UNIWERSYTET
MIKOŁAJA KOPERNIKA
W TORUNIU

Eksploracja Danych

Projekt zaliczeniowy – białe wina

Konrad Domeradzki

Toruń 2023

Spis treści

1. Wprowadzenie

1.1. Problematyka

Problemem jaki poruszymy w tym projekcie to ocenianie jakości wina poprzez jego wyniki testów fizyczno-chemicznych. W tym celu zostaną stworzone dwa modele, klasyfikacji oraz szacowania. Poza tym zostanie stworzone grupowanie, po czym zostanie sprawdzone powiązanie grupowania wraz z oceną jakości wina. Projekt jest wykonywany na potrzeby zajęć **Eksploracyjna analiza danych**, studiowanych na ***Uniwersytecie Mikołaja Kopernika na Wydziale Informatyki i Matematyki***.

1.2. Zbiór danych

Dane, na których został oparty ten raport to wyniki testów fizyko-chemicznych białego wina. Plik zawiera 3934 rekordów, a zmienną celu, czyli wartością, którą chcemy przewidywać jest **quality**. Opis danych z naszego zbioru wygląda następująco:

- **Fixedacidity** – stała kwasowość
- **Volatileacidity** – lotna kwasowość
- **Citricacid** – kwasek cytrynowy
- **Residualsugar** – cukier resztkowy
- **Chlorides** – chlorki
- **Freesulfurdioxide** – wolny dwutlenek siarki
- **Totalsulfurdioxide** – całkowity dwutelek siarki
- **Density** – gęstość
- **pH** – chemiczne pH
- **Sulphates** – siarczany
- **Alcohol** – procent alkoholu w winie
- **Quality** – jakość wina (ocena od 1 do 7)

1.3. Plan działania

Nasz plan opiera się na przeprowadzeniu eksploracyjnej analizy danych na zbiorze, który dostaliśmy, następnie stworzymy model szacowania metodą MLP oraz model klasyfikacji metodą XGBoost. Modele zostaną zoptymalizowane po czym porównane. Następnie zostanie stworzone grupowanie. Zostanie ono ocenione, scharakteryzowane oraz na koniec sprawdzimy czy ma powiązanie z jakością wina.

2. Eksploracyjna analiza danych

2.1. Analiza wartości, badanie występowania braków danych i usuwanie duplikatów

Eksploracyjna analiza danych była kluczowym etapem pracy, obejmując badanie wartości, identyfikację braków danych oraz usuwanie ewentualnych duplikatów w zbiorze. Po dokładnym przeglądzie danych, stwierdzono, że dane są źle sformatowane dla języka Python, elementem rozdzielającym wartość całkowitą od dziesiętnej był przecinek, w języku Python jest to kropka. Dane zostały wczytane w większości jako obiekty. Przy wartościach które powinny mieć wartość na przykład 0.45 była wpisana wartość ,45. Dane trzeba było przekonwertować oraz pozamieniać przecinki na kropki. Wszystki zmienne zostały przekonwertowane na wartości typy float64 (oprócz zmiennej celu). Poza tym nie wykazano braków w danych.

Dodatkowo, sprawdzono, czy w zbiorze nie występują duplikaty, a analiza wykazała sporą ich ilość. Z 3934 rekordów powtórzyło się 645. Zostały one usunięte ze wzlędu na możliwość przekłamywania wyników. W zbiorze pozostało 3289 rekordów.

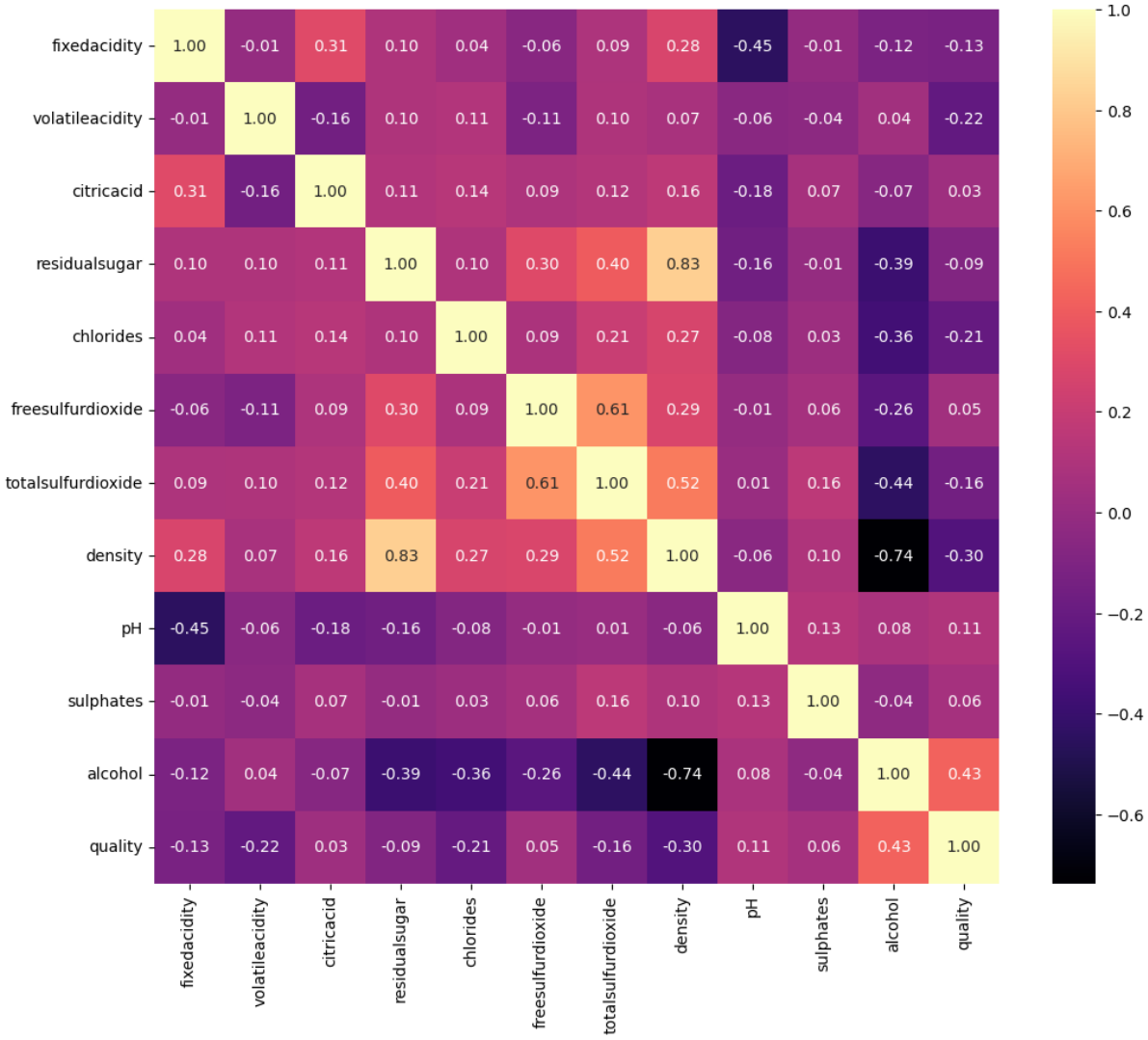
Eksploracyjna analiza danych została wykonana na zbiorze danych uczących.

Podczas eksploracyjnej analizy danych, dla każdej zmiennej przeprowadziliśmy szczegółową analizę statystyczną. W naszych danych znaleźliśmy tylko zmienne ilościowe i dla każdej z nich policzyliśmy średnią, odchylenie standardowe, minimum, maksimum oraz kwantyle.

Po przeanalizowaniu podstawowych statystyk naszych danych możemy zauważyć pewne własności. Pierwszą z nich jest fakt, że średnie oraz miediany mają bardzo zbliżone do siebie wartości, może to sugerować dobre zróżnicowanie danych bez nieporządanych dziur i proporcjonalną ilość obserwacji ze skrajnymi wartościami.

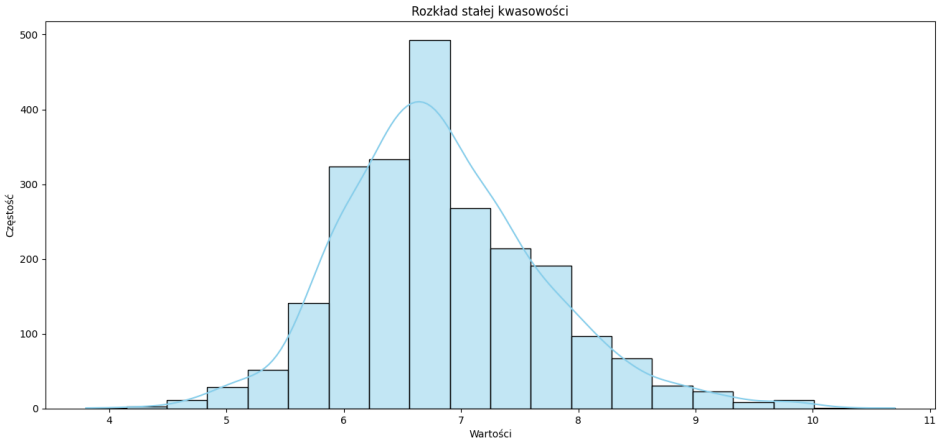
Można również spostrzec dużą amplitudę w wartościach niektórych zmiennych, takich jak: lotna kwasowość, kwas cytrynowy, cukier resztkowy, chlorki, wolny i całkowity dwutlenek siarki. Ciężko powiedzieć w jaki sposób oddziałuje to na ocenę jakości wina, z tego co można śmiało powiedzieć to fakt, że ważnymi elementami w ocenie wina są na pewno kwasowość oraz cukier resztkowy. Poza tym najważniejszymi elementami przy ocenie są alkohol, siarczany oraz gęstość.

2.2. Budowa macierzy korelacji predyktorów

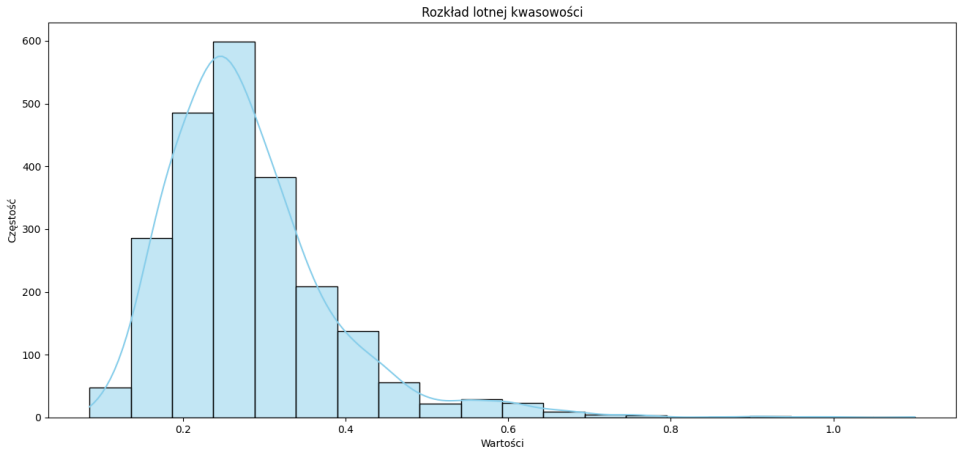


Z analizy korelacji naszych zmiennych wynika, że mocno skorelowane są ze sobą takie pary jak: {gęstość, cukier resztkowy}, {gęstość, alkohol}, {gęstość, całkowity dwutlenek siarki}, {wolny dwutlenek siarki, całkowity dwutlenek siarki}. Poza tym wiemy, że zmienna celu najmocniej jest skorelowana ze zmiennymi: alkohol, gęstość z odpowiednio wartościami korelacji: 0.43 oraz -0.30. Będą to najprawdopodobniej dwa najważniejsze predyktory.

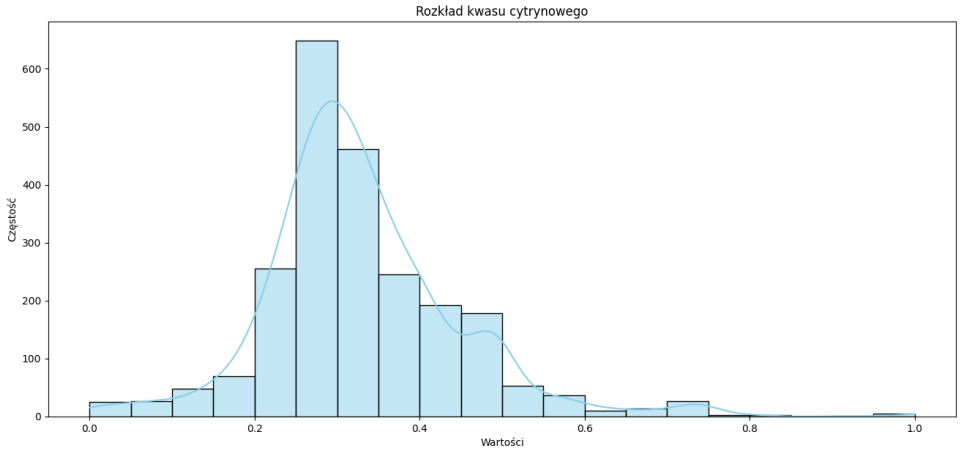
2.3. Analiza statystyczna



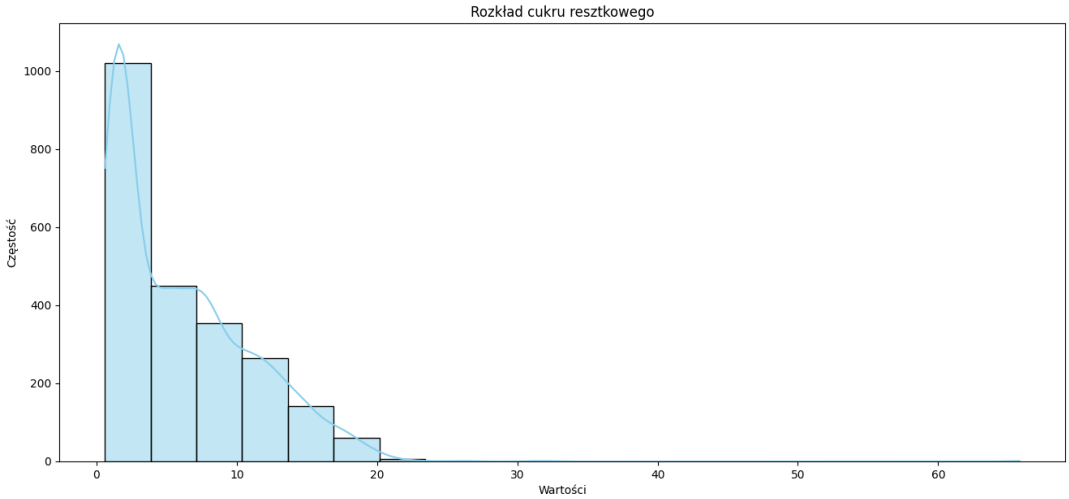
Rozkład stałej kwasowości jest bardzo zbliżony do rozkładu normalnego.



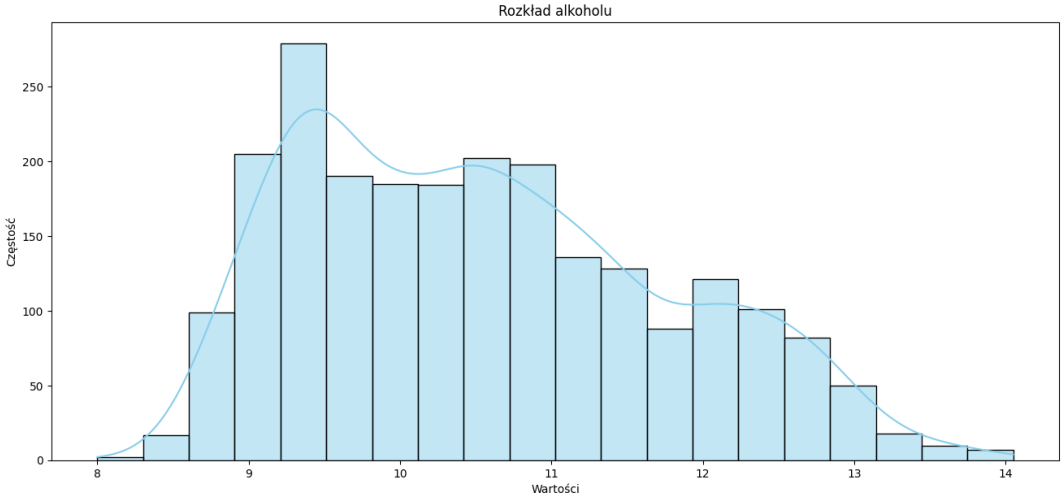
W przypadku zmiennej lotna kwasowość istnieje wyraźna asymetria prawostronna..



W przypadku kwasu cytrynowego również widać asymetrię prawostronną, poza tym widać niewielkie zaburzenia rozkładu normalnego przy wartości około 0.5.



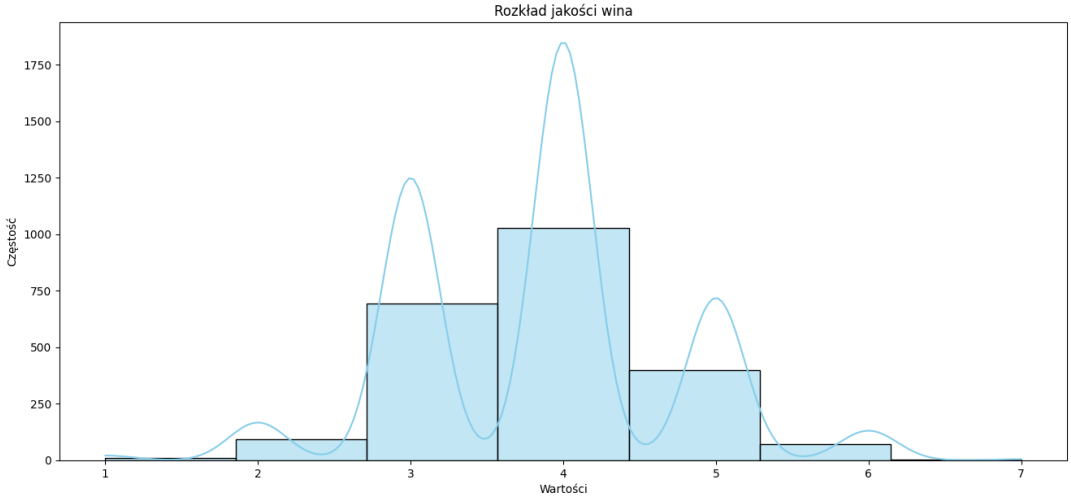
Cukier resztkowy nie wykazuję rozkładu normalnego, bardziej przypomina to stosunek odwrotnie proporcjonalny, im większy poziom cukru tym mniejsza ilość takich obserwacji.



Rozkład alkoholu odbiega od rozkładu normalnego.

Pozostałe zmienne posiadają rozkład normalny, bądź rozkład normalny z asymetrią prawostronną.

Spójrzmy jeszcze na rozkład zmiennej celu.



Widać zaburzenie w krzywej rozkładu, wiadome jest, że wartości są całkowite, aczkolwiek w danych są oznaczone teraz jako wartości rzeczywsite, z tego względu sprawdza nam to również wartości takie jak 3.5 które nie istnieją. Można jednak samemu śmiało stwierdzić, że nasza zmienna celu ma rozkład mocno przypominający rozkład normalny.

2.4. Wnioski

Po starannej analizie danych stwierdzono, by pozostawić wszystkie zmienne jako predyktory, nie wykazują one między sobą aż tak wysokich korelacji ani nie dają podstaw do odrzucenia ich ze względu na zbyt słaby wpływ na zmienna celu.

3. Model szacowania - Multilayer Perceptron (MLP)

3.1. Wprowadzenie

Wielowarstwowy perceptron (MLP) to model sztucznej sieci neuronowej, stosowany do rozwiązywania zadań klasyfikacyjnych i regresyjnych. W tym dokumencie skupimy się na jego zastosowaniu w problemie regresji. MLP, składający się z warstw wejściowej, ukrytych i wyjściowej, przechodzi przez fazy propagacji w przód i wstecznej propagacji, dostosowując wagi neuronów na podstawie błędu prognozy.

3.2. Normalizacja predyktorów

Normalizacja predyktorów stanowi kluczowy etap w przygotowaniu danych do sztucznych sieci neuronowych, zwłaszcza wielowarstwowego perceptronu (MLP). W procesie tym skorzystano z narzędzia z pakietu sklearn, a mianowicie **MinMaxScaler.**

MinMaxScaler został zastosowany do normalizacji zmiennych ilościowych, przekształcając wartości na przedział (-1, 1). Ten proces eliminuje problemy związane z różnicami w rozrzucie wartości między cechami, co pozwala uniknąć trudności wynikających z różnic w skali wartości pomiędzy różnymi cechami.

W rezultacie zastosowania MinMaxScaler, dane predykcyjne zostały odpowiednio znormalizowane. Ten proces przyczynia się do poprawy wydajności MLP, minimalizując wpływ różnic w skali wartości i umożliwiając bardziej efektywne uczenie się modelu.

3.3. Optymalizacja hiperparametrów modelu

W dążeniu do optymalizacji modelu, skorzystano z narzędzia **RandomizedSearchCV** z pakietu **sklearn**. Proces ten polegał na systematycznym przeszukiwaniu przestrzeni hiperparametrów w celu znalezienia konfiguracji, która maksymalizuje skuteczność modelu.

Przestrzeń hiperparametrów, którą eksplorowano, obejmowała różne ustawienia warstw ukrytych, funkcji aktywacji, współczynnika regularyzacji, tempa uczenia, liczby iteracji oraz solvera:

- **Liczba neuronów w warstwie ukrytej** - Testowano różne konfiguracje, takie jak (50,), (100,), (150,), (50, 50), (100, 100).

- **Funkcje aktywacji** - Eksperymentowano z funkcjami aktywacji, takimi jak 'logistic', 'tanh' oraz 'relu'.
- **Współczynnik regularyzacji (alpha)** - Testowano wartości alpha w zakresie [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001].
- **Tempo uczenia (learning_rate)** - Przetestowano różne opcje, w tym 'constant', 'invscaling', 'adaptive'.
- **Liczba iteracji (max_iter)** - Eksplorowano różne liczby iteracji, w tym 100, 200, 400, 700, 1000, 1500.
- **Solver** - Testowano różne solvery, takie jak 'adam', 'lbfgs', 'sgd'.

Algorytm **RandomizedSearchCV** został wywołany, aby losowo próbować różne kombinacje hiperparametrów. Przy użyciu 200 iteracji (n_iter=200) i 5-krotnej domyślnie walidacji krzyżowej (cv=5), algorytm przeszukał przestrzeń hiperparametrów, oceniając skuteczność modelu dla każdej kombinacji. Na podstawie otrzymanych wyników zbudowano model MLP.

3.4. Model zoptymalizowany

3.4.1. Konfiguracja modelu

Zmodyfikowano konfigurację modelu opartego na implementacji klasy **MLPRegressor** z pakietu **sklearn**. W wyniku procesu optymalizacji hiperparametrów, otrzymano konfigurację, której kluczowe parametry są następujące:

- **Solver ('adam') -** algorytm adam, efektywny dla dużych zbiorów danych
- **Liczba Iteracji (1000) -** 1000 iteracji dla lepszej konwergencji.
- **Tempo uczenia ('invscaling') -** 'invscaling' – stopniowo zmniejszany współczynnik uczenia.
- **Liczba neuronów w warstwie ukrytej ((150,)) -** jedna warstwa ukryta (150 neuronów).
- **Współczynnik regularyzacji (0.01) -** średni współczynnik regularyzacji (0.01) na rzecz kontroli przeuczenia.
- **Funkcja aktywacji ('relu') -** użyto funkcji max(0, x) dla warstw ukrytych.

Te modyfikacje wynikają z analizy przestrzeni hiperparametrów, w której wykorzystano klasę **RandomizedSearchCV**, co pozwoliło na dostrojenie modelu regresyjnego.

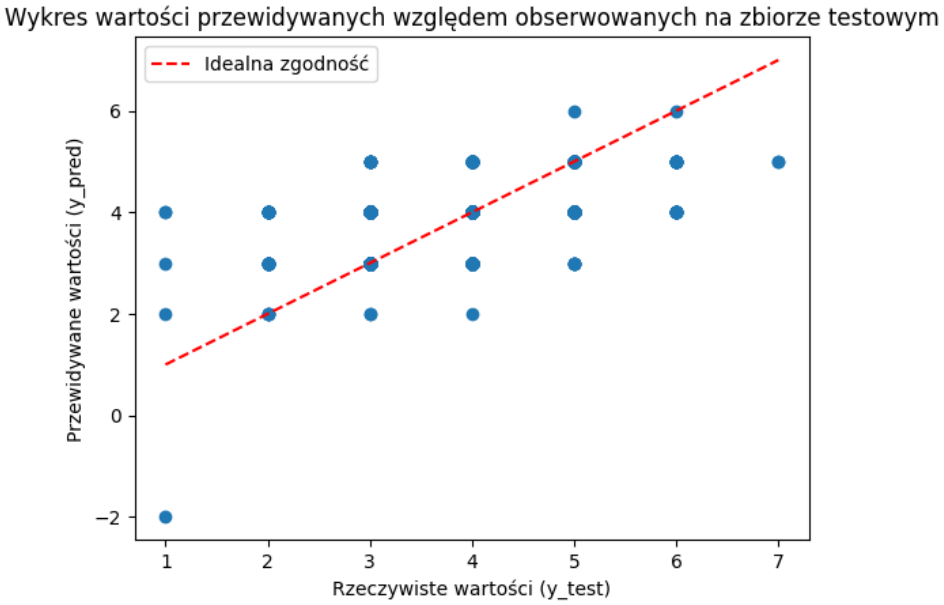
3.4.2. Ocena jakości modelu: miary jakości

Zbiór uczący	
Trafność	0.592
Trafność z odstępstwem o 1	0.967
Średni błąd bezwzględny (MAE)	0.444

Zbiór testowy	
Trafność	0.546
Trafność z odstępstwem o 1	0.949
Średni błąd bezwzględny (MAE)	0.508

Oceniając ogólną wydajność zoptymalizowanego modelu regresji, można jednoznacznie stwierdzić, że model działa nienajgorzej, ale nie dobrze. Model średnio myli się o pół oceny, oraz trafnie ocenia połowę obserwacji.

3.4.3. Ocena jakości modelu: wykres wartości przewidywanych względem obserwowanych



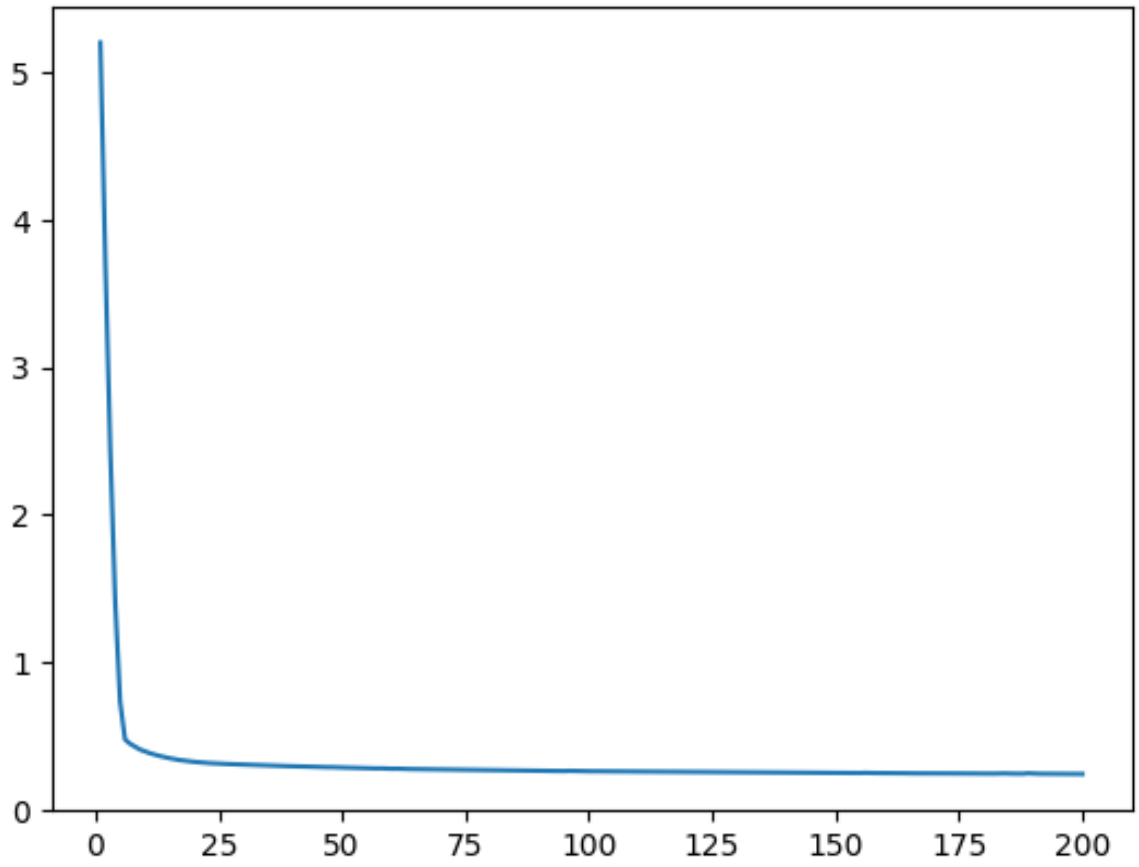
Analizując wykres wartości przewidywanych względem obserwowanych dla zoptymalizowanego modelu regresji, można zauważyć prawie brak poprawnie przewidzianych wartości 1 jak i 7, oceny skrajne w zbiorze występują tylko w kilku obserwacjach, jest to prawdopodona przyczyna dlaczego model słabo przewiduje te wartości. Poza tym model przewiduje z reguły w odstępstwie do 2 wartości. Zdarzyła się też jedna obserwacja o prawdziwej wartości 1 a przewidzianej -2, jest to zapewnie błąd modelu.

3.4.4. Budowa sieci

Budowa sieci	
Liczba warstw	3
Liczba neuronów w warstwie wejściowej	11
Liczba neuronów w warstwach ukrytych	(150,)
Funkcja aktywacji w warstwach ukrytych	Relu
Liczba neuronów w warstwie wyjściowej	1
Funkcja aktywacji w warstwie wyjściowej	identity

Analizując strukturę zoptymalizowanego modelu sieci neuronowej na podstawie dostarczonych danych, obserwujemy trzy warstwy, z których pierwsza to warstwa wejściowa z jedenastoma neuronami, co odpowiada ilości predyktorów. W warstwach ukrytych, istnieje jedna warstwa ze sto pięćdziesięcioma neuronami, gdzie zastosowano funkcję aktywacji relu. Warstwa wyjściowa zawiera jeden neuron, a jej funkcja aktywacji to identity, co wskazuje na to, że model jest zaprojektowany do bezpośredniego przewidywania wartości numerycznych bez wprowadzania dodatkowej nieliniowości.

3.4.5. Analiza zachowania funkcji straty



Funkcja straty dla modelu szacowania wykazuje wyraźny spadek z początkową fazą uczenia oraz wysoką stabilizację niskiej straty (0.5 jednostki) poprzez pozostały czas uczenia przy 200 epokach. Potwierdza nam wcześniejsze stwierdzenie o średniej pomyłce modelu o pół oceny.

3.4.6. Podsumowanie

Podsumowując, zoptymalizowany model regresji dobrze sprawuje się przy winach o średniej ocenie, natomiast nie przewiduje win bardzo słabych jak i arcydzieł. Model jest stabilny i można śmiało założyć, że przewidzi poprawnie wino z odstępstwem o 1 ocenę. Jest to sprawny model, który można wykorzystać przy większości win.

4. Model klasyfikacji - XGBoost

4.1. Wprowadzenie

Do stworzenia modelu klasyfikacji został wybrany algorymt XGBoost. Extreme Gradient Boosting to otwartoźródłowa biblioteka udostępniająca implementacje algorytmów uczenia maszynowego wykorzystujących wzmocnienie gradientowe. Była wykorzystywana w wielu projektach, które wygrywały prestiżowe zawody uczenia maszynowego. Bibliotekę trzeba było doinstalować, gdyż nie stanowi standardowej części dystrybucji Anacondy.

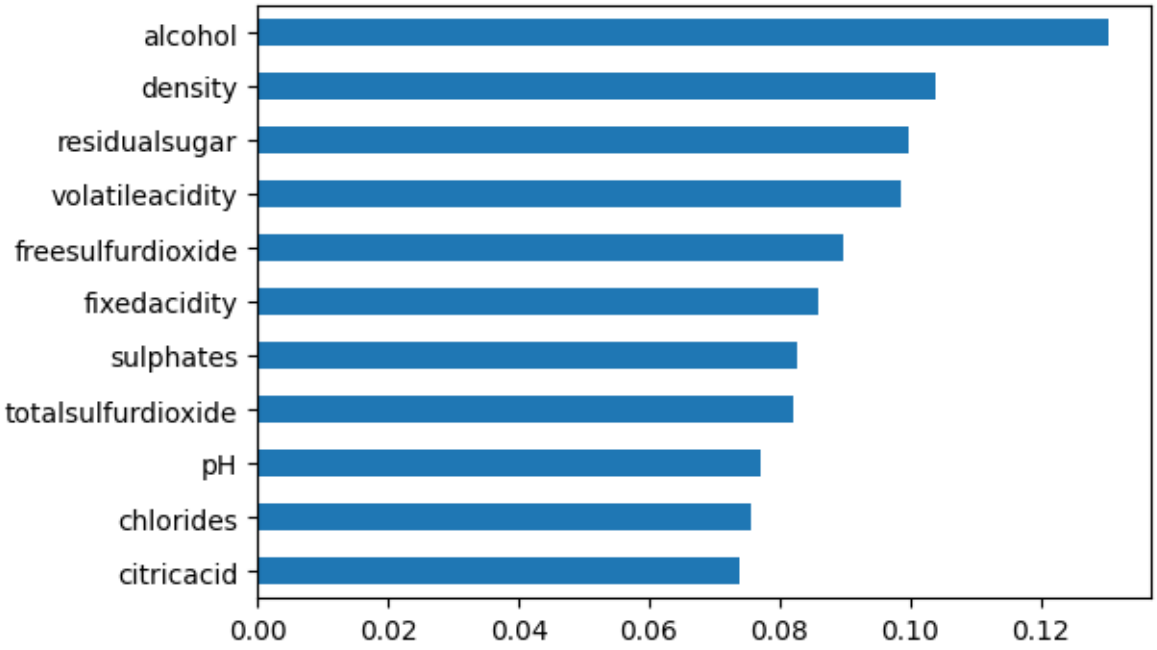
4.2. Model 1: pierwsza konfiguracja

Model został stworzony za pomocą klasy **XGBClassifier**, do stworzenia modelu zostały wykorzystane takie parametry o podanych wartościach:

- **objective = multi:softmax –** funkcja straty, która ma zostać zminimalizowania, multi:softmax jest wykorzystywany przy większej liczbie klas, dodatkowo wymagany w tym wypadku parametr to num_class
- **num_class = 7 –** ilość klas do przewidywania
- **n_estimators = 150 –** ilość rund boostingu
- **colsample_bytree = 0.3 –** odsetek zmiennych losowanych przy budowaniu pojedynczego drzewa
- **learning_rate = 0.1 –** tempo uczenia
- **max_depth = 20 –** maksymalna głębokość każdego drzewa
- **seed = 308161 –** parametr losowości, ustawiony na wartość indeksu

Do budowy modelu trzeba było przekonwertować wartości naszych klas, oceny jakości wina są z zakresy 1 – 7 natomiast do budowy modelu potrzebny był zbiór z zakresu zaczynającego się od 0, więc za pomocą klasy **LabelEncoder** wartość w tablicy y_train zawierającej dane ocenionych win zostały zamienione na te z zakresu 0 – 6.

4.2.1. Ważność predyktorów



Jak podejrzewaliśmy najważniejszymi predyktorami były alkohol oraz gęstość. Widać jednak, że jest dużo zmiennych które mają prawie takie samo znaczenie dla oceny modelu. Dokładniej mówiąc nie ma takiej zmiennej, która nie odegrałby istotnej roli w tworzeniu modelu.

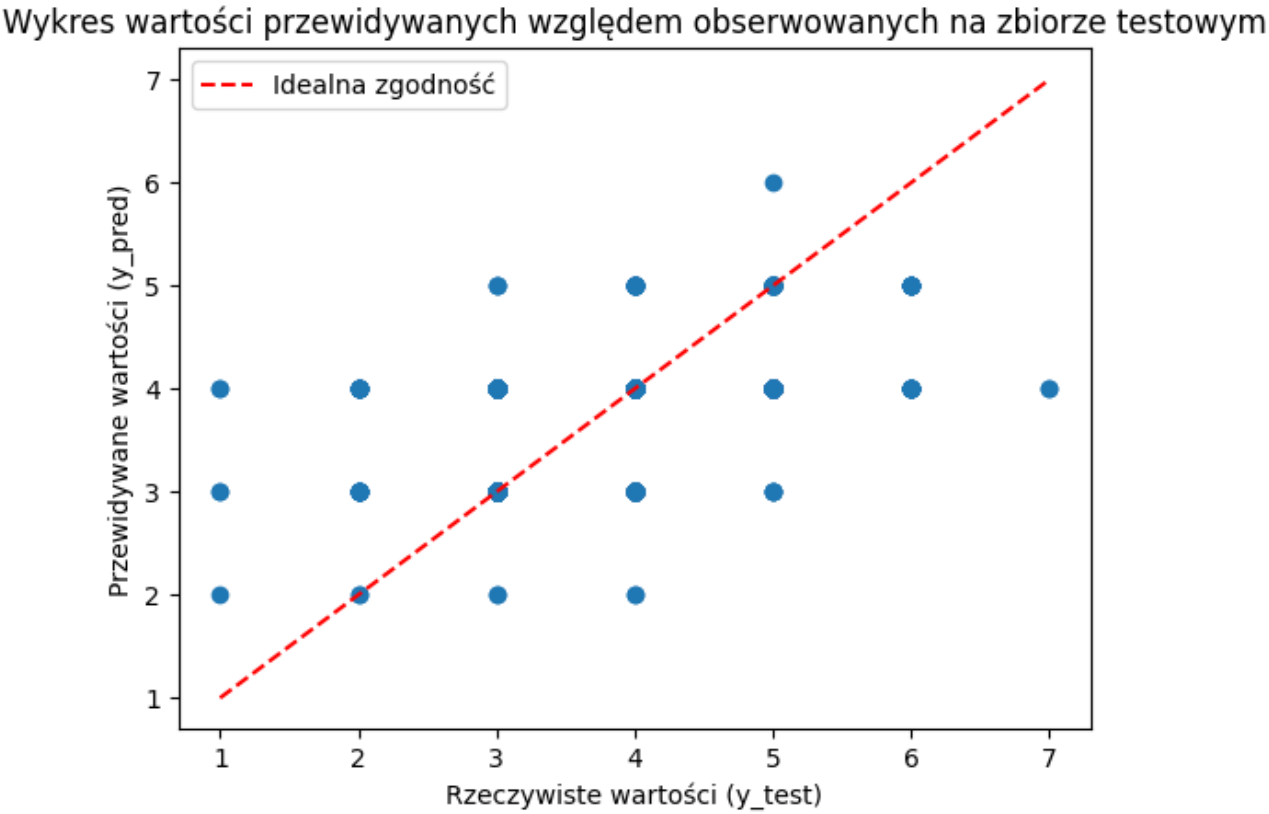
4.2.2. Ocena jakości modelu: miary jakości

Zbiór uczący	
Trafność	1.0
Trafność z odstępstwem o 1	1.0
Średni błąd bezwzględny (MAE)	0.0

Zbiór testowy	
Trafność	0.540
Trafność z odstępstwem o 1	0.945
Średni błąd bezwzględny (MAE)	0.519

Nasz model jest wyraźnie przeuczony dlatego zostanie on poprawiony.

4.2.3. Ocena jakości modelu: wykres wartości przewidywanych względem obser-wowanych



Narysowanie wykresu obserwacji rzeczywistych do przewidywanych pokazuje istotny fakt w modelu. Model nie przewiduje poprawnej oceny jakości wina dla win o ocenach skrajnych. Nie znalazły się obserwacje zaklasyfikowane jako wina o jakości: 1, 6 ani 7. Poza tym model klasyfikuje wina z najwyższymi ocenami jako wina o jakości 4 lub 5, czy to może oznaczać, że te wina mają bardzo podobne wyniki testów fizyko-chemicznych?

4.3. Model 2: Najlepsze hiperparametry

Poprawimy nasz model za pomocą klasy **RandomizedSearchCV**. Oto przestrzeń hiperparametrów jakie wykorzystamy do stworzenia zoptymalizowanego modelu.

- **n_estimators**: Testowano różne wartości, takie jak 100, 150, 300, 500.
- **max_depth**: Eksperymentowano z głębokościami jak: 5, 10, 15, 20.
- **grow_policy**: Testowano polityki jak: ‘depthwise’, ‘lossguide’.
- **subsample**: Przetestowano różne opcje, w tym: 0, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7.
- **gamma**: Eksplorowano różne wartości minimalnej ilości obserwacji potrzebnej do podzielenia węzła: 2, 3, 4, 5, 10, 15.
- **colsample_bytree**: Testowano różne próbki: 0.1, 0.2, 0.3, 0.4, 0.5.
- **learning_rate**: Sprawdzano takie tempa uczenia: 0.01, 0.02, 0.05, 0.1, 0.2, 0.3.
- **reg_alpha**: Spróbowano takie wartości regularyzacji: 0, 0.2, 0.4, 0.6, 0.8, 1.
- **reg_lambda**: Spróbowano takie wartości regularyzacji: 0, 0.2, 0.4, 0.6, 0.8, 1.

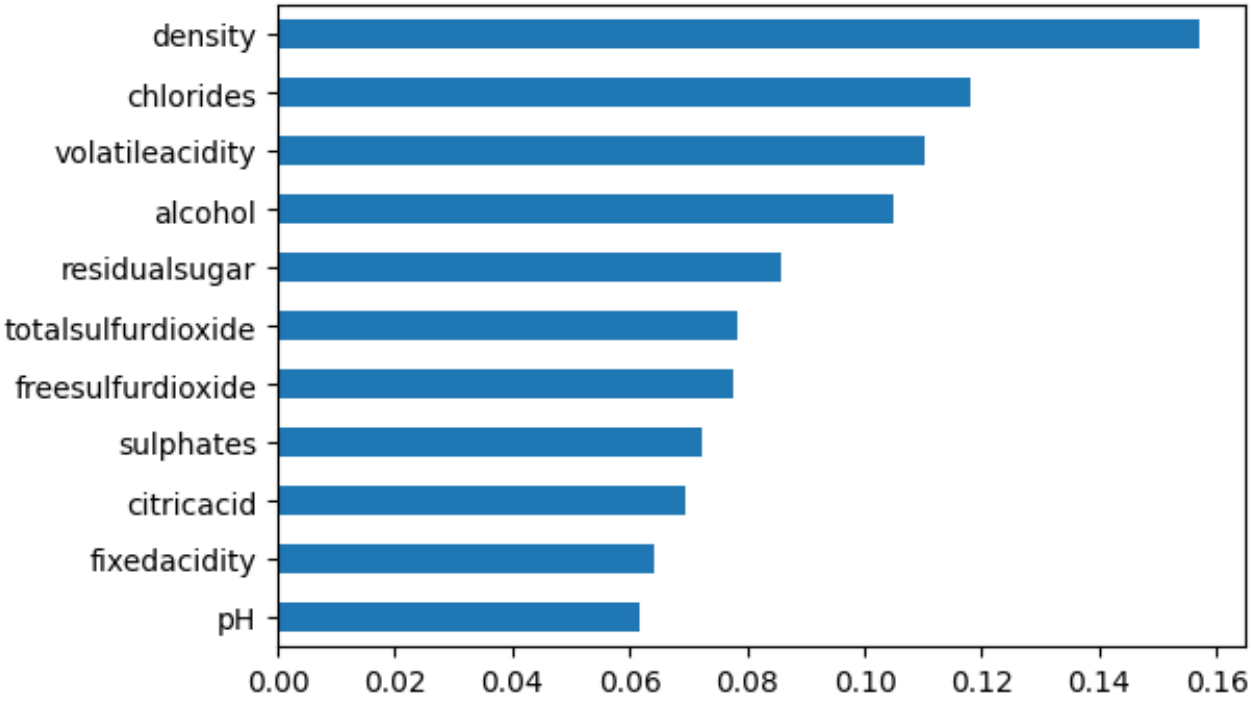
Algorytm **RandomizedSearchCV** został wywołany, aby losowo próbować różne kombinacje hiperparametrów. Przy użyciu 200 iteracji (n_iter=200), algorytm przeszukał przestrzeń hiperparametrów, oceniając skuteczność modelu dla każdej kombinacji. Po 40 minutowej kalkulacji wybrał najlepsze parametry dla zbudowania modelu XGBoost. Oto one:

- **objective = multi:softmax**
- **num_class = 7**
- **n_estimators = 500**
- **max_depth = 15**
- **grow_policy = depthwise**
- **subsample = 0.6**
- **gamma = 5**

- **colsample_bytree = 0.2**
- **learning_rate = 0.3**
- **reg_alpha = 0.4**
- **reg_lambda = 0.2**
- **seed = 308161**

Z takimi hiperparametrami zbudujemy nasz kolejny model.

4.3.1. Ważność predyktorów



O dziwo alkohol spadł w ważności predyktorów, z pierwszego miejsca znalazł się na czwartym. Najważniejszym czynnikiem w tym wypadku okazała się gęstość, poza tym dalej wszystkie predyktory odgrywają swoją rolę w działaniu modelu.

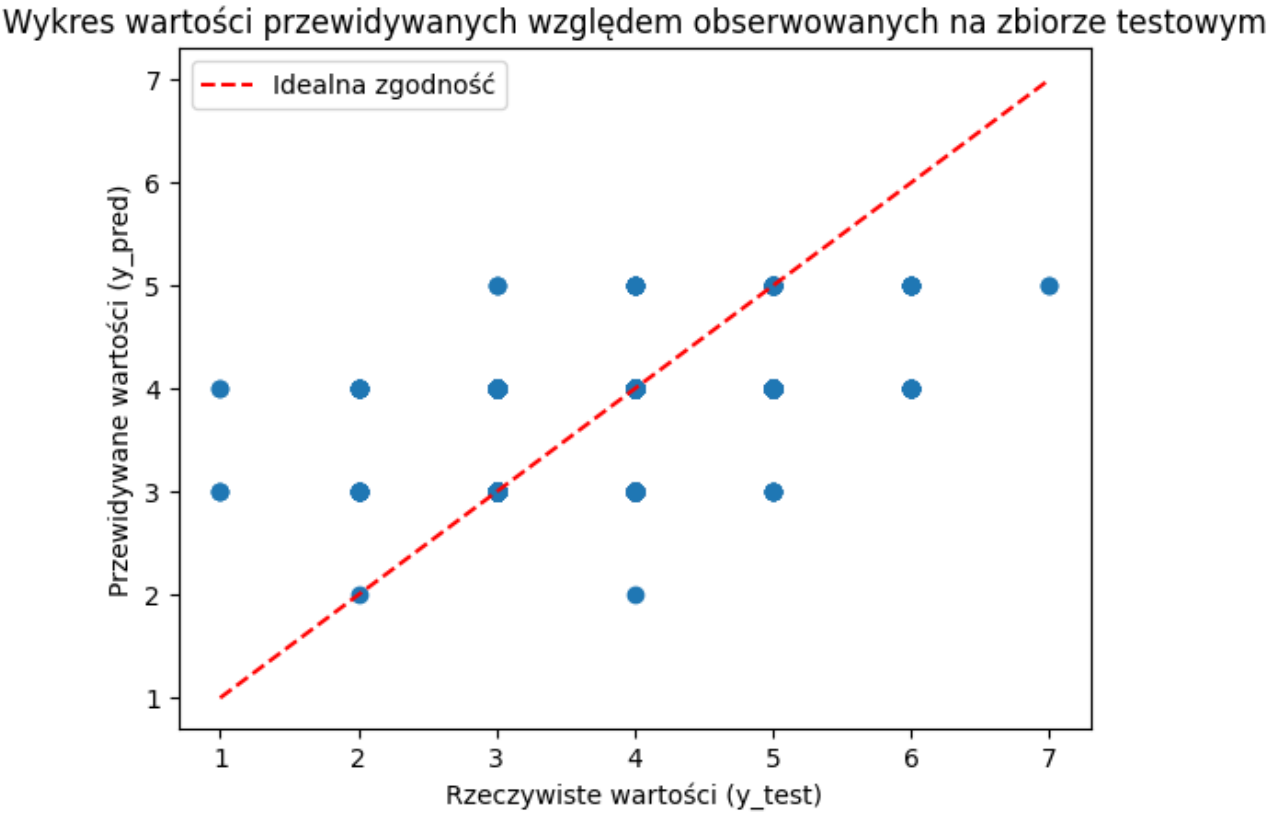
4.3.2. Ocena jakości modelu: miary jakości

Zbiór uczący	
Trafność	0.576
Trafność z odstępstwem o 1	0.943
Średni błąd bezwzględny (MAE)	0.486

Zbiór testowy	
Trafność	0.544
Trafność z odstępstwem o 1	0.944
Średni błąd bezwzględny (MAE)	0.514

Ten model nie wykazuję efektów przeuczenia. Osiąga on bardzo podobne wyniki do modlu szacowania, średnio klasyfikuje klasy z błędem o pół oceny. Również jego trafność wskazuje na prawidłowe klasyfikowanie połowy obserwacji.

4.3.3. Ocena jakości modelu: wykres wartości przewidywanych względem obserwowanych



Poprawiony model dalej nie odgaduje poprawnie win o skrajnych jakościach, również wina o jakościach 6 czy 7 klasyfikuje jako wina o jakościach 4 lub 5. Ten model nie jest idealny, możliwe że nadaje się do oceny jakości średnich win, jednak na pewno nie nadaje się do znajdowania win o wysokiej jakości. Taką ocene lepiej pozostawić ekspertom.

4.4. Wnioski

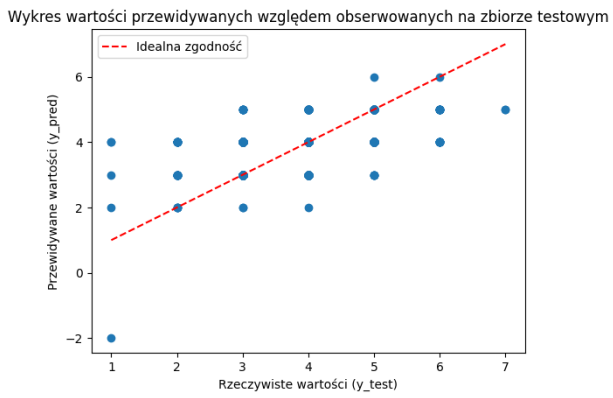
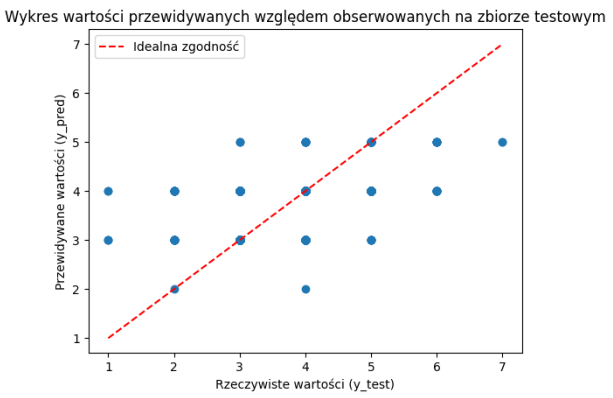
Pierwszy stworzony przez nas model był zdecydowanie przeuczony. Stworzenie modelu poprzez dobranie hiperparametrów za pomocą klasy RandomizedSearchCV pozwoliło na usunięcie efektu przeuczenia. Nie poprawiło nam jednak to znacznie możliwości przewidwania modelu klasyfikacji. Model jest przeciętnej jakości, aczkolwiek jest użyteczny. To on zostanie porównany z modelem szacowania zbudowanym za pomocą MLP.

5. Zestawienie metod predykcyjnych (MLP vs XGBoost)

5.1. Miary jakości

	Model 1: MLP		Model 2: XGBoost	
	Zbiór uczący	Zbiór testowy	Zbiór uczący	Zbiór testowy
Trafność	0.592	0.546	0.576	0.544
Trafność z odstępstwem o 1	0.967	0.949	0.943	0.944
Średni błąd bezwzględny (MAE)	0.444	0.508	0.486	0.514

5.2. Wykres wartości przewidywanych względem oczekiwanych

Model 1: MLP	Model 2: CART
	

5.3. Wnioski

Zestawienie tych dwóch modeli, które oceniają zmienną celu jako inny rodzaj, MLP jako zmienną ilościową oraz XGBoost jako zmienną jakościową pokazuje, że w przypadku tego zbioru danych oraz tego problemu osiągają prawie identyczne wyniki. Różnica jest marginalna, dla zbioru testowego różnią się o wartości tysięczne. Model regresyjny uzyskał delikatnie lepsze wyniki dla zbioru testowego, natomiast model klasyfikacyjny wydaje się mieć bardziej stabilne wyniki, różnice między wynikami dla zbiorów są mniejsze. Nie jest możliwe jednoznaczne stwierdzenie, który model jest lepszy.

6. Grupowanie

6.1. Wstęp

Do pogrupowania danych został wykorzystany algorytm DBSCAN (Density-based spatial clustering of application with noise), który grupuje obserwacje na podstawie ilości ich sąsiadów, natomiast oznacza jako obserwacje odstające (szum), te których najbliżsi sąsiedzi są zbyt daleko.

Do stworzenia grupowania za pomocą takiej klasy potrzebne są dwa parametry:

- **min_samples** – minimalna liczba obserwacji w sąsiedztwie danej obserwacji(włącznie z tą obserwacją).
- **eps** – promień sąsiedztwa, maksymalna odległość między obserwacjami pozwalająca uznać je za sąsiadów

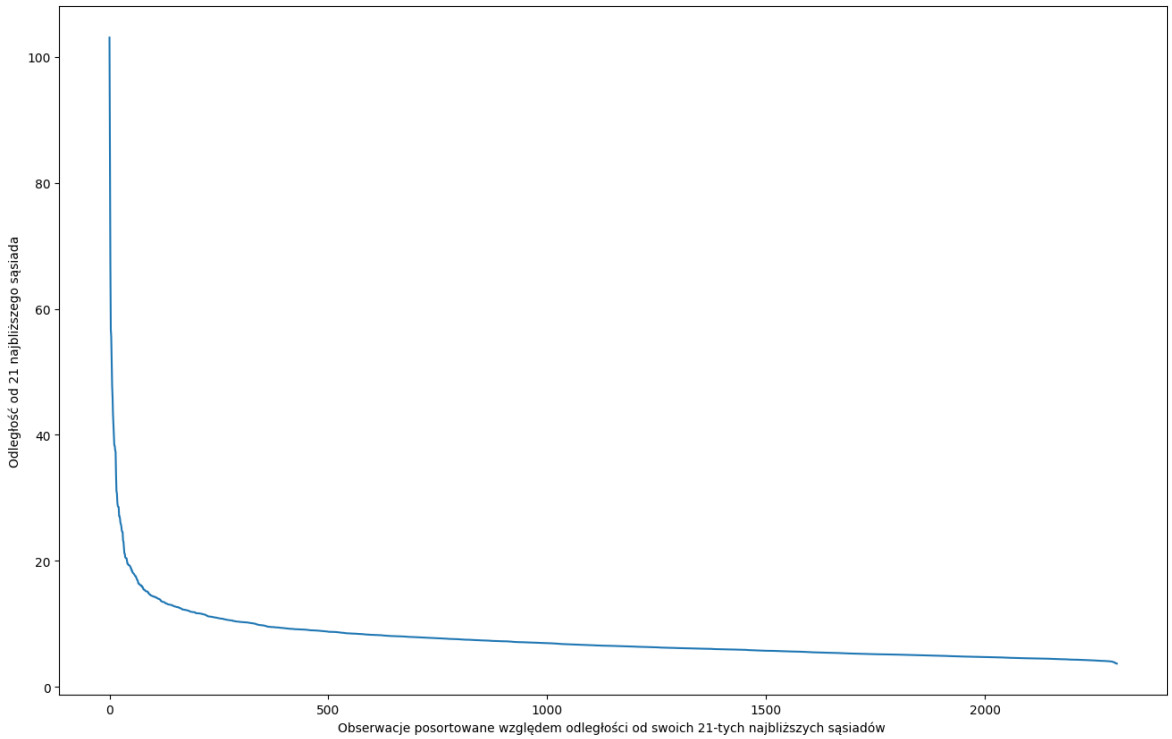
Wartość parametru **min_samples** najczęściej ustala się jako podwojony wymiar przestrzeni danych.W tym wypadku da nam to wartość 22, jako że ilość parametrów branych pod uwagę w grupowaniu wynosi 11.

Do ustalenia wartości parametru **eps** została zastosowana metoda zalecana przez twórców algorytmu DBSCAN. Wygląda ona następująco:

1. Dla każdej obserwacji wyznacz odległości od jej (k-1)-ych najbliższych sąsiadów, k = **min_samples**
2. Posortuj otrzymane odległości malejąco i narysuj wykres
3. Znajdź na wykresie punkt, w którym krzywa zmienia nachylenie (punkt łokcia)
4. Współrzędna y-owa tego punktu jest szukaną wartością parametru **eps**

6.2. Dobieranie wartości **eps**

Za pomocą klasy **NearestNeighbors** z pakietu **sklearn.neighbors** znaleźliśmy 21 najbliższych sąsiadów dla każdej obserwacji. Następnie posortowaliśmy je malejąco i narysowaliśmy następujący wykres:

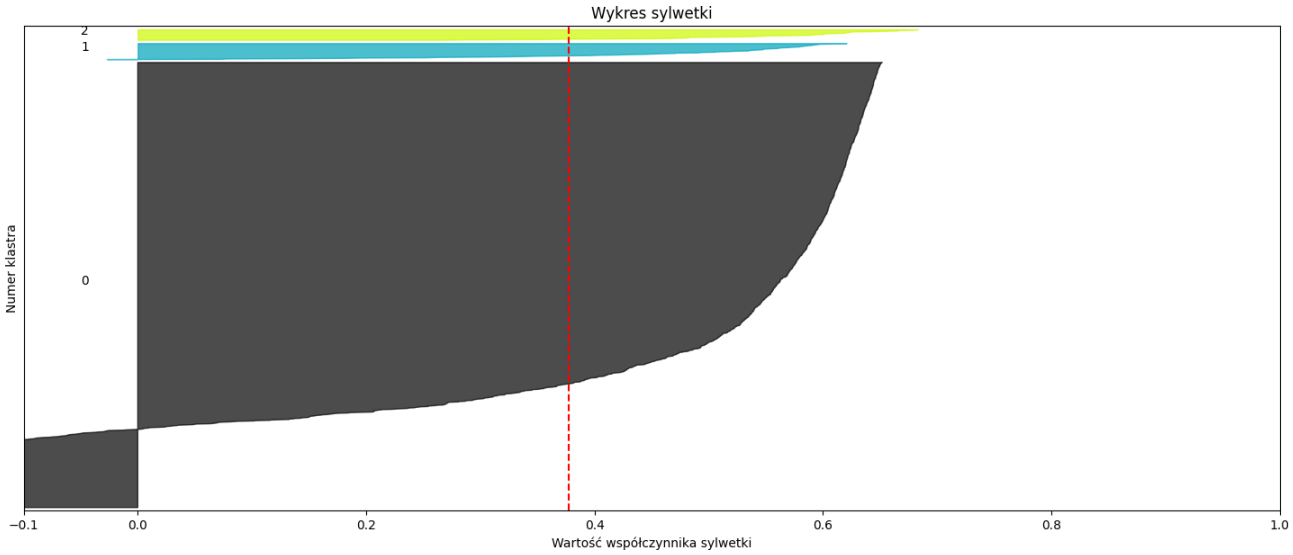


Skupiliśmy się na znalezieniu tzw. punktu łokcia, to znaczy miejsca, w którym krzywa zmienia nachylenie. Możnaby powiedzieć, że jest to proste zadanie i stwierdzić, że taka wartość to liczba z przedziału [10 ; 20], ale dla algorytmu dobierania wartości parametru **eps** nie był to dobry zakres. Stworzono więc tablicę z wartościami dla przykładowych obserwacji. Dla obserwacji o wartościach ze zbioru: [5, 10, 25, 35, 75, 150, 250, 500, 1000, 1300, 2000, 2300], uzyskano następujące im wartości parametru **eps:** [51.3, 40.4, 26.1, 21.1, 15.9, 12.8, 10.9, 8.8, 7.0, 6.2, 6.1, 4.8, 3.7]. Jak się okazało dla większości tych wartości algorytm grupował wina na dwie grupy: grupę 0 oraz grupę -1 czyli obserwacje odstające. Były to grupowania, które absolutnie nie spełniały swojego zadania. Wartości **eps,** dla których algorytm zwrócił sensowne ilości grup (4, 6 – z grupą obserwacji odstających) to: 6.2 oraz 4.8. Dlatego zostały stworzone grupowania przy takich parametrach.

Dla wartości **eps = 4.8** grupowanie nie spełniało norm do zaakceptowania jego jakości więc zostało odrzucone. Opisane zostanie grupowanie dla **eps = 6.2**

6.3. Miara sylwetki

Średnia miara sylwetki dla grupowania na 3 grupy (4 z obserwacjami odstającymi) została policzona jako 0.377. Uznaje się takie grupowanie za poprawne, nie koniecznie dobre. Do zobrazowania tych wyliczeń został stworzony następujący wykres:



Jak widać grupowanie nie jest równoliczne, można nawet stwierdzić o bardzo dużej przewadze grupy 0. Poza tym widać sporo obserwacji, które absolutnie nie powinny się znaleźć w grupach 0 czy 1. Bardzo wyraźnie obniżyło to średnią miarę sylwetki dla tego grupowania. Obserwacje, które zostały oznaczone jako odstające stworzyły grupę o ilości aż 764 obserwacji. Spowodowało to wyrzucenie z grupowania 33% win! Jest to bardzo duża liczba. Nie jest to dobre grupowanie.

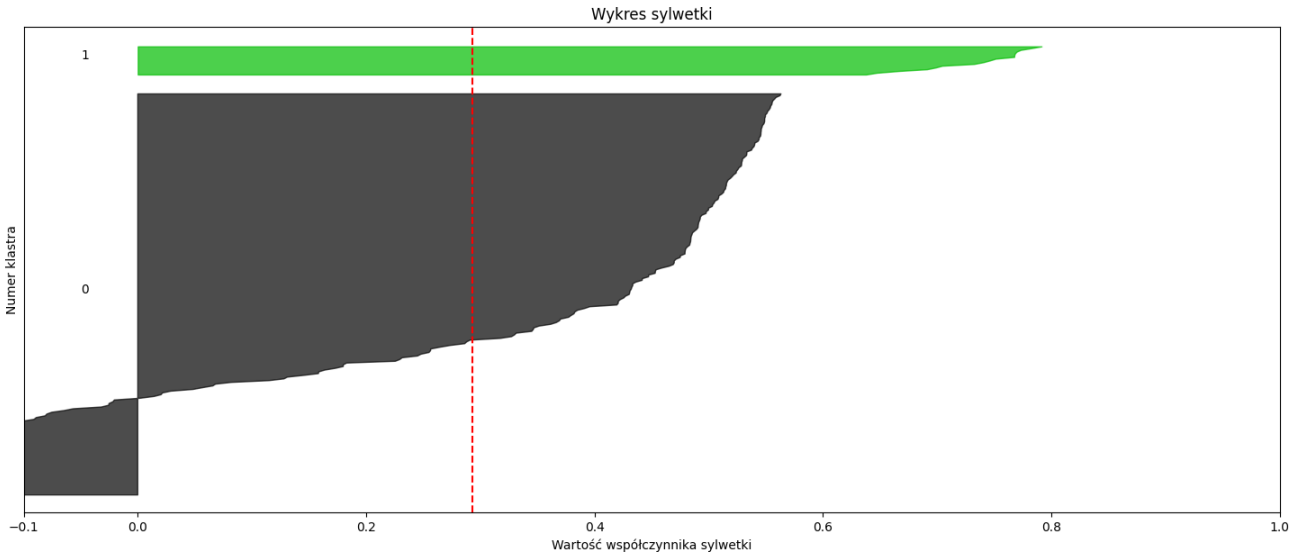
6.4. Charakterystyka grup

Podsumowując opisy konkretnych cech każdej z grup, można powiedzieć że grupa 0 to wina mniej słodkie, delikatnie mocniejsze oraz o krótszym terminie spożycia(wskazuje na to niższe stężenie dwutlenku siarki, odpowiedzialnego za hamowanie rozwoju drobnoustrojów oraz przeciwdziałanie utlenianiu), natomiast grupy 1 i 2 to wina słodsze, delikatniejsze w procentach, trwalsze oraz lekko kwaśniejsze. Trzeba oczywiście przypomnieć o gigantycznej przewadze w ilości win w grupie 0, oraz o fakcie zaklasyfikowania aż 764 win jako obserwacje odstające. Moim zdaniem obserwacje odstające mogłyby zostać połączone z grupami 1 i 2, tworząc w ten sposób podział na dwie grupy z możliwe, że mniejszymi ale widocznymi różnicami.

6.5. Zastosowanie grupowania dla zbioru testowego

W przypadku zbioru testowego, grupowanie algorytmem DBSCAN z ustalonymi wcześniej parametrami nie sprawdziło się. Zostały stworzone tylko dwie grupy, oprócz obserwacji odstających. Z czego grupa 0 miała 229 obserwacji a grupa 1 tylko 17. Obserwacje odstające natomiast liczyły aż 741 rekordów. Średnia miara sylwetki dla tych dwóch grup wynosiła co prawda 0.292, ale przez

odrzucenie aż ¾ całego zbioru nie można uznać tego grupowania za poprawne. Tak prezentuje się wykres miary sylwetki dla zbioru testowego:



6.6. Związek grupowania ze zmienną celu

Korelacja między grupowaniem a oceną jakości wina wynosi 0.13, co pokazuje o bardzo małym poziomie zależności tych dwóch charakterystyk. Tak mały związek nie daje nam podstaw do uznania zależności.