero po zbudowaniu całego drzewa. Polega na zastąpieniu kolejnych węzłów (poddrzew) liśćmi do momentu, aż liściem nie stanie się korzeń drzewa. Jednym z kryteriów pozwalający określić, które z drzew w procesie przycinania jest najbardziej efektywne dla

analizowanego problemu jest metoda minimalizacji drzewa z uwzględnieniem złożoności i kosztu *(ang. cost-complexity pruning)*. Dla kolejnych drzew wyznacza się parametr C_α *(ang. complexity parameter, tree score)*, który wyznacza się nastepująco [32]:

$$C_\alpha(T) = SSR(T) + \alpha \cdot |\widehat{T}| \tag{4.29}$$

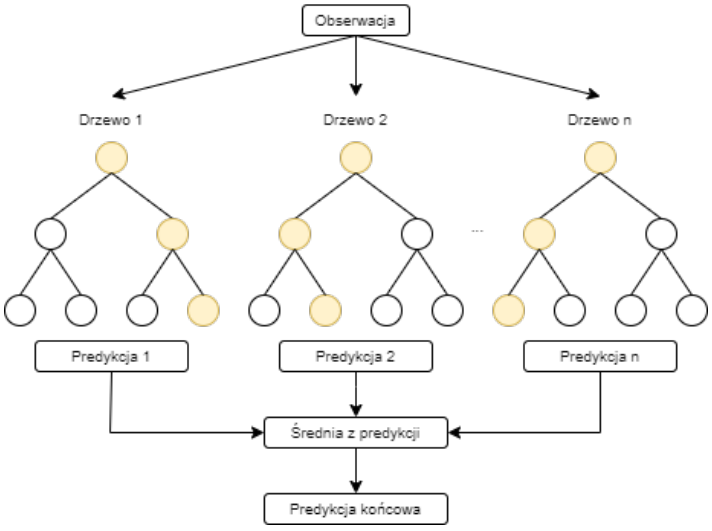
gdzie, $C_\alpha(T)$ oznacza wartość parametru, $SSR(T)$ sumę kwadratów różnic dla danego drzewa T, $|\widehat{T}|$ ilość liści w danym drzewie T, a α jest parametrem złożoności, który można odszukać z użyciem walidacji krzyżowej.

4.5.4 Las losowy

Metoda Lasu losowego jest kolejnym z popularnych algorytmów używanych przy analizie regresji. Las losowy jest metodą z grupy tzw. algorytmów zespołowych *(ang. ensemble algorithm)*, które łączą kilka modeli bazowych, aby uzyskać jeden, możliwie optymalny model predykcyjny. W przypadku Lasu losowego modelami bazowymi są drzewa decyzyjne, a finalna predykcja jest średnią arytmetyczną z predykcji wszystkich zbudowanych drzew. Model Lasu losowego można opisać przy pomocy formuły:

$$\hat{y} = \frac{1}{B}\sum_{i=1}^{B} T_i(x) \tag{4.30}$$

gdzie, $\hat{y}$ to wartość przewidywana przez model dla danych wejściowych x, B to liczba drzew w lesie losowym, a $T_i(x)$ to wartość przewidywana przez i-te drzewo dla danych wejściowych x. Ideę predykcji pojedynczej obserwacji lasu losowego przedstawia **Rys. 4.15.**



Rys. 4.15. Schemat ideowy lasu losowego.

Główne założenia algorytmu opierają się na modyfikacji metody bagging'u inaczej agregacja bootstrap'owej [27]. Bagging polega na wielokrotnym utworzeniu podzbiorów, na które składają się próby (obserwacje) z zestawu danych wybrane w sposób losowy ze zwracaniem w taki sposób aby nowo powstały zbiór był równy zbiorowi pierwotnemu. Przy tego typu wyborze obserwacje w nowym zestawie mogą wystąpić wielokrotnie, a niektóre mogą w ogóle nie wystąpić. Na podstawie każdego z utworzonych podzbiorów budowane jest drzewo decyzyjne zgodnie z klasycznym algorytmem. Modyfikacja metody bagging'u w lesie losowym ma miejsce na etapie konstrukcji drzewa. Przy budowie każdego węzła wybiera się w sposób losowy m zmiennych niezależnych ze wszystkich zmiennych niezależnych i na podstawie tego wyboru odpowiednio konstruowane są węzły w drzewie i następuje podział danych. Operację tę powtarza się dla każdego węzła do momentu aż nie zostanie osiągnięte minimum liczby próbek dla węzła n_{min}. W takim wypadku tworzone są liście. Domyślnie przyjmuje się, że podzbiór zmiennych wynosi $\lfloor p/3 \rfloor$, a minimalna liczba próbek dla węzła $n_{min} = 5$. Warto jednak obydwie wartości parametrów dostosować do analizowanego problemu.

Ważną cechą Lasu losowego jest możliwość wykorzystania obserwacji, które nie zostały użyte przy podziałach na podzbiory do tworzenia poszczególnych drzew, jako zestawy walidacyjne. Takie obserwacje określa się w języku angielskim jako *Out of Bag (oob)*. Proces takiej walidacji można znaleźć zastosowanie w sytuacjach, w których inne metody walidacji będą czasochłonne lub będą wymagać dużych zasobów obliczeniowych.

4.5.5 Wzmocnienie Gradientu

Wzmocnienie gradientu *(ang. Gradient Boosting)* jest techniką, której założeniem jest wykorzystanie wielu podstawowych modeli uczenia maszynowego, aby stworzyć jeden złożony model. Algorytm ten należy więc do grupy algorytmów zespołowych *(ang. ensemble algorithms)*, w którym każdy z podstawowych modeli jest drzewem decyzyjnym.

Algorytm Wzmocnienia gradientu wykorzystuje metodę tzw. wzmocnienia *(ang. boosting)* [27], która opiera się na sekwencyjnym konstruowaniu kolejnych modeli w taki sposób, aby poprawić rezultat uzyskany przez poprzedni model. W przypadku wzmocnienia gradientu tworzy się drzewo inicjujące, a następnie dodaje się kolejne drzewa zbudowane na podstawie błędów drzewa poprzedzającego. Dzięki dodawaniu kolejnych drzew skuteczność modelu ulega poprawie, a predykcje są bardziej dokładne.

Podchodząc do Wzmocnienia gradientu w sposób bardziej detaliczny algorytm rozpoczyna się od wyznaczenia stałej wartości na podstawie danych uczących, inaczej

mówiąc od budowy drzewa decyzyjnego składającego się z liścia. Wartość tą wyznacza się z wyrażenia:

$$F_0(x) = \text{argmin}\,\gamma \sum_{i=1}^{n} L(y_i, \gamma) \tag{4.31}$$

gdzie, $L(y_i, \gamma)$ oznacza funkcję straty między wartością prawdziwą y_i, a wartością przewidzianą γ. Optymalną wartość γ można uzyskać poprzez przyrównanie pochodnej z funkcji strat względem γ do 0.

Budowa kolejnych modeli rozpoczyna się od wyznaczenia błędów dla każdej z predykcji:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right] F(x)=F_{m-1}(x) \tag{4.32}$$

gdzie, m to numer iteracji algorytmu, r_{im} oznaczają błędy dla i-tej obserwacji, które są wyznaczane na podstawie gradientu (pochodnej) funkcji strat $L(y, F(x))$ względem $F(x)$, a $F_{m-1}(x)$ to model składający się z $m-1$ poprzednich modeli.

Mając wyznaczone wartości poszczególnych błędów r_{im} dla danej obserwacji, w następnym kroku można przystąpić do zbudowania drzewa na ich podstawie zgodnie z klasyczną konstrukcją drzewa decyzyjnego, to znaczy dla wszystkich zmiennych niezależnych. Otrzymane liście oznacza się przy pomocy indeksów R_{jm}, dla $j = 1 \ldots J_m$ gdzie, j oznacza numer kolejnego liścia, J stanowi liczbę liści danego drzewa, a m jest indeksem drzewa. W dalszym kroku wykorzystując odpowiednie indeksy końcowych węzłów dla drzewa wyznacza się predykcje na podstawie poniższej formuły:

$$\gamma_{jm} = argmin\gamma \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma) \tag{4.33}$$

gdzie, wartość predykcji γ_{jm} dla j-tego liścia drzewa m minimalizuje funkcję starty, $x_i \in R_{jm}$ oznacza, że brane są pod uwagę te obserwacje, które zawierają się w konkretnym liściu.

W ostatnim kroku uzupełnia się predykcję dodając nowo zbudowane drzewo. Prezentuje to poniższa formuła:

$$F_m(x) = F_{m-1}(x) + v \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm}) \tag{4.34}$$

gdzie, $F_{m-1}(x)$ to poprzednie predykcje, γ_{jm} to wartości predykykcji dla obecnego drzewa, dla którego poszczególne obserwacja x należą do danewgo węzła końcowego R_{jm}, a v oznacza współczynnik uczenia z zakresu $[0, 1]$, który określa wpływ jakie każde drzewo ma na końcową predykcję. Dodawanie kolejnych modeli kończy się w momencie, gdy nowe drzewa nie wpływają na poprawę jakości modelu.

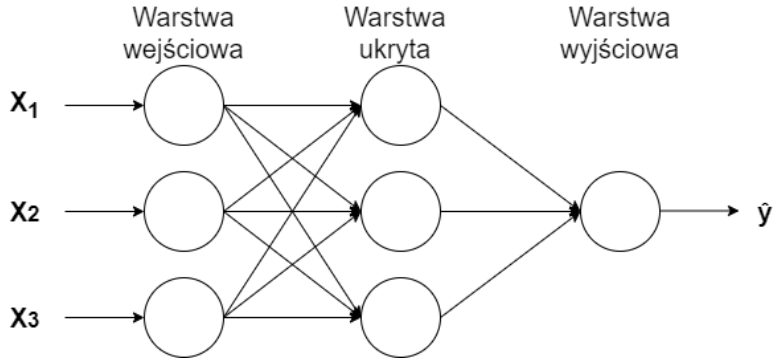
4.5.6 Sieci neuronowe

Inspiracją do stworzenia sieci neuronowych *(ang. Artificial neural network)* było odwzorowanie struktury neuronowej występującej w ludzkim mózgu [33]. Modele oparte na sieciach neuronowych są pewnym uproszczeniem procesu przetwarzania informacji w sposób jaki robi to mózg. Podstawową jednostką każdej sieci jest tzw. sztuczny neuron, który jest główną jednostką obliczeniową przetwarzając odebrane informacje. W pewien sposób odzwierciedlają one działanie biologicznych odpowiedników. Wszystkie neurony występujące w sieci są ze sobą wzajemnie połączone, co tworzy złożoną sieć pozwalającą na dokonywanie złożonych i kompleksowych predykcji. Strukturalnie sieć neuronową można więc przedstawić jako graf skierowany, w którym węzłami są neurony, a krawędzie grafu stanowią połączenia między neuronami.

Na potrzeby tej pracy inżynierskiej zostały przedstawione sieci neuronowe jednokierunkowe, zwane również perceptronami wielowarstwowymi. Taka sieć jest najprostszym i podstawowym typem sieci neuronowych, która cechuje się tym, że informacje między neuronami przekazywane są w jednym kierunku, bez występowania sprzężeń zwrotnych. Jednokierunkowa sieć neuronowa zawiera w sobie trzy rodzaje warstw, którymi są:

- **warstwa wejściowa**
- **warstwa ukryta**
- **warstwa wyjściowa**

Graficzne przedstawienie jednostronnej sieci neuronowej prezentuje **Rys. 4.16.**



Rys. 4.16. Przykład jednokierunkowej sieci neuronowej składającej się z trzech zmiennych wejściowych

Pierwszą warstwą sieci jest warstwa wejściowa. Przyjmuje ona dane wejściowe do sieci. Liczba neuronów w warstwie wejściowej jest tożsama z liczbą zmiennych niezależnych w danych. Kolejną warstwą jest tzw. warstwa ukryta. W każdej sieci występuje minimum

jedna warstwa ukryta. Neurony w tej warstwie odpowiadają za odpowiednie przetwarzanie otrzymanych danych. Ostatnią warstwą sieci jest warstwa wyjściowa, która odpowiada za ostateczną predykcję modelu.

W rozumieniu sztucznej sieci neuronowej pojedynczy neuron posiada wiele wejść i tylko jedno wyjście. W zapisie matematycznym pojedynczy neuron realizuje funkcję daną wzorem:

$$y = f(\sum_{i=1}^{n} w_i x_i + w_0) \tag{4.35}$$

gdzie, w_i są wagami dla odpowiedniego wejścia x_i neurona, w_0 jest pewną wartością stałą, która stanowi dodatkowy parametr. Parametr ten nazywany jest biasem. Wartość sumy ważonej wraz z biasem poddawana jest działaniu tzw. funkcji aktywacyjnej f, która przetwarza wprowadzone dane na dane wyjściowe y. Dzięki funkcji aktywacyjnej model jest w stanie odwzorować nieliniowe zależności, które występują między zmiennymi niezależnymi, a zmienną zależną. Przykładowymi takich funkcji aktywacyjnych są:

- **ReLu (Rectified Linear Unit)**:

$$f(x) = \max(0, x) \tag{4.36}$$

- **Tanh (Tangens hiperboliczny)**:

$$f(x) = \ln(1 + e^x) \tag{4.37}$$

- **SoftPlus**:

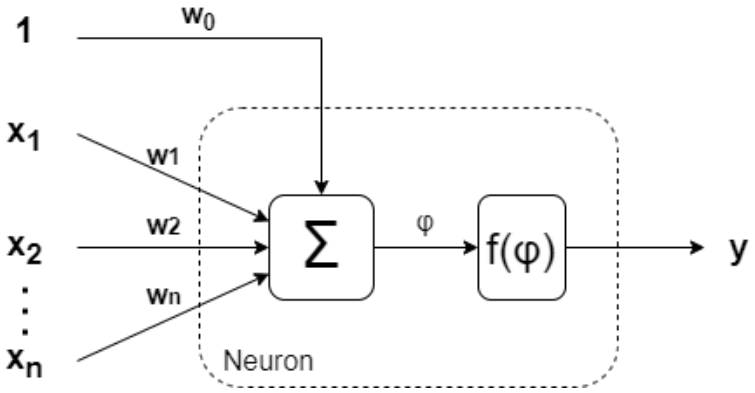
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.38}$$

W przypadku gdy zależności nieliniowe w modelu nie są pożądane stosuje się funkcję liniową jako funkcje aktywacyjną, która po prostu zwraca wartość wejściową. W zapisie matematycznym:

- **Funkcja liniowa**:

$$f(x) = x \tag{4.39}$$

Schemat pojedynczego neuronu prezentuje **Rys. 4.17.**, gdzie $\sum$ odnosi się do sumy wartości wejściowych i jako φ poprzez funkcję aktywacyjną $f(\varphi)$ podawane jest na wyjście neuronu y.



Rys. 4.17. Budowa pojedynczego neuronu.

Po skonstruowaniu sieci neuronowej, następnym krokiem jest przystąpienie do procesu trenowania. Trenowanie sieci polega na stopniowym dostosowywaniu parametrów sieci, którymi są wagi oraz biasy aby poprawić jej możliwości predykcyjne. Najbardziej popularną metodą, która jest stosowana do tego celu jest propagacja wsteczna *(ang. backpropagation)*. Propagacja wsteczna jest metodą iteracyjną, w której w każdym kroku podaje się dane treningowe do sieci, porównuje uzyskane wyniki z rzeczywistymi wartościami wyznaczając funkcję strat, a następnie na podstawie otrzymanej funkcji oblicza się gradient z uwzględnieniem każdego z parametrów. Gradient wskazuje kierunek, w którym funkcja rośnie najszybciej, a kierunek przeciwny wskazuje, gdzie funkcja maleje najszybciej, więc jest wykorzystywany przez tzw. optymalizatory, aby zmienić wartości poszczególnych parametrów tak, aby zminimalizować błąd predykcji. Przykładem często stosowanych algorytmów optymalizujących dla propagacji wstecznej są takie optymalizatory jak:

- **Spadek gradientu** *(ang. Gradient descent, DG)*
- **Stochastyczny spadek gradientu** *(ang. Stochastic Gradient Descent, SGD)*
- **Metoda średniego kwadratu propagacji** *(ang. Root Mean Square Propagation, RMSProp)*
- **Metoda Adaptacyjnej Estymacji Momentu** *(ang. Adaptive Moment Estimation, Adam)*

Proces trenowania jest powtarzany do momentu, aż zostanie osiągnięta określona liczba epok lub gdy wartość funkcji straty jest wystarczająco niska, a sieć neuronowa jest w stanie dokładnie przewidywać wartości wyjściowe dla nowych danych.

5 Wybrane narzędzia programistyczne do przetwarzania, wizualizacji danych oraz uczenia maszynowego wykorzystane przy realizacji projektu dyplomowego

5.1 Podstawowe narzędzia programistyczne

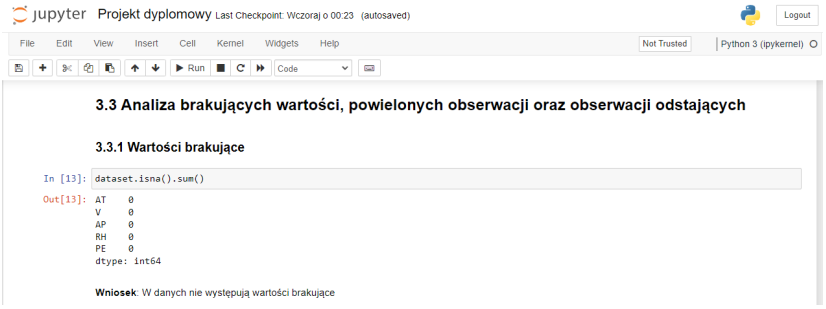
5.1.1 Język programowania Python

Językiem programowania wykorzystanym w projekcie jest język wysokiego poziomu o ogólnym przeznaczeniu *Python*. Kod napisany w tym języku działa od razu, bez konieczności korzystania z długotrwałego procesu kompilacji oraz innych narzędzi zewnętrznych, co zapewnia szybkość tworzenia kodu. Łączy w sobie zarówno paradygmaty programowania proceduralnego, funkcjonalnego oraz obiektowego [34]. Dodatkowo, charakteryzuje się niewielkim stopniem skomplikowania przy tworzeniu kodu, a jego składnia jest czytelna. W języku *Python* są stale tworzone, wspierane i udoskonalane różne biblioteki, między innymi te, które ułatwiają prace oparte na danych. Dzięki temu jest szeroko wykorzystywany przy realizacji projektów, która łączy w sobie takie dziedziny nauki, jak statystykę, matematykę oraz sztuczną inteligencję.

5.1.2 Środowisko programistyczne Jupyter Notebook

Środowiskiem programistycznym, które zostało użyte przy realizacji projektu dyplomowego jest *Jupyter Notebook*. Jest to interaktywne środowisko obliczeniowe, które pozwala na tworzenie dokumentów składających się z komórek, które mogą zawierać zarówno fragmenty kodu, tekst z szerokimi możliwościami formatowania dzięki językowi znaczników *Markdown* oraz multimedia. Język znaczników *Markdown* pozwala między innymi na tworzenie nagłówków, dodawanie stylu do tekstu, umieszczanie linków, obrazów, tworzenie tabel oraz list. Dzięki takim możliwościom środowisko *Jupyter Notebook* stało się popularnym narzędziem wykorzystywanym do tworzenia różnego rodzaju raportów, analiz danych, czy modeli uczenia maszynowego i wielu innych. Jest

również w pełni kompatybilne z językiem programowania *Python*. Fragment prezentujący wykorzystanie środowiska *Jupyter Notebook* prezentuje **Rys. 5.1.**



Rys. 5.1. Zaprezentowanie środowiska Jupyter Notebook

5.2 Narzędzia do manipulacji i analizy danych

5.2.1 Biblioteka NumPy

Numpy jest podstawową biblioteką służącą do obliczeń naukowych używaną między innymi w zastosowaniach matematycznych i informatycznych w języku *Python* [35]. Zapewnia takie obiekty jak wielowymiarowe tablice oraz szereg procedur pozwalający na operowaniu na nich w sposób matematyczny, logiczny, manipulując ich kształtami, sortując je i wiele innych. Biblioteka ta zapewnia wiele funkcjonalności z takich dziedzin nauki jak Algebra liniowa czy Statystyka.

5.2.2 Bibliotek Pandas

Biblioteka *Pandas* [36] jest biblioteką języka *Python*, która znajduje zastosowanie w pracy z danymi. Dostarcza szybkie i efektywne struktury ze zintegrowanym indeksowaniem takie jak *DataFrame* oraz *Series*. Pozwala na odczytywanie i zapisywanie danych w takich formatach jak: *CSV*, plik tekstowy, arkusz kalkulacyjny, *SQL*, *HDF5*. Zapewnia niezbędne narzędzia do przycinania, przekształcania, indeksowania, tworzenia mniejszych zestawów będący częścią całości danych, dodawania i usuwania danych i wielu innych użytecznych operacji przy analizie i manipulacji danymi. Fragment możliwości biblioteki *Pandas* prezentuje **Kod 1.**

```
# Zaimportowanie biblioteki
import pandas as pd

#wczytanie zestawu danych
dataset = pd.read_excel('Dane/Folds5x2_pp.xlsx')

# Wyświetlenie podstawowych statystyk opisowych danych liczbowych
print(dataset.describe())
```

Kod 1. Zaprezentowanie podstawowych operacji wczytania danych i wygenerowania statystyk opisowych dla danych przy użyciu biblioteki Pandas

5.3 Narzędzia pomocnicze przydatne do implementacji uczenia maszynowego

5.3.1 Biblioteka Scikit-learn

Scikit-learn jest kolejną biblioteką udostępnioną za pośrednictwem języka *Python*, która jest wykorzystywana do uczenia maszynowego. Zapewnia narzędzia, których zastosowanie pozwala na wdrożenie takich operacji jak: przetwarzanie wstępne danych, wybór modelu, zredukowanie wymiarowości, stworzenie modeli klasyfikacyjnych, regresji oraz klasteryzacji [37]. Przy realizacji projektu zostały wykorzystane następujące modele regresji:

- **Regresja liniowa**

  Moduł: *sklearn.linear_model*

  Klasa: *LinearRegression*

- **K-najbliższych sąsiadów**

  Moduł: *sklearn.neighbors*

  Klasa: *KNeighborsRegressor*

- **Drzewo decyzyjne**

  Moduł: *sklearn.tree*

  Klasa: *DecisionTreeRegressor*

- **Las losowy**

  Moduł: *sklearn.ensemble*

Klasa: *RandomForestRegressor*

- **Wzmocnienie gradientu**

  Moduł: *sklearn.ensemble*

  Klasa: *GradientBoostingRegressor*

Przykładową implementację budowy i trenowania modelu regresji, która została użyta w ramach projektu prezentuje **Kod 11.**

```
# Zaimportowanie modelu Gradient Boosting Regressor
from sklearn.ensemble import GradientBoostingRegressor

# Inicjalizacja modelu Gradient Boosting Regressor
tuned_model = GradientBoostingRegressor(learning_rate =
 ↪  0.1, n_estimators = 50, max_depth = 4)

# Trenowanie modelu na zestawie uczącym
tuned_model.fit(X_train, y_train)

```

Kod 2. Implementacja modelu Wzmocnienia gradientu przy użyciu biblioteki Scikit Learn

Poza modelami zapewnionymi przez bibliotekę zostały zastosowane w projekcie również inne narzędzia, między innymi użyteczne przy wyborze modelu. Oto zestawienie kilku takich narzędzi z biblioteki *Scikit-learn*:

- **train_test_split**

  Stosowane do przeprowadzenia losowego podziału zbioru danych na dwa podzbiory: zestaw uczący oraz zestaw testujący.

- **KFold**

  Stosowane do zaimplementowania k-krotnej walidacji krzyżowej. Generuje indeksy, które pozwalają na podział danych na zestaw uczący i testujący.

- **GridSearchCV**

  Stosowane do przeprowadzenie procesu optymalizacji hiperparametrów przy użyciu metody przeszukiwania siatki (*ang. grid search*).

- **r2_score, mean_squared_error, mean_absolute_error**

  Stosowane do zaimplementowania miar do oceny jakości modeli predykcyjnych.

Działanie jednego z wymienionych wyżej narzędzi prezentuje **Kod 3.**

```
# Funkcja do przeszukiwania hiperparametrów danego estymatora
from sklearn.model_selection import train_test_split
# Zeminne niezależne
input_cols = dataset.columns[dataset.columns != 'EP']
# Zmienna zależna
target_col = dataset.columns[dataset.columns == 'EP']
# Podział danych na grupę uczącą oraz testującą w proporcjach 80:20
X_train, X_test, y_train, y_test =
 ↪ train_test_split(dataset[input_cols], dataset[target_col],
 ↪ test_size=0.2, random_state=42)
```

Kod 3. Implementacja podziału losowego danych na zbiór treningowy i testujący

5.3.2 Biblioteka Keras

Keras to bibliotek służąca do tworzenia oraz trenowania modeli opartych na sztucznych sieciach neuronowych [38]. Został stworzony z myślą o zapewnieniu elastyczności, wydajności i skalowalności, przy jednoczesnym zachowaniu prostej formy. Pozwala w łatwy i szybki sposób skomponować strukturę sieci z dostępnych komponentów i odpowiednio je skonfigurować, ale także pozwala na tworzenie własnych komponentów. Zapewnia wsparcie dla takich jednostek obliczeniowych, jak *CPU*, *GPU* czy *TPU*.

Przykład wykorzystania biblioteki *Keras* do budowy sieci neuronowej pokazuje **Kod 12.**

```
# Zaimportowanie niezbędnych narzędzi biblioteki Keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Zainicjowanie modelu sztucznej sieci neuronowej
ann = Sequential()

# Dodanie warstwy wejściowej oraz pierwszej  warstwy ukrytej
ann.add(Dense(units=6, activation='relu'))

# Dodanie drugiej warstwy ukrytej
ann.add(Dense(units=6, activation='relu'))

# Dodanie warstwy wyjściowej
ann.add(Dense(units=1))

# Kompilacja sieci
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Trenowanie sieci
ann.fit(X_train, y_train, batch_size = 32, epochs = 100, validation_split=0.2)
```

Kod 4. Implementacja sztucznej sieci neuronowej przy użyciu biblioteki Keras

5.4 Narzędzia wizualizacyjne

5.4.1 Biblioteki Matplotlib oraz Seaborn

Matplotlib [39] jest biblioteką do tworzenia różnego rodzaju wizualizacji w języku programowania *Python*. Wizualizacje te mogą być równie dobrze statyczne, co animowane lub interaktywne. Pozwala między innymi na dostosowywanie układu i stylu wykresu, tworzenie wielu wykresów w ramach jednego rysunku oraz udostępnia możliwość łatwego eksportu stworzonych wizualizacji do wielu formatów.

Biblioteka *Seaborn* [40] jest rozwinięcie biblioteki *Matplotlib*, z racji tego, że jest na niej

zbudowana. Pozwala na tworzenie bardziej zaawansowanych form wizualizacji, którymi są między innymi różne formy wykresów takich, jak wykresy pudełkowe, wykresy gęstości, czy mapy cieplne.

Poniżej został umieszczony **Kod 5.** pochodzący z projektu, który skupia w sobie obydwie opisane biblioteki.

```
# Zaimportowanie bibliotek
import matplotlib.pyplot as plt
import seaborn as sns

# Stworzenie dwóch wykresów pudełkowych znajdujących
↪ się na jednym rysunku
fig, axes = plt.subplots(1, 2, figsize=(10,5))
sns.boxplot(data=dataset,ax=axes[0], x ='AP',color=
↪ 'g')
axes[0].set_xlabel('Ciśnienie otoczenia (AP)')
sns.boxplot(data=dataset,ax=axes[1], x ='RH',color=
↪ 'b')
axes[1].set_xlabel('Wilgotność względna (RH)')
fig.suptitle('Wizualizacja obserwacji odstających przy
↪ użyciu wykresów pudełkowych')

# Zapis rysunku w popularnym formacie PNG
plt.savefig('boxplot.png')

# Wyświetlenie wykresu
plt.show()

```

Kod 5. Implementacja wykresów pudełkowych przy użyciu biblioteki Seaborn oraz Matplotlib

6 Projekt inżynierski

6.1 Dane

6.1.1 Opis wykorzystanego zbioru danych

Na potrzeby projektu zostały wykorzystane dane, które składają się z czterech zmiennych niezależnych [41], gdzie pojedyncza obserwacja prezentuje uśrednione pomiary z każdej godziny. Zmienne te pozwalają na predykcję najwyższej wartości mocy pobranej w danej godzinie dla przedsiębiorstwa, którym jest elektrownia gazowo-parowa pracująca w cyklu kombinowanym. Proces gromadzenia danych trwał 6 lat i miał miejsce w okresie od 2006 do 2011 roku. Dane pochodzą z otwartego źródła i zostały udostępnione przez *the Machine Learning Repository of Center for Machine Learning and Intelligent Systems at the University of California, Irvine*.

W ramach tego zestawu danych zostało zebrane 9568 obserwacji, zawierających takie zmienne niezależne jak uśredniony godzinny pomiar temperatury w stopniach Celsjusza $°C$, ciśnienie otoczenia wyrażone w *milibarach*, wilgotność względna wyrażona w procentach $\%$ oraz podciśnienie w układzie wydechowym wyrażone w centymetrach słupa rtęci $cmHg$. Natomiast zmienną zależną, celu jest wspomniana już wcześniej najwyższa wartość mocy pobranej wyrażona w megawatach MW.

O ile temperatura i ciśnienie otoczenia są zmiennymi, które nie wymagają większych wyjaśnień, o tyle wilgotność względna oraz podciśnienie w układzie wydechowym nie są zmiennymi oczywistymi. Wilgotność względna jest miarą ilości pary wodnej występującej w powietrzu, w odniesieniu do maksymalnej ilości pary wodnej, która mogłaby wystąpić w tych konkretnych warunkach. Innymi słowy wilgotność względna określa, jaką ilość wilgoci zawiera powietrze, w stosunku do maksymalnej wilgoci, jaką może zawierać.

Podciśnienie w układzie wydechowym jest parametrem określającym wydajność odprowadzania spalin powstałych w procesie produkcyjnym. Podciśnienie to odnosi się do ciśnienia panującego wewnątrz układu wydechowego, którego wartość jest niższa od ciśnienia atmosferycznego.

Zarówno zmienne niezależne, jak i zmienna celu są formatu liczbowego i zawierają dane numeryczne w formacie liczb rzeczywistych, a używając nomenklatury programistycznej

liczb zmiennoprzecinkowych.

Podsumowanie zestawu danych prezentuje **Tab. 6.1**.

Nazwa zmiennej	Symbol	Rodzaj zmiennej	Jednostka
Temperatura	AT	niezależna (X)	$°C$
Ciśnienie otoczenia	AP	niezależna (X)	$milibar$
Wilgotność względna	RH	niezależna (X)	$\%$
Podciśnienie w układzie wydechowym	V	niezależna (X)	$cmHg$
Moc pobrana	EP	zależna (Y)	MW

Tab 6.1. Zestawienie zmiennych

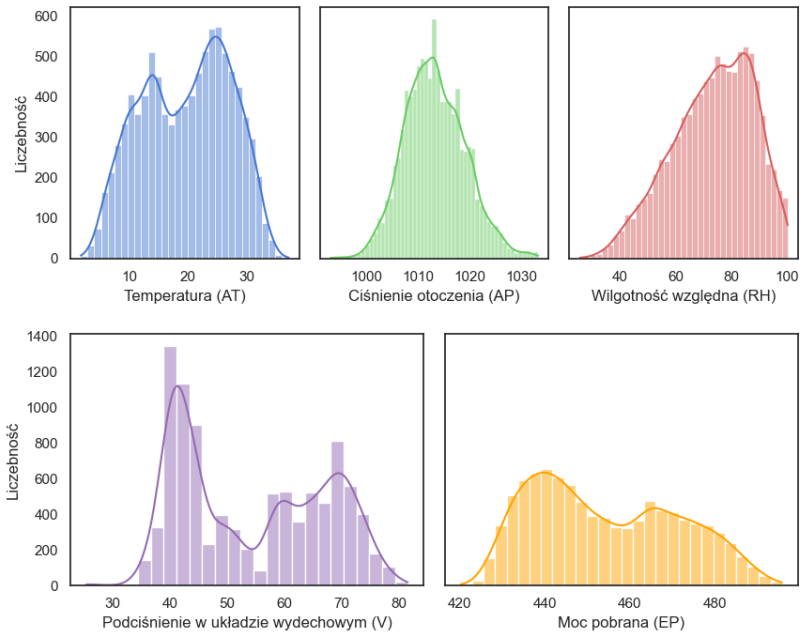
6.1.2 Analiza opisowa zmiennych

Jednym z podstawowych i początkowych etapów w toku analizy danych było poznanie podstawowych charakterystyk, specyfiki oraz pewnych tendencji, które występują w każdej z analizowanych zmiennych. Pozwoliło to na bardziej precyzyjne poznanie danych, wykrycie ewentualnych nieprawidłowości oraz określenie czy używane dane są wystarczająco ogólne i zawierają wysoki stopień zróżnicowania. Analiza ta opiera się zarówno na metodach wizualizacyjnych jak i na statystykach. W sposób ogólny można określić ją jako analiza statystyk opisowych, która składa się z trzech głównych elementów, którymi są:

- **Analiza rozkładu zmiennych** - jest to analiza, która pozwala poznać charakterystykę danej zmiennej w sposób kompleksowy. Jest techniką wizualizacyjną, która określa częstości występowania wartości obserwacji w określonym przedziale, a co za tym idzie pozwala określić wartości charakterystyczne dla zmiennej. Wizualizacja w dość oczywisty sposób prezentuje, czy obserwacje skupiają się wśród jednej centralnej wartości, zazwyczaj wartości średniej, czy też mają charakter bardziej rozproszony.
- **Analiza centralnych tendencji** - jest to analiza oparta na wartościach liczbowych, która skupia się na wyznaczeniu takich miar jak wartość średnia, mediana oraz wartości dominującej tzw. mody lub dominanty, której nie wyznacza się w przypadku, gdy zmienna ma charakter wielomodalny. Miary te opisują typowe wartości dla zmiennej.
- **Analiza zróżnicowania** - jest to analiza liczbowa, pozwalająca poznać zakres zmiennych oraz wartości skrajne przy użyciu takich miar jak odchylenie standardowe, wartość minimalna i maksymalna. Określa więc czy w danej zmiennej

występuje duże zróżnicowanie wartości obserwacji. Ponadto analiza takich statystyk jak kwartyle pozwala określić, gdzie obserwacje się kumulują.

W oparciu o charakterystykę rozkładu zmiennych **Rys. 6.1.** oraz z uwzględnieniem podstawowych statystyk **Tab. 6.2** została przeprowadzona analizy dla każdej ze zmiennych.



Rys. 6.1. Rozkład zmiennych

	AT	**AP**	**RH**	**V**	**EP**
Wartość średnia	19.65	1013.26	73.31	54.31	454.37
Odchylenie standardowe	7.45	5.94	14.60	12.71	17.07
Wartość minimalna	1.81	992.89	25.56	25.36	420.26
Kwartyl pierwszy	13.51	1009.10	63.33	41.74	439.75
Mediana	20.34	1012.94	74.97	52.08	451.55
Kwartyl trzeci	25.72	1017.26	84.83	66.54	468.43
Wartość maksymalna	37.11	1033.30	100.16	81.56	495.76
Zakres zmiennej	35.30	40.41	74.60	56.20	75.50

Tab 6.2. Podstawowe statystyki zmiennych

Zmienna *Temperatura* charakteryzuje się rozkładem dwumodalnym, a więc takim w którym występują dwa charakterystyczne szczyty, inaczej dwa przedziały obserwacji o największej częstości, które są dominujące w stosunku do wartości sąsiadujących. Układ

dwóch szczytów rozkłada się po dwóch stronach od wartości średniej. Co więcej nieznacznie więcej obserwacji skupia się po prawej stronie wykresu, co oznacza, że znajduje się powyżej wartości średniej.

Ciśnienie otoczenia charakteryzuje się występowaniem jednego wyróżniającego się przedziału pod względem liczebności obserwacji w centralnej części wykresu rozkładu, w bliskim sąsiedztwie wartości średniej oraz mediany. Obserwacje rozkładają się w sposób w miarę symetryczny po dwóch stronach od wartości średniej. Wskazuje to na to, że zmienna ma rozkład zbliżony do rozkładu normalnego.

W przypadku zmiennej niezależnej *Wilgotność względna* widać jedno skupienie częstości. Można więc stwierdzić, że zmienna charakteryzuje się rozkładem jednomodalnym, w którym przedział częstości obserwacji dominującej koncentruje się po prawej stronie wykresu. Więc rozkład jednomodalny jest także lewoskośny.

Podciśnienie w układzie wydechowym podobnie jak zmienna *Temperatura* cechuję się występowaniem dwóch charakterystycznych szczytów na wykresie. Szczyt znajdujący się po lewej stronie wykresu, w zakresie wartości poniżej wartości średniej, cechuje się wysoką liczebnością przedziału wartości dominującej w porównaniu do drugiego szczytu, występującego po prawej stronie od wartości średniej.

Zmienna zależna, określana również zmienną celu *Moc pobrana* podobnie jak zmienna *Temperatura* oraz *Podciśnienie w układzie wydechowym* jest rozkładu dwumodalnego. Zarysowuje się wyraźna tendencja do występowania większej liczby obserwacji po lewej stronie wykresu.

Wyraźnie widać, że zmienne mają różne typy rozkładu. Tylko jedna ze zmiennych posiada rozkład zbliżony do rozkładu normalnego, dla pozostałych są to albo rozkład dwumodalny, albo jednomodalny przesunięty.

Na bazie obserwacji zmienności i zakresu zmiennych można wywnioskować, że zmienne posiadają szeroki zakres. Oznacza to, że pomiary poszczególnych zmiennych dokonywane były w różnych warunkach pogodowych, w różnych porach roku i porach dnia. Zmienne zawierają zarówno obserwacje typowe dla tego typu pomiarów, jak i te, które można uznać za rzadkie, bądź nietypowe. Na tym etapie nie uwydatniły się żadne nieprawidłowości występujących w danych.

6.1.3 Wykrywanie oraz obsługa wartości brakujących, powielonych obserwacji oraz wartości odstających.

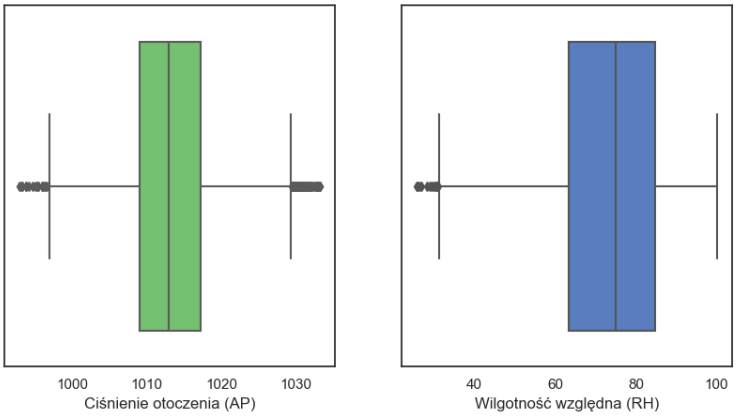
Analiza wartości brakujące, powielonych rekordów oraz wartości odstających pozwala na sprawdzenie, czy pewne szczególne zjawiska, które mogą mieć wpływ na jakość zbudowanego modelu, są obecne w danych i w przypadku ich występowania ułatwia dobranie odpowiedniej strategii, jak z tymi zjawiskami należy postępować.

Pierwszym elementem poddanym sprawdzeniu było wykrycie braków w danych. W rezultacie tejże analizy wynikło, że w zestawie danych nie ma brakujących wartości.

Odnosząc się jednak do powielonych obserwacji, to takie obserwacje w danych występują, konkretnie jest ich 41. Po bliższej analizie tych obserwacji, została podjęta decyzja o ich pozostawieniu. Wystąpienia identycznych obserwacji jest wysoce prawdopodobne w odniesieniu do pomiarów, które były przeprowadzane na przestrzeni 6 lat. Została więc wykluczona możliwość, że identyczne obserwacje wynikają z błędu.

Ze względu na to, że w większości zmienne są odmiennego rozkładu niż rozkład normalny oraz typy rozkładu są mocno zróżnicowane, to do wykrycia elementów odstających w zmiennych została zastosowana metoda oparta na *rozstępie międzykwartylowym IQR*. Metoda ta charakteryzuje się odpornością na typy rozkładu zmiennej, ponieważ opiera się na kwartylach: kwartylu pierwszym q_1, medianie oraz kwartylu trzecim q_3. W metodzie wykorzystującej *IQR* wszystkie obserwacje, które nie znajdują się w zakresie $[q_1 - 1.5 \cdot IQR, q_3 + 1.5 \cdot IQR]$ są zakwalifikowane jako obserwacje odstające. *IQR* opisuje się jako różnicę między q_3, a q_1.

W toku analizy, okazało się, że tylko dwie zmienne zawierają obserwacje odstające. Są nimi *Ciśnienie otoczenia (AP)* oraz *Wilgotność względna (RH)*. Obserwacje odstające dla wspomnianych zmiennych wizualizuje wykres pudełkowy **Rys. 6.2.**



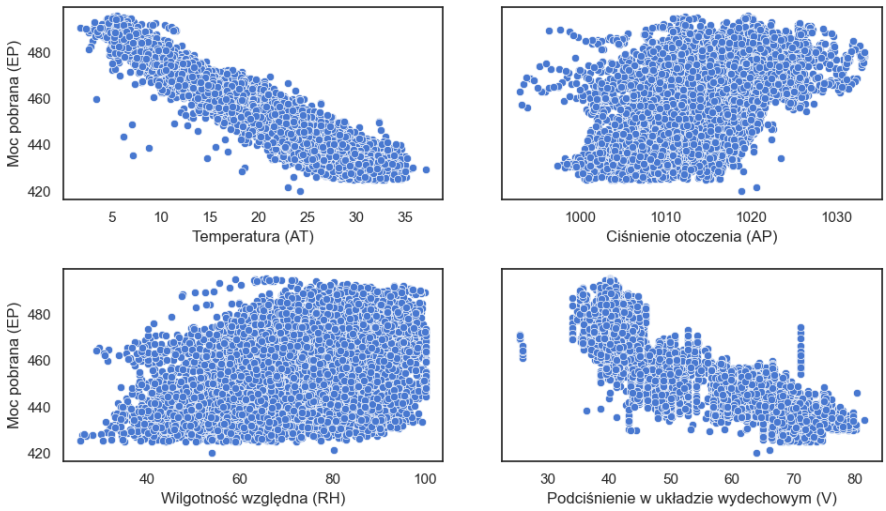
Rys. 6.2. Wizualizacja obserwacji odstających przy użyciu wykresów pudełkowych

Dla zmiennej *AP* obserwacje odstające występują zarówno poniżej dolnej granicy, jak i granicy górnej. Przedział obserwacji odstających dla dolnej granicy wynosi $[996.55, 996.86]$, gdzie wartość 996.86 jest równa dolnej granicy przedziału. Przedział dla obserwacji powyżej górnej granicy obejmuje zakres $[1029.5, 1029.54]$, gdzie $1029,5$ jest wartością górnej granicy . Zmienna *RH* natomiast obserwacje odstające zawiera tylko poniżej dolnej granicy, który mieści się w zakresie $[30.99, 31.07]$. Analogicznie wartość 31.07 jest wartością dolnej granicy.

Widoczne jest to, że w przypadku obydwu zmiennych obserwacje, które zostały zakwalifikowane jako odstające znajdują się w bliskim otoczeniu granic, dla których obserwacje kwalifikuje się jako prawidłowe. W danych nie ma więc wartości odstających w sposób zdecydowany. Można więc przyjąć, że wartości te z wysokim stopniem prawdopodobieństwem nie są wartościami wynikającymi z błędu pomiarowego, czy też z niewłaściwego uzupełnienia bazy pomiarów, w której obserwacje są przechowywane. Wartości te wydają się wartościami prawidłowymi i mogą wystąpić w warunkach rzeczywistych. W związku z tym obserwacje te zostały pozostawione w niezmienionej formie w zestawie danych.

6.1.4 Badanie zależności między zmiennymi niezależnymi, a zmienną zależną

Po sprawdzeniu jakości zestawu danych zostały przeprowadzone badania zależności jakie występują w między każdą ze zmiennych niezależnych X, a zmienną zależną y. Relacje te prezentuje grupa wykresów punktowych **Rys. 6.3.**



Rys. 6.3. Wykresy punktowy relacji między zmiennymi niezależnymi, a zmienną zależną

Powyższy wykres prezentuje liniową zależność występująca między zmienną *Temperatura (AT)* a zmienną celu *Moc pobrana (EP)* oraz między *Podciśnienie w układzie wydechowym (V)* i *EP*. Dla pozostałych dwóch zmiennych, którymi są *Ciśnienia otoczenia (AP)* i *Wilgotności względnej (RH)*, a *EP* zależność ta wydaje się słabsza.

Z uwagi na wyraźną zależność liniowej widoczną na wykresach relacji między zmiennymi, przeprowadzono analizę korelacja za pomocą współczynnika korelacji *Pearsona*. Współczynnik ten służy do określenia stopinia zależności liniowej występującej między zmiennymi, a określony jest następującym wzorem:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \; r_{xy} \in [-1, 1] \tag{6.1}$$

gdzie, r_{xy} to współczynnik korelacji liniowej Pearsona między zmiennymi x i y, n liczba obserwacji, x_i i y_i to wartości i-tej obserwacji zmiennych x i y, a $\bar{x}$ i $\bar{y}$ to odpowiednio średnie wartości dla zmiennych x i y.

Współczynnik Korelacja między zmienną *AT*, a zmienną celu *EP* wynosi -0.95. Współczynnik dla zmiennych *V* i *EP* jest także ujemny i wynosi -0.87. Współczynniki dla pozostałych zmiennych tj. *AP* i *EP* oraz *RH* i *EP* wynosi kolejno 0.52 oraz 0.39.

Na podstawie powyższych wartości współczynników wyraźnie widać, że zmienne niezależne są powiązane w sposób liniowy ze zmienną celu. Współczynniki między zmiennymi *T*, *V*, *AP*, a zmienną *EP* są wysokie, przy założeniu, że wysoki współczynniki korelacji zawiera się w przedziale $[-1, -0.5] \cup [0.5, 1]$, natomiast między *RH*, a *EP*

współczynnik można uznać za słabszy, jednak pewna zależność liniowa występuje.

6.2 Wybór optymalnego modelu

Wybór finalnego modelu w ramach projektu opierał się na zachowaniu równowagi między obciążeniem oraz wariancją. Pojęcie obciążenia jest miarą, która określa jak predykcje dokonywane przez model są zbliżone do wartości rzeczywistych. Wariancja opisuje natomiast jak bardzo zdolności prognozowania modelu zmieniają się pod wpływem różnych zestawów danych wejściowych. Zatem preferowanymi modelami były te, które zachowując wysoką precyzję predykcji, zachowywały również niski poziom wariancji.

Zestaw danych początkowo został podzielony na dwa podzbiory. Pierwszy podzbiór jest to zestaw uczący, który służył do wytrenowania rozpatrywanych modeli, ich walidacji oraz w konsekwencji wyboru najlepszego. Pozostałą część danych stanowił zestaw testujący, który po wytrenowaniu finalnie wybranego modelu na danych uczących służył jako podstawa do końcowej oceny modelu.

Pierwszy etap podziału był w stosunku 80 : 20, co stanowi 7654 obserwacje dla zestawu uczącego oraz 1914 obserwacji w grupie testującej. Następnie przy użyciu pięciokrotnej walidacji krzyżowej na grupie uczącej został wykonane dostrojenie oraz selekcja modelu. Dzięki podziałowi na 5 podzbiorów w ramach walidacji krzyżowej możliwe było uzyskanie podziału danych w stosunku zbliżonym do 80 : 20, co daje ok. 6123 obserwacje do uczenia modeli i 1531 obserwacji służących do walidacji.

W ramach selekcji wybraną miarą był średni błąd kwadratowy *MSE*. Zastosowanie błędu średniokwadratowego wynika z łatwości interpretacji wyników, im błąd jest mniejszy, tym predykcje są bardziej dokładne. Kolejną istotną zaletą zastosowania tej miary, jest fakt, że wysokie błędy predykcji są uwypuklone oraz podkreślane. Pozwala to ocenić czy model radzi sobie skutecznie z obserwacjami nietypowymi, co w przypadku tworzenia ogólnego i skutecznego modelu jest istotne.

Algorytmy, które zostały wykorzystane w procesie selekcji modelu różnią się między sobą metodami rozwiązywania analizowanego zagadnienia oraz stopniem złożoności obliczeniowej. Tymi technikami sztucznej inteligencji są:

- **Regresja Liniowa**
- **Algorytm K-najbliższych sąsiadów**
- **Drzewo decyzyjne**
- **Las losowy**

- **Wzmocnienie gradientu**
- **Sztuczna sieć neuronowa**

6.2.1 Proces optymalizacji rozważanych algorytmów

Wybór optymalnych hiperparametrów oparł się na dwóch zasadniczych kryteriach. Pierwszym z tych kryteriów jest minimalizacja uśrednionego wyniku *MSE* dla danych walidacyjnych uzyskanych z pięciokrotnej walidacji krzyżowej. Drugie kryterium jest minimalizacja wariancji pomiędzy uśrednionymi wynikami z zestawu danych na których model był trenowany, a zestawem walidacyjnym. Graniczną wartością wariancji między wspomnianymi zestawami są 2 jednostki. Celem doboru takich kryteriów było uzyskanie możliwie dokładnych modeli, przy jednoczesnym zachowaniu wysokiego stopnia ogólności.

Dla algorytmów K-najbliższych sąsiadów, Drzewa decyzyjnego, Lasu losowego oraz Wzmocnienia gradientu została zaimplementowana metoda **Kod 6.** do poszukiwania hiperparametrów metodą siatki przy użyciu klasy *GridSearchCV* z biblioteki *Scikit-learn*.

```
1  # Zaimportowanie implementacji algorytmu przeszukiwania siatki z biblioteki Scikit-learn
2  from sklearn.model_selection import GridSearchCV
3
4  # Funkcja do przeszukiwania hiperparametrów danego estymatora
5  def hyperparam_checking(estimator, params, X_train, y_train):
6      scoring = 'neg_mean_squared_error'
7      grid_search = GridSearchCV(estimator, param_grid=params, cv=5, scoring=scoring, return_train_score=True, n_jobs=-1)
8      grid_search.fit(X_train, y_train.values.ravel())
9      results = grid_search.cv_results_
10
11     return results
12
```

Kod 6. Implementacja algorytmu przeszukiwania siatki do znalezienia hiperparametrów danej metody uczenia maszynowego.

W przypadku Regresji liniowej oraz sztucznej sieci neuronowej zostało zastosowane odmienne podejście, a implementacja obranej metodologii została zaprezentowana w poszczególnych punktach tych metod.

6.2.1.1 Regresja liniowa

W niniejszej pracy wykorzystano podstawową formę algorytmu Regresji liniowej, co oznacza, że została wykorzystana technika dopasowania linii prostej, bez stosowania żadnej formy regularyzacji. Taka forma Regresji liniowej nie zawiera parametrów algorytmu, a mówiąc inaczej hiperparametrów.

Do zaimplemenotwania Regresji liniowej została użyta klasa *LinearRegression()* biblioteki *Scikit-learn*, a następnie model został oceniony poprzez zastosowanie metody *cross_validate* zapewnionej także przez bibliotekę *Scikit-learn*. Praktyczne zastosowanie wspomnianych wyżej narzędzi prezentuje **Kod 7.**

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_validate

linear = LinearRegression()
scoring = 'neg_mean_squared_error'
scores_lin = cross_validate(linear, X_train,
↪  y_train.values.ravel(), cv=5, scoring=scoring,
↪  return_train_score=True)
```

Kod 7. Implementacja Regresji liniowej przy użyciem biblioteki Scikit-learn

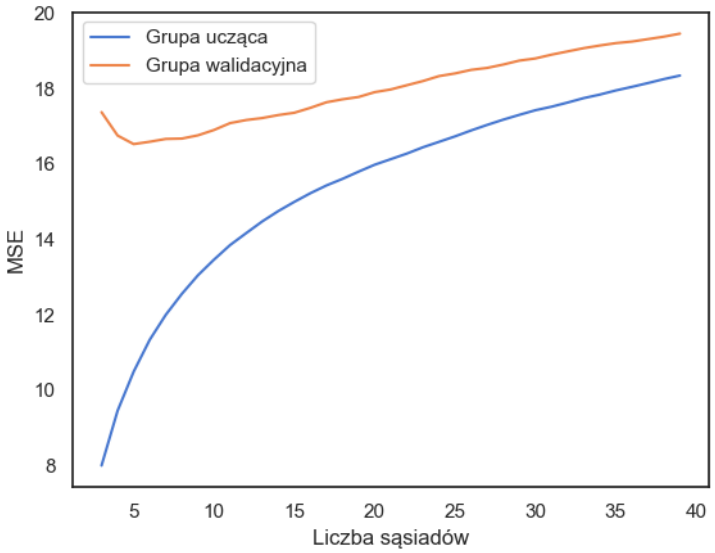
Wyniki algorytmu uzyskane podczas eksperymentu dla pięciokrotnej walidacji krzyżowej prezentują się w sposób następujący:

- **Średni wynik MSE grupy uczącej**: 20.89
- **Średni wynik MSE grupy walidacyjnej**: 20.92
- **Różnica między grupą uczącą, a walidacyjną**: 0.03

6.2.1.2 Algorytm K-najbliższych sąsiadów

Dla algorytmu K-najbliższych sąsiadów rozważanym hiperparametrem była liczba sąsiadów, która determinuje jak dużo punktów zostanie użyte w procesie poszukiwania nowej wartości.

Algorytm został sprawdzony dla liczby sąsiadów w zakresie $[1,40]$. Wpływ liczby sąsiadów na błąd średniokwadratowy zarówno na zestaw uczący, jak i na zestaw walidacyjny prezentuje **Rys. 6.4.**.



Rys. 6.4. Wpływ liczby sąsiadów na błąd średniokwadratowy dla zestawu uczącego oraz walidacyjnego

Powyższy wykres prezentuje, że wraz ze wzrostem liczby sąsiadów wariancja między obydwoma zestawami danych maleje, jednak wraz z malejącą wariancją rośnie błąd średniokwadratowy *MSE*. Po analizie wyników dla poszczególnych wartości parametru zgodnie z przyjętymi kryteriami wybraną wartością liczby sąsiadów była liczba 20.

Algorytm został zaimplementowany przy użyciu klasy *KNeighborsRegressor()*, a następnie wywołanie metody *hyperparam_checking* (**Kod 6.**) implementującej algorytm przeszukiwania siatki pozwoliło na wyselekcjonowanie wartości liczby sąsiadów. Obydwie klasy udostępnia biblioteka *Scikit-learn*. Implementacje przedstawia **Kod 8.**

```
1  from sklearn.neighbors import KNeighborsRegressor
2
3  estymator_knn = KNeighborsRegressor()
4  param_knn  = {'n_neighbors': list(range(1,41))}
5  knn_results = hyperparam_checking(estymator_knn,
   ↪  param_knn, X_train, y_train)
```

Kod 8. Implementacja algorytmu k-najbliższych sąsiadów przy użyciem biblioteki Scikit-learn

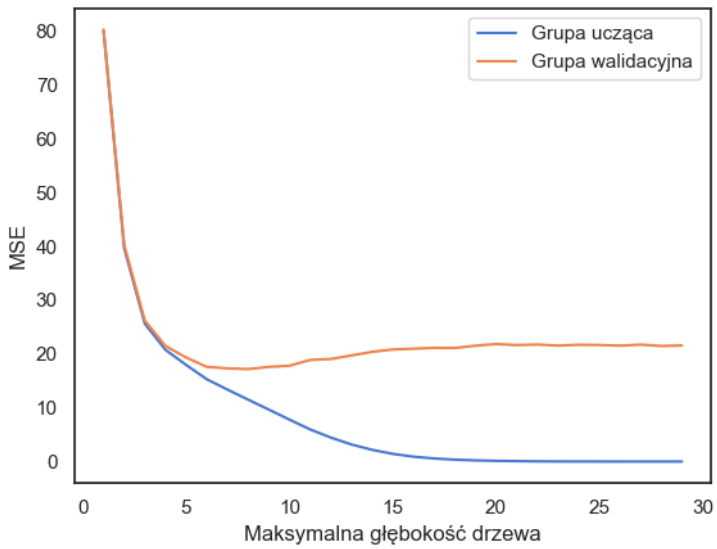
Wyniki dla dobranej wartości liczby sąsiadów uzyskane podczas eksperymentu dla pięciokrotnej walidacji krzyżowej prezentują się w sposób następujący:

- **Średni wynik MSE grupy uczącej**: 16.44
- **Średni wynik MSE grupy walidacyjnej**: 18.36
- **Różnica między grupą uczącą, a walidacyjną**: 1.92

6.2.1.3 Drzewo decyzyjne

Dla algorytmu Drzewa decyzyjnego [42] hiperparametrem branym pod uwagę w procesie dostrajania była maksymalna głębokość drzewa, która określa maksymalną liczbę warstw w drzewie, która może zostać wykorzystana przy tworzeniu gałęzi drzewa.

Głównym mankamentem drzew decyzyjnych jest tendencja do wysokiego stopnia dopasowania do danych uczących, co sprawia, że wariancja między zbiorem uczącym, a walidacyjnym może przyjmować wysoką wartość. Dlatego w kontekście projektu dla Drzewa decyzyjnego został przeanalizowany wpływ maksymalnej głębokości w zakresie $[1, 30]$. Zależność taką zarówno dla zestawu uczącego oraz walidacyjnego prezentuje **Rys. 6.5.**.



Rys. 6.5. Wpływ maksymalnej głębokości drzewa na błąd średniokwadratowy dla zestawu uczącego oraz walidacyjnego

Powyższy wykres przedstawia fakt, że od maksymalnej głębokości drzewa zależy zarówno wysokość błędu *MSE*, jak i wariancja występująca między zestawami. W zakresie $[1, 5]$ maksymalnej głębokości drzewa *MSE* maleje, a wariancja między zestawami danych jest niewielka. Po przekroczeniu wartości ok. 5 wariancja zaczyna rosnąć, aż do osiągnięcia

wartości ok. 15 dla maksymalnej głębokości, gdzie wzrost się stabilizuje. Po analizie wyników zgodnie z przyjętymi kryteriami dobraną wartością parametru dla drzewa decyzyjnego jest 5.

Implementacje algorytmu przedstawia **Kod 9.**

```
from sklearn.tree import DecisionTreeRegressor

estymator_tree = DecisionTreeRegressor()
param1_tree = {'max_depth': list(range(1,31))}
tree_results =
 ↪ hyperparam_checking(estymator_tree,param1_tree,
 ↪ X_train, y_train)
```

Kod 9. Implementacja algorytmu Drzewa decyzyjnego przy użyciem biblioteki Scikit-learn

Algorytm Drzewa decyzyjnego został zaimplementowany przy użyciu klasy *DecisionTreeRegressor()* z biblioteki *Scikit-learn*, a następnie poprzez zastosowanie metody *hyperparam_checking* (**Kod 6.**) implementującej algorytm przeszukiwania siatki została dobrana maksymalna głębokość drzewa.

Wyniki dla dobranej wartości hiperparametru algorytmu uzyskane podczas eksperymentu dla pięciokrotnej walidacji krzyżowej prezentują się w sposób następujący:

- **Średni wynik MSE grupy uczącej**: 18.28
- **Średni wynik MSE grupy walidacyjnej**: 19.87
- **Różnica między grupą uczącą, a walidacyjną**: 1.59

6.2.1.4 Las losowy

Dla Lasu losowego rozpatrywanymi parametrami w procesia strojenia modelu były maksymalna głębokość drzewa oraz liczba estymatorów. Parametr maksymalnej głębokości drzewa jest analogiczny do Drzewa decyzyjnego, natomiast parametr liczby estymatorów określa liczbę drzew, na którą składa się na las.

Przy doborze hiperparametrów dla lasu losowego została zastosowana technika przeszukiwania siatki (*ang. Grid Search*), która to metoda przeszukuje wszystkie dostępne kombinacje określonych wartości hiperparametrów. W przypadku maksymalnej głębokości drzewa wartości te stanowiły $[3,4,5,6,7,10,15,20,30]$, a liczba estymatorów $[50,70,90,100,200,300,400,500]$.

Implementacje algorytmu przedstawia **Kod 10.**

```
from sklearn.tree import RandomForestRegressor

estymator_forest = RandomForestRegressor()
param1_forest = {'max_depth': [3,4,5,6,7,10,15,20,30],
 ↪  'n_estimators' : [50,70,90,100,200,300,400,500]}
forest_results =
 ↪  hyperparam_checking(estymator_forest,param1_forest,
 ↪  X_train, y_train)
```

Kod 10. Implementacja algorytmu Lasu losowego

Algorytm Lasu losowego został zaimplementowany przy użyciu klasy *RandomForestRegressor()* z biblioteki *Scikit-learn*, a następnie wspomniana już wcześniej metoda *hyperparam_checking* (**Kod 6.**) posłużyła jako zaimplementowanie metody przeszukiwania siatki.

Najlepszy wynik zgodnie z kryteriami został osiągnięty dla głębokości drzewa równej 5 oraz liczbie estymatorów równej 200, a prezentują się w sposób następujący dla pięciokrotnej walidacji krzyżowe:

- **Średni wynik MSE grupy uczącej**: 16.42
- **Średni wynik MSE grupy walidacyjnej**: 17.85
- **Różnica między grupą uczącą, a walidacyjną**: 1.43

6.2.1.5 Wzmocnienie gradientu

Dla algorytmu Wzmocnienia gradientu (*ang. Gradient Boosting*) lista analizowanych parametrów metody składa się z maksymalnej głębokości drzewa, liczby estymatorów oraz współczynnika uczenia. O ile liczba estymatorów i głębokość drzewa zostały już opisane, o tyle współczynnik uczenia pojawia się po raz pierwszy. Hiperparametr ten określa prędkość z jaką modelu się uczy, a mówiąc bardziej precyzyjnie współczynnik uczenia określa wpływ dodawanych kolejno drzew na poprawę prognozy. Do doboru hiperparametrów podobnie jak w algorytmie Lasu losowego dla Wzmocnienia gradientu została zastosowana technika przeszukiwania siatki (*ang. Grid Search*). Dla maksymalnej głębokości drzewa rozpatrywanymi wartościami były $[2,4,5,6,10,15,30]$, dla liczby estymatorów $[10,50,75,100,200,300]$, a dla współczynnika uczenia $[0.1,0.2,0.3,0.5,1]$.

Implementacje algorytmu przedstawia **Kod 11.**

```
from sklearn.tree import GradientBoostingRegressor

estymator_gradient = GradientBoostingRegressor()
param1_gradient = { 'max_depth': [2,4,5,6,10,15,30],
        'n_estimators' : [10, 50, 75, 100, 200, 300],
        'learning_rate' : [0.1,0.2, 0.3,0.5, 1]}
gradient_results =
 ↪ hyperparam_checking(estymator_gradient,
 ↪ param1_gradient, X_train, y_train)
```

Kod 11. Implementacja algorytmu Wzmocnienia gradientu

Wzmocnienie gradientu zostało zaimplementowane przy użyciu klasy *GradientBoostingRegressor()* z biblioteki *Scikit-learn*, a nastepnie metoda zaprezentowana w kodzie **Kod 6.** *hyperparam_checking* pozwoliła na wdrożenie metody przeszukiwania siatki.

Najlepszy wynik zgodnie z przyjętymi kryteriami został osiągnięty dla głębokości drzewa równej 4, liczbie estymatorów równej 50 oraz współczynnika uczenia równego 0.1, a dla pięciokrotnej walidacji krzyżowej prezentują się w sposób następujący:

- **Średni wynik MSE grupy uczącej**: 13.26
- **Średni wynik MSE grupy walidacyjnej**: 15.12
- **Różnica między grupą uczącą, a walidacyjną**: 1.86

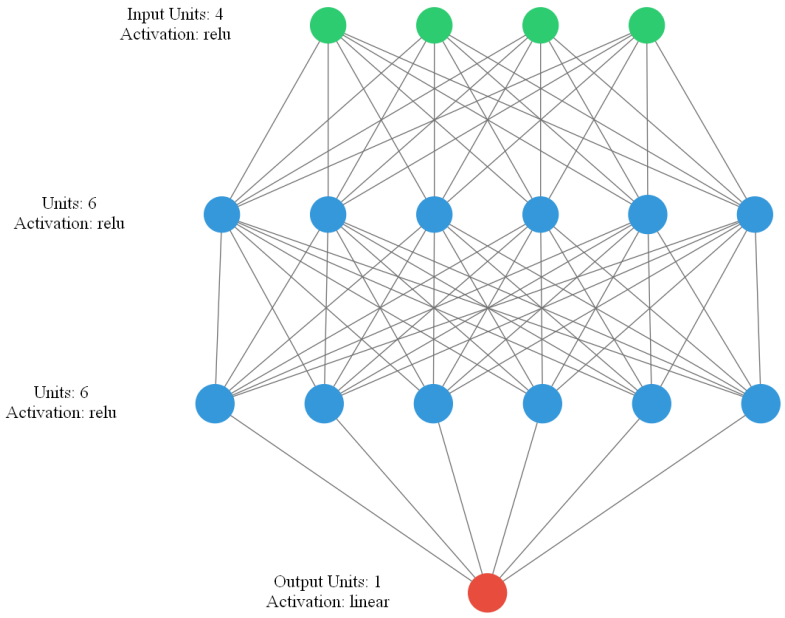
6.2.1.6 Sztuczna sieć neuronowa

W ramach projektu została skonstruowana jednokierunkowa sztuczna sieć neuronowa [43], która składa się z czterech głównych warstw:

- **Warstwa wejściowa**
- **Pierwsza warstwa ukryta**
- **Druga warstwa ukryta**
- **Warstwa wyjściowa**

Warstwa wejściowa składa się z 4 neuronów, obydwie warstwy ukryte tworzy po 6 neuronów, a w finalnej warstwie tzn. warstwie wyjściowej znajduje się jeden neuron. Zarówno dla warstwy wejściowej, jak i warstw ukrytych zastosowaną funkcją aktywacyjną była tzw. Rektyfikowana Jednostka Liniowa (*ang. Rectified Linear Unit, ReLU*), określana

jako funkcja $max(0,x)$. Dla warstwy wyjściowej funkcją aktywacyjną była funkcja liniowa. Pełną strukturę sieci ilustruje **Rys. 6.6.**



Rys. 6.6. Struktura sieci neuronowej wykorzystanej w projekcie

Kolejnymi istotnymi parametrami sieci jest rozmiar partii (*ang. Batch size*) oraz liczba epok *(ang. Epoch)*. Rozmiar partii określa ilość danych wejściowych, które są przetwarzane jednocześnie przez sieć w jednym kroku trenowania. Wszystkie dane są więc dzielone na odpowiednie partie i każda z nich jest podawana do sieci. Liczba epok definiuje ile razy dana sieć przetworzy cały zestaw danych w procesie uczenia. Dla skonstruowanej sieci neuronowej został przyjęty rozmiar partii na poziomie 32, a liczba epok na poziomie 100. Ponadto do optymalizacji sieci został wykorzystany algorytm Adaptacyjnego Estymowania Momentu (*ang. Adaptive Moment Estimation, Adam*).

Sieć neuronowa została zaimplementowana przy użyciu biblioteki Tensorflow, za pomocą klasy *Sequential* reprezentującej model sekwencyjny oraz klasy *Dense* określającej warstwy sieci, a następnie została poddana ocenie za pomocą klasy *KFold* z biblioteki *Scikit-learn*. Implementacje pokazuje **Kod 12.**

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold

#funkcja implementująca sieć neuronową
def eval_ann(hidden_layers, units, X_train, y_train,
 ↪  epochs, batch_size):
"""
Parametry:
----------
hidden_layers: liczba warstw ukrytych
units: liczba neuronów w warstwie ukrytej
X_train: Wektor zmiennych niezaleznych (X) zestawu
 ↪  uczącego
y_train: Wektor zmiennej celu (u) zestawu uczącego
epochs: liczba epok
batch_size: liczba danych w partii

Zwraca:
----------
DataFrame ze średnim wynikiem dla grupy uczącej i
 ↪  walidacyjnej oraz ich odchylenia
"""
ann = tf.keras.models.Sequential()
for i in range(hidden_layers):
ann.add(tf.keras.layers.Dense(units=units,
 ↪  activation='relu'))
ann.add(tf.keras.layers.Dense(units=1))
ann.compile(optimizer = 'adam', loss =
 ↪  'mean_squared_error')

kfold = KFold(n_splits=5, random_state=42,
 ↪  shuffle=True)
train_mse = []
```

```
28                  valid_mse = []
29                          # trenowanie modelu i obliczenie metryk
30                  ann.fit(X_train_fold, y_train_fold, epochs=epochs,
                     ↪           batch_size=batch_size, verbose=0)
31                  train_pred = ann.predict(X_train_fold, verbose=0)
32                  test_pred = ann.predict(X_test_fold, verbose=0)
33                  train_mse.append(mean_squared_error(y_train_fold,
                     ↪  train_pred))
34                  valid_mse.append(mean_squared_error(y_test_fold,
                     ↪  test_pred))
35
36                  results_df = pd.DataFrame({
37                  'Warstwy ukryte' : hidden_layers,
38                  'Neurony w warstwach ukrytych' : units,
39                  'Średnia z wyników dla grupy uczącej':
                     ↪  np.mean(train_mse),
40                  'Odchylenie std dla grupy uczącej': np.std(train_mse),
41                  'Średnia z wyników dla grupy walidacyjnej':
                     ↪  np.mean(valid_mse),
42                  'Odchylenie std dla grupy walidacyjnej':
                     ↪  np.std(valid_mse),
43                  }, index=[0])
44                  return results_df
45                  #wywołanie sieci
46                  ann_score_df = eval_ann(2, 6, X_train, y_train, 100,
                     ↪  32)
```

Kod 12. Implementacja sztucznej sieci neuronowej

Wyniki osiągnięte przez sieć dla pięciokrotnej walidacji krzyżowej prezentują się w sposób następujący:

- **Średni wynik MSE grupy uczącej**: 24.20
- **Średni wynik MSE grupy walidacyjnej**: 24.26
- **Różnica między grupą uczącą, a walidacyjną**: 0.06

6.2.2 Analiza porównawcza i wybór najlepszego modelu

Analiza porównawcza modeli odbyła się w oparciu o uśredniony wynik błędu średniokwadratowego oraz odchylenie standardowe tego błędu dla grupy walidacyjnej, które zostały wyznaczone na podstawie pięciokrotnej walidacji krzyżowej. Uśredniony błąd *MSE* wskazuje na skuteczność predykcji modelu. Im mniejsza wartość, tym bardziej dokładny jest dany model w przewidywaniu danych. Wartość odchylenia dla błędu *MSE* określa stabilność modelu. Im wartość odchylenie jest mniejsza, tym stabilność modelu wzrasta. Porównanie metod prezentuje **Tab. 6.3**.

Algorytm	**Średni wynik MSE**	**Odchylenie std MSE**
Regresja Liniowa	20.92	1.02
K-najbliższych sąsiadów	18.36	0.85
Drzewo decyzyjne	19.87	1.24
Las losowy	17.85	1.10
Wzmocnienie gradientu	15.12	1.00
Sztuczna sieć neuronowa	24.26	1.80

Tab 6.3. Analiza porównawcza rozważanych algorytmów dla grupy walidacyjnej z zastosowaniem metody pięciokrotnej walidacji krzyżowej

W analizowanej tabeli można zauważyć, że Wzmocnienie gradientu osiąga najniższą wartość średniego błędu *MSE* z analizowanych algorytmów. Wartość odchylenie standardowego dla błędu MSE dla tego algorytmu ma również stosunkowo niską wartość. Algorytmy K-najbliższych sąsiadów oraz Lasu losowego osiągnęły również niski średni błąd, co oznacza, że w równym stopniu mogą być skutecznymi modelami predykcyjnymi, z czego algorytm K-najbliższych sąsiadów uzyskuje najniższą wartość odchylenia standardowego ze wszystkich analizowanych algorytmów, co pokazuje stabilność tego algorytmu.

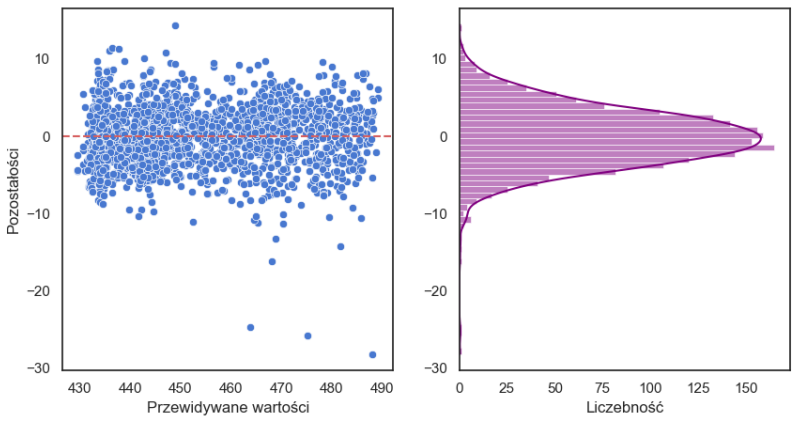
W przeciwieństwie do wspomnianych wcześniej modeli, zarówno Regresja liniowa, jak i Sztuczna sieć neuronowa, charakteryzują się wysokim uśrednionym błędem *MSE*. Szczególnie sieć neuronowa, której błąd jest najwyższy w stosunku do pozostałych algorytmów. Dodatkowo sztuczna sieć wykazała się najwyższym odchyleniem, co wskazuje na wysoką zmienność modelu.

Na podstawie analizy tabeli sugerowanym algorytmem, który wydaje się być najskuteczniejszy dla analizowanego problemu, cechując się zarówno dokładnością predykcji i stabilnością jest Wzmocnienie gradientu. Dlatego też został on wybrany jako finalny model, w swojej dostrojonej formie, gdzie współczynnik uczenia równa się 0.1, liczba estymatorów równa się 50 oraz maksymalna głębokość drzewa równa się 4.

6.3 Ewaluacja końcowego modelu

Do wytrenowania i ewaluacji wybranego modelu Wzmocnienia gradientu posłużyły dane pochodzące ze wstępnego podziału na zestaw uczący oraz zestaw testujący w stosunku 80 : 20, co stanowi 7654 obserwacji w zestawie uczącym i 1914 obserwacji w zestawie treningowym.

W procesie ewaluacji modelu zostało przeprowadzone badanie pozostałości, inaczej reszt (*ang. residuals*), które stanowią różnicę między wartością rzeczywistą, a wartością przewidywaną przez model. Analiza pozostałości jest jedną z metod, która pozwala ocenić jakość modelu oraz ewentualne problemy. W ramach tej analizy zostały zastosowane dwie metody wizualizacyjne, którymi są wykres pozostałości oraz rozkład pozostałości. Wykres pozostałości przedstawia zależność między pozostałościami, a wartościami przewidywanymi przez model, natomiast rozkład pozostałości prezentuje w jaki sposób te wartości są rozłożone. Metody te prezentuje **Rys. 6.7.**

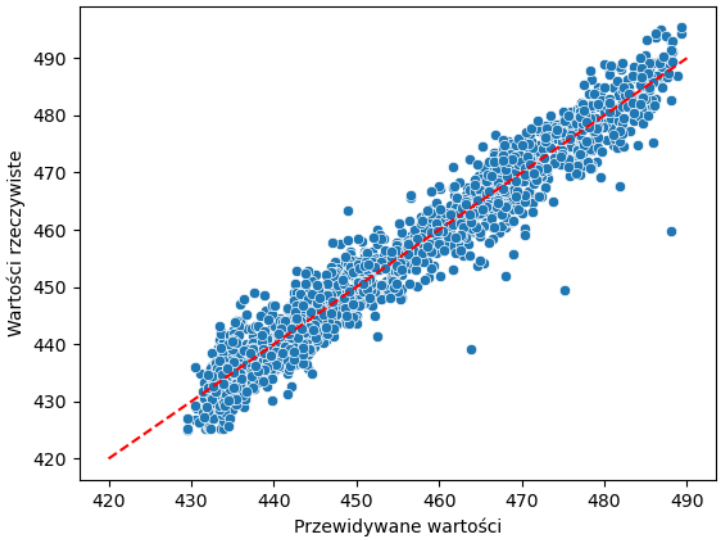


Rys. 6.7. Wykresy prezentujące zestawienia pozostałości, po lewej wykres pozostałości i po prawej rozkład

Na powyższych wykresach można zaobserwować, że wartości pozostałości wykazują względną symetrię wokół wartość 0. Na całym przedziale wartości przewidywanych reszty utrzymują podobną wartość, bez znaczących wzrostów, czy spadków. Dla większości

obserwacji, wartości pozostałości zawierają się w zakresie $[-10, 10]$. Można zauważyć, że występują trzy punkty wyraźnie wykraczające poza ten zakres. Znajdują się one w przedziale $[-25, -27]$, co wskazuje na to, że model dla tych obserwacji przewidział zbyt wysoką wartość. Wartości reszt charakteryzują się liniową charakterystyką wzdłuż wartości pozostałości 0, a także rozkładem zbliżonym do rozkładu normalnego.

Kolejnym punktem ewaluacji modelu, było wizualne zestawienie wartości przewidywanych przez model i wartości rzeczywistych. Jakość modelu na podstawie tego wykresu określana jest poprzez ułożenie punktów względem przekątnej, linii prostej o nachyleniu 45°. Im punkty na wykresie są bardziej skupione wzdłuż przekątnej, tym jakość modelu jest wyższa. Wykres ten dla analizowanego modelu Wzmocnienia gradientu prezentuje **Rys. 6.8.**.



Rys. 6.8. Wykres prezentujący zestawienie wartości przewidywanych i wartości rzeczywistych

Pierwszym wnioskiem wynikającym z obserwacji powyższego wykresu jest to, że punkty są rozłożone symetrycznie wokół przekątnej w sposób równomierny w całym zakresie wartości, posiadając wyraźnie liniową charakterystykę. Znaczna większość punktów skupia się w stosunkowo bliskim położeniu względem przekątnej. Można jednak zaobserwować trzy charakterystyczne wartości, które odstają od większości punktów.

Ostatnim elementem oceny modelu było zastosowanie miar liczbowych, którymi są błąd średniokwadratowy *RMSE*, średni błąd bezwzględny *MAE* oraz współczynnik determinacji R^2. Wyniki wyszczególnionych miar dla analizowanego modelu są następujące:

- **RMSE**: 3.80

- **MAE**: 2.93
- $\mathbf{R^2}$: 0.95

Współczynnik determinacji R^2 wynosi 0.95 w skali $[0, 1]$, co oznacza , że model wyjaśnia 95% zmienności występującej w danych. Wartości *RMSE* i *MAE* są wartościami niskimi i wynoszą kolejno 3.80 oraz 2.93. Nieznaczna różnica między tymi wartościami oznacza, że w danych występują wyróżniające się wartości błędów, co pokazały wcześniejsze analizy wizualizacyjne.

Podsumowując zarówno na podstawie analizy opierające się na technikach wizualizacyjnych, jak i na podstawie analizy miar można założyć, że model charakteryzuje się wysoką dokładnością predykcji w całym zakresie wartości zmiennej zależnej y, co świadczy o odpowiednim dopasowaniu modelu do danych oraz ogólności tego modelu.

7 Podsumowanie

Projekt inżynierski miał na celu zaproponowanie modelu predykcyjnego do wyznaczenia wielkości mocy umownej w zakładzie przemysłowym, którym była elektrownia gazowo-parowa pracująca w cyklu kombinowanym. Zbiór danych służący do wytrenowania modelu składał się z 9568 obserwacji i obejmował cztery zmienne niezależne X, którymi były *Temperatura (AT)*, *Ciśnienie otoczenia (AP)*, *Wilgotność względna (RH)*, *Podciśnienie w układzie wydechowym (V)* oraz zmienna zależna, celu y, którą była *Moc pobrana (EP)*. W oparciu o dane zostało przetestowanych sześć metod uczenia maszynowego, którymi były *Regresja Liniowa*, *K-najbliższych sąsiadów*, *Las losowy*, *Wzmocnienie gradientu*, oraz *Sztuczna sieć neuronowa*, których implementacje pochodziły z biblioteki *Scikit-learn* oraz *Kears*. Na podstawie analizy porównawczej rozważanych metod, jako najlepszy model została zaproponowana technika Wzmocnienia gradientu (*ang. Gradient Boosting*), z takimi parametrami jak współczynnik uczenia z wartością 0.1, liczba estymatorów równa 50 oraz maksymalna głębokość drzewa równa 4. Wykorzystaną miarą w procesie wyboru najlepszego modelu był błąd średniokwadratowy *MSE*. Dobrana technika została poddana procesowi ewaluacji, który opierał się na wizualnym przedstawieniu pozostałości, reszt *(ang. residuals)*, wizualnym przedstawieniu wartości przewidywanych w zestawieniu z rzeczywistymi wartościami oraz trzech miarach, którymi były pierwiastek błędu średniokwadratowego *RMSE*, średni błąd bezwzględny *MAE* i współczynnik determinacji R^2. Na podstawie uzyskanych wyników można stwierdzić, że model Wzmocnienia gradientu jest precyzyjnym, skutecznym i stabilnym modelem i mógłby być wdrożony do praktycznych zastosowań. Model osiągnął wysoką precyzję predykcji dla zróżnicowanych danych.

Bibliografia

[1] Rafik Nafkha, Tomasz Ząbkowski i Krzysztof Gajowniczek. "Two Stage Approach to Optimize Electricity Contract Capacity Problem for Commercial Customers". W: *Computational Science – ICCS 2021. 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part IV* 12745 (2021). Red. M. Paszynski i in., s. 173–184. DOI: 10.1007/978-3-030-77970-2_14.

[2] Rafik Nafkha i Dariusz Strzęciwilk. "Optimizing the contractual capacity volume to minimize the network financial charge". W: *Computer Algebra Systems in Teaching and Research. Vol. X: Applications in Mathematical Physics and Mechanics* X (2021). Red. Alexander Prokopenya, Dorota Kozak-Superson i Marek Siłuszyk, s. 137–150.

[3] Minister Gospodarki. "Rozporządzenie Ministra Gospodarki z dnia 4 maja 2007 r. w sprawie szczegółowych warunków funkcjonowania systemu elektroenergetycznego". W: *Dziennik Ustaw* 93 (2007).

[4] Minister Klimatu i Środowiska. "Rozporządzenie Ministra Klimatu I Środowiska z dnia 29 listopada 2022 r. w sprawie sposobu kształtowania i kalkulacji taryf oraz sposobu rozliczeń w obrocie energią elektryczną". W: *Dziennik Ustaw* (2022).

[5] Manning Christopher. *AI Definitions*. 2020. URL: https://hai.stanford.edu/sites/default/files/2020-09/AI-Definitions-HAI.pdf (term. wiz. 11. 04. 2023).

[6] Dennis Michael Aaron. *Marvin Minsky*. 2023. URL: https://www.britannica.com/biography/Marvin-Lee-Minsky (term. wiz. 11. 04. 2023).

[7] Zhi-Hua Zhou. *Machine Learning*. Singapore: Springer Singapore, 2021, s. 1–24. ISBN: 978-981-15-1967-3. DOI: 10.1007/978-981-15-1967-3.

[8] Sebastian Miśtak. *Podział modeli uczenia maszynowego wraz z przykładami zastosowania*. 2022. URL: https://www.gov.pl/web/popcwsparcie/podzial-modeli-uczenia-maszynowego-wraz-z-przykladami-zastosowania (term. wiz. 11. 04. 2023).

[9] Pádraig Cunningham, Matthieu Cord i Sarah Jane Delany. *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 21–49. ISBN: 978-3-540-75171-7. DOI: 10.1007/978-3-540-75171-7_2.

[10] Adam Meissner. “Elementy uczenia maszynowego”.
W: *Instytut Automatyki i Inżynierii Informatycznej Politechniki Poznańskiej* (2016).

[11] YCAP Reddy, P Viswanath i B Eswara Reddy.
“Semi-supervised learning: A brief review”.
W: *Int. J. Eng. Technol* 7.1.8 (2018), s. 81.

[12] Richard S Sutton i Andrew G Barto. *Reinforcement learning: An introduction*.
MIT press, 2018, s. 1–13. ISBN: 9780262039246.

[13] Marcin Luckner. *Podstawy Przetwarzania Danych, Wykład 6: Braki w danych*. 2021.
URL: https://pages.mini.pw.edu.pl/~lucknerm/wp-content/uploads/2021/03/HandoutPodstawyPrzetwarzaniaDanychWyklad05.pdf (term. wiz. 11. 04. 2023).

[14] Tomasz Kulpa. “Metody uzupełniania brakujących danych na przykładzie liczby zarejestrowanych pojazdów”.
W: *Transport Miejski i Regionalny* 10 (2013), s. 22–25.

[15] Chris Albon. *Uczenie maszynowe w Pythonie. Receptury*. Helion, 2018,
s. 73–88, 91–99. ISBN: 978-83-283-5046-5.

[16] Adam Walanus. *Czym się różni sześć sigma od trzech sigma?* 2005.
URL: https://media.statsoft.pl/_old_dnn/downloads/czym_sie_rozni_szesc_sigma_od_trzy_sigma.pdf (term. wiz. 11. 04. 2023).

[17] Piotr Denderski. *Metody identyfikacji obserwacji odstających*. 2019.
URL: https://web.sgh.waw.pl/~gkoloch/pliki/Podypl/outliers.pdf (term. wiz. 11. 04. 2023).

[18] IBM Corporation. *Modified Z-score*. 2021.
URL: https://www.ibm.com/docs/pl/cognos-analytics/11.1.0?topic=terms-modified-z-score (term. wiz. 11. 04. 2023).

[19] H. P. Vinutha, B. Poornima i B. M. Sagar.
“Detection of Outliers Using Interquartile Range Technique from Intrusion Dataset”.
W: *Information and Decision Sciences*. Red. Suresh Chandra Satapathy i in.
Singapore: Springer Singapore, 2018, s. 511–518. ISBN: 978-981-10-7563-6.

[20] Adrian Horzyk. *Metody Inżynierii Wiedzy: Transformacje i jakość danych*. 2016.
URL: https://home.agh.edu.pl/~horzyk/lectures/miw/MIW-TransformacjeJakoscDanych.pdf (term. wiz. 11. 04. 2023).

[21] Wikipedia. *Training, validation, and test data sets*. 2023. URL: https://en.wikipedia.org/wiki/Training,_validation,_and_test_data_sets (term. wiz. 11. 04. 2023).

[22] Hailiang Du. *Machine Learning and Neural Networks: Chapter 4: Model Assessment and Selection*. 2020. URL: https://bookdown.org/hailiangdu80/Machine_Learning_and_Neural_Networks/model-assessment-and-selection.html (term. wiz. 11. 04. 2023).

[23] Wikipedia. *Hyperparameter optimization*. 2023. URL: https://en.wikipedia.org/wiki/Hyperparameter_optimization (term. wiz. 11. 04. 2023).

[24] Ibrahim Abayomi Ogunbiyi. *Top Evaluation Metrics for Regression Problems in Machine Learning*. 2022. URL: https://www.freecodecamp.org/news/evaluation-metrics-for-regression-problems-machine-learning/ (term. wiz. 11. 04. 2023).

[25] Scott Fortmann-Roe. "Understanding the bias-variance tradeoff". W: *URL: http://scott. fortmann-roe. com/docs/BiasVariance. html (hämtad 2019-03-27)* (2012).

[26] Shayan Doroudi. "The bias-variance tradeoff: How data science can inform educational debates". W: *AERA open* 6.4 (2020), s. 1–18. DOI: 10.1177/2332858420977208.

[27] Trevor Hastie i in. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, s. 28, 289–335, 337–384, 587–603. ISBN: 9780387848587.

[28] Giuseppe Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017, s. 72–93. ISBN: 978-1-78588-962-2.

[29] Sadegh Bafandeh Imandoust, Mohammad Bolandraftar i in. "Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background". W: *International journal of engineering research and applications* 3.5 (2013), s. 605–610.

[30] Grzegorz Dudek. "OPTYMALIZACJA MODELI PROGNOSTYCZNYCH OPARTYCH NA ESTYMATORACH NAJBLIŻSZEGO SĄSIEDZTWA". W: *Instytucie Elektroenergetyki Politechniki Czestochowskiej* (2012).

[31] Hung Son Nguyen. *Systemy decyzyjne*. Wydział Matematyki, Informatyki i Mechaniki UW. URL: https://mst.mimuw.edu.pl/lecture.php?lecture=syd&part=Ch6 (term. wiz. 11. 04. 2023).

[32] The Pennsylvania State University. *Regression Trees*. 2023. URL: https://online.stat.psu.edu/stat508/lesson/11/11.8/11.8.2 (term. wiz. 11. 04. 2023).

[33] Zdzisław Stęgowski. “Sztuczne sieci neuronowe”.
W: *Kraków: Wydawnictwo Naukowo-Techniczne* (2004), s. 16–19.

[34] Mark Lutz. *Python. Wprowadzenie*. Helion, 2020, s. 53–60.
ISBN: 978-83-283-6150-8.

[35] Charles R. Harris i in. “Array programming with NumPy”.
W: *Nature* 585 (2020), s. 357–362. DOI: 10.1038/s41586-020-2649-2.

[36] Wes McKinney. “Data Structures for Statistical Computing in Python”.
W: *Proceedings of the 9th Python in Science Conference*.
Red. Stéfan van der Walt i Jarrod Millman. 2010, s. 56–61.
DOI: 10.25080/Majora-92bf1922-00a.

[37] F. Pedregosa i in. “Scikit-learn: Machine Learning in Python”.
W: *Journal of Machine Learning Research* 12 (2011), s. 2825–2830.

[38] Francois Chollet i in. *Keras*. 2015.
URL: https://github.com/fchollet/keras (term. wiz. 11. 04. 2023).

[39] J. D. Hunter. “Matplotlib: A 2D graphics environment”.
W: *Computing in Science & Engineering* 9.3 (2007), s. 90–95.
DOI: 10.1109/MCSE.2007.55.

[40] Michael L. Waskom. “Seaborn: statistical data visualization”.
W: *Journal of Open Source Software* 6.60 (2021), s. 3021.
DOI: 10.21105/joss.03021.

[41] Krzysztof Karpio, Piotr Łukasiewicz, Rafik Nafkha i in.
“Description of Electricity Consumption by Using Leading Hours Intra-day Model”.
W: *Computational Science – ICCS 2021. 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part IV* 12745 (2021). Red. M. Paszynski i in., s. 392–404. DOI: 10.1007/978-3-030-77970-2_30.

[42] Krzysztof Karpio, Piotr Łukasiewicz i Rafik Nafkha. “Regression Technique for Electricity Load Modeling and Outlined Data Points Explanation”. W: *Advances in Soft and Hard Computing* 889 (2019). Red. Jerzy Pejaś i in., s. 56–67.
DOI: 10.1007/978-3-030-03314-9_5.

[43] Rafik Nafkha, Tomasz Ząbkowski i Krzysztof Gajowniczek.
“Deep Learning-Based Approaches to Optimize the Electricity Contract Capacity Problem for Commercial Customers”. W: *Energies* 14.8 (2021), s. 1–17.
DOI: 10.3390/en14082181.

Wyrażam zgodę na udostępnienie mojej pracy w czytelniach Biblioteki SGGW w tym w Archiwum Prac Dyplomowych SGGW.

. . . . . .
(czytelny podpis autora pracy)