

Predykcja poziomu otyłości - porównanie modeli probabilistycznych

AA, Wiktor Sadowy, Kamil Matejuk

June 27, 2024

Abstract

W tej pracy została dokonana analiza zbioru zawierającego dane o poziomie otyłości i nawykach żywieniowych oraz kondycji fizycznej badanych. Następnie dokonano porównania trzech algorytmów probabilistycznych w zadaniu klasyfikacji poziomu otyłości - Gaussian Processes (GP), Bayesowska Sieć Neuronowa (BNN) oraz Naive Bayes.

1 Eksploracyjna analiza danych

1.1 Opis zmiennych

W zbiorze danych znajduje się 17 zmiennych objaśniających (w tym 7 numerycznych oraz 10 kategorycznych) i 1 zmienna objaśniana. Poniższa tabela prezentuje charakterystykę tych zmiennych:

Zmienna	Typ	Ilość Kategorii	Jednostka	Zakres	Brakujące wartości
Gender (Gender)	Cat	2	-	-	Nie
Age (Age)	Num	-	Lata	0-99	Nie
Height (Height)	Num	-	Metr	0.5-2.5	Nie
Weight (Weight)	Num	-	Kg	3-200	Nie
Family History with Overweight (fam_hist)	Cat	2	-	-	Nie
Frequent Consumption of High Caloric Food (FAVC)	Cat	2	-	-	Nie
Frequency of Consumption of Vegetables (FCVC)	Num	-	Liczba	1-3	Nie
Number of Main Meals (NCP)	Num	-	Posiłki	1-10	Nie
Consumption of Food Between Meals (CAEC)	Cat	2	-	-	Nie
Consumption of Water Daily (CH2O)	Num	-	Litr	0-10	Nie
Calories Consumption Monitoring (SCC)	Cat	2	-	-	Nie
Physical Activity Frequency (FAF)	Num	-	-	-	Nie
Time Using Technology Devices (TUE)	Num	-	Godz.	0-24	Nie
Consumption of Alcohol (CALC)	Cat	2	-	-	Nie
Smoking Habit (SMOKE)	Cat	2	-	-	Nie
Transport Used (MTRANS)	Cat	5	-	-	Nie
Obesity Level	Cat	6	-	-	Nie

Table 1: Opis zmiennych z datasetu

W późniejszej analizie nie będziemy brali pod uwagę zmiennych Height i Weight, gdyż zależy nam na sprawdzeniu czy na podstawie nawyków jesteśmy w stanie wnioskować o poziomie otyłości.

Rozkład poziomów otyłości wśród próbek jest zbalansowany, we wszystkich klasach mamy do czynienia z od 13 do 16% udziałem w całkowitej populacji, dokładne dane znajdują się na rysunku poniżej.

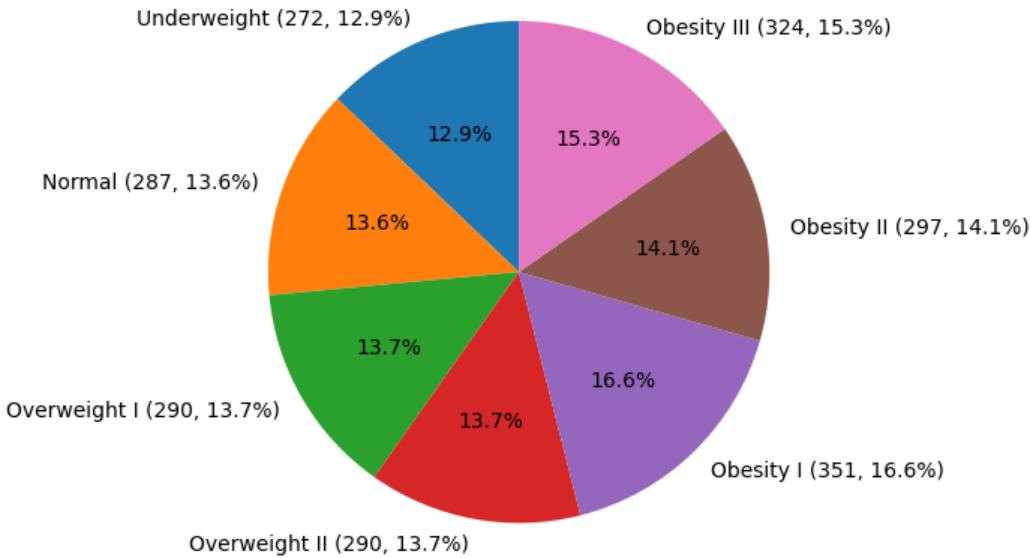


Figure 1: Rozkład klas zmiennej objaśnianej

Zastosowano dwie metody redukcji wymiarowości - PCA oraz T-SNE aby uzyskać wgląd w strukturę oraz wzorce w danych.

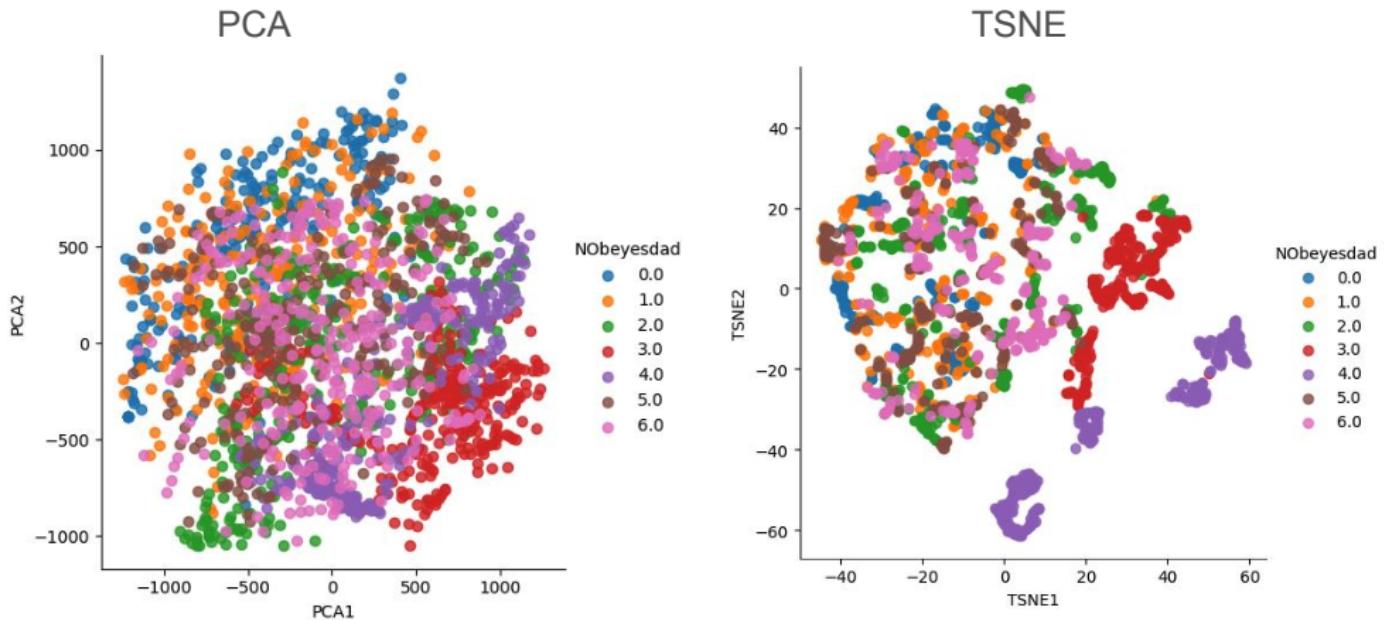


Figure 2: Dwuwymiarowe reprezentacje danych po przetworzeniu przez PCA i T-SNE

Algorytm T-SNE odkrył klastry w poszczególnych klasach. Na pierwszy rzut oka widać, że klasa 4 (nadwaga II stopnia, fioletowa) jest liniowo separowalna od reszty. Nie można tego jednak powiedzieć o większości innych klas. PCA słabiej oddziela zaobserwowane dane.

Przeprowadzono również analizę wyjaśnialności zmienności danych poprzez główne składowe. Okazuje się, że PCA potrzebuje 8 głównych składowych, żeby wyjaśnić 99% zmienności w danych.

Oznacza to, że redukcja wymiarowości danych do mniej niż 8 wymiarów będzie się wiązała za znaczną utratą informacji.

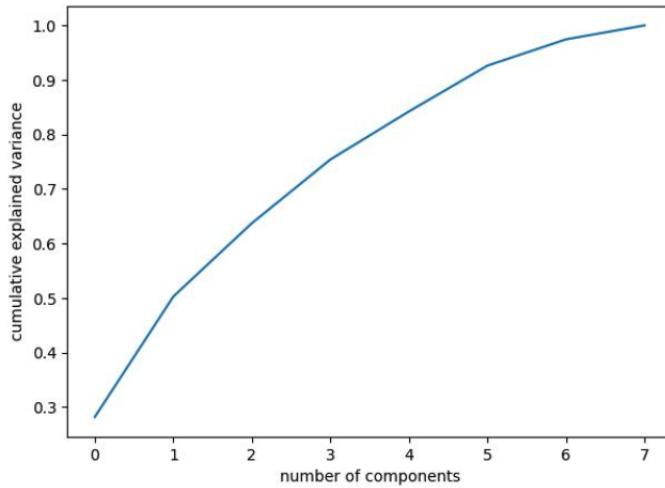


Figure 3: PCA potrzebuje 8 głównych składowych by wyjaśnić 99% zmienności w danych

1.2 Obserwacje odstające

Żeby zobaczyć jakie obserwacje są odstające obliczyliśmy z-score dla każdej zmiennej numerycznej. Jeżeli wartość bezwzględna z-score jest wyższa od 2 to jest szansa, że dany odczyt jest outlierem. Poniżej przedstawiamy liczbę outlierów dla każdej zmiennej numerycznej

Zmienna	Liczba wartości odstających
Age	144
Frequency of Consumption of Vegetables (FCVC)	82
Number of Main Meals (NCP)	243
Consumption of Water Daily (CH2O)	0
Physical Activity Frequency (FAF)	99
Time Using Technology Devices (TUE)	144

Table 2: Wartości odstające

Po bliższej analizie tych wartości wyszło, że to są naturalnie występujące outliersy tzn. w datasetie mamy znacznie starsze osoby niż średnia wieku. Analogicznie sytuacja wygląda w przypadku pozostałych zmiennych.

Po przeanalizowaniu sytuacji ustaliliśmy, że nie ma sensu usuwać outlierów w naszych danych gdyż wówczas znacząco zmienimy informacje wynikające z naszego datasetu.

1.3 Zależności zmiennych parami

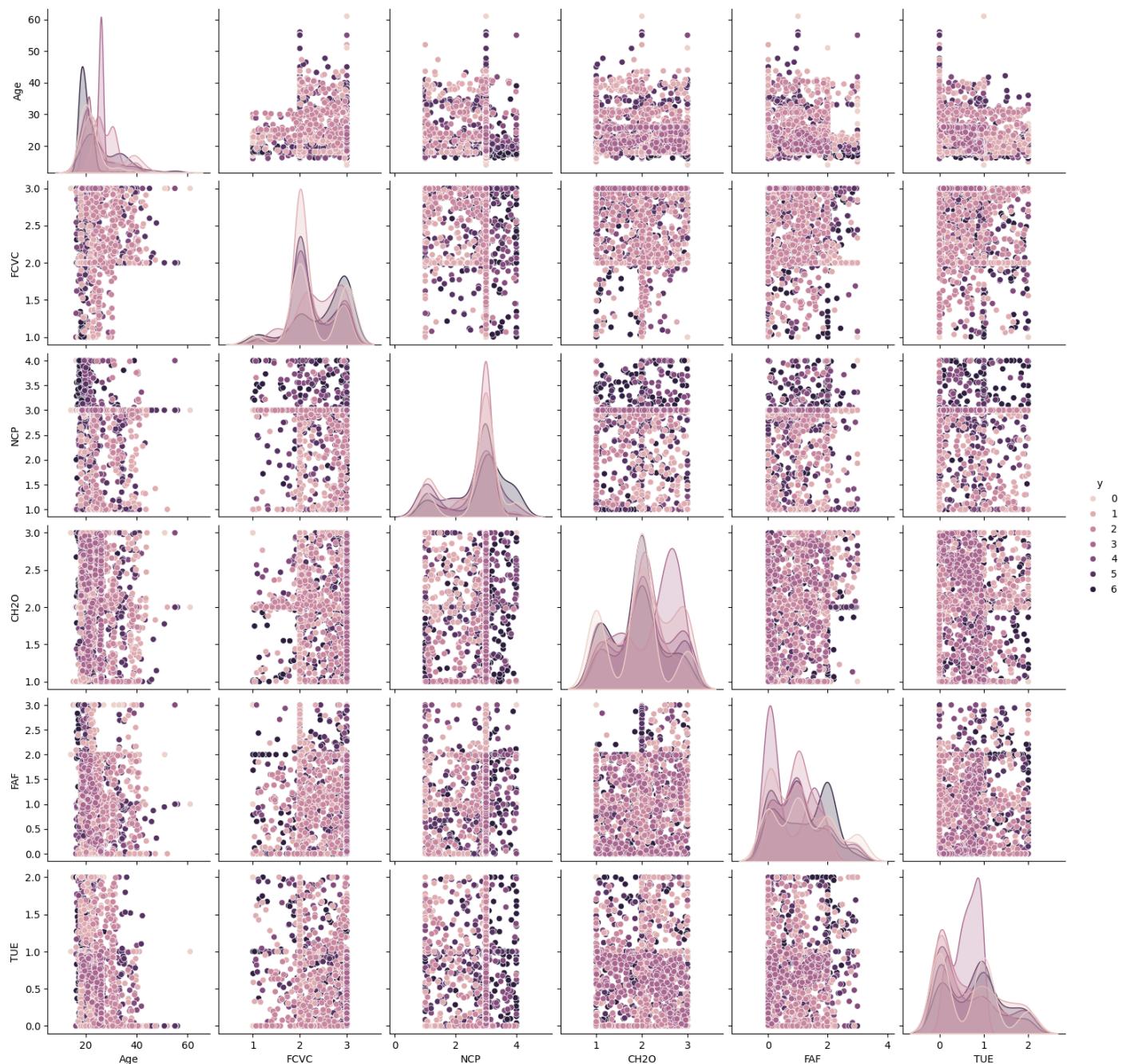


Figure 4: Wykresy zależności zmiennych parami

Patrząc na wykres zmiennych numerycznych zestawionych ze sobą parami (rysunek 4), widzimy, że z reguły krańcowe kategorie zmiennej wyjaśnianej (otyłość 3 stopnia, czarna) są separowalne liniowo względem dwóch zmiennych. Widzimy również po histogramach (na przekątnej), że rozkłady zmiennych można reprezentować jako skrzywiony rozkład normalny (ang. skewed normal distribution). Ponadto nie zaobserwowano innych zależności, na co również wskazuje macierz korelacji.

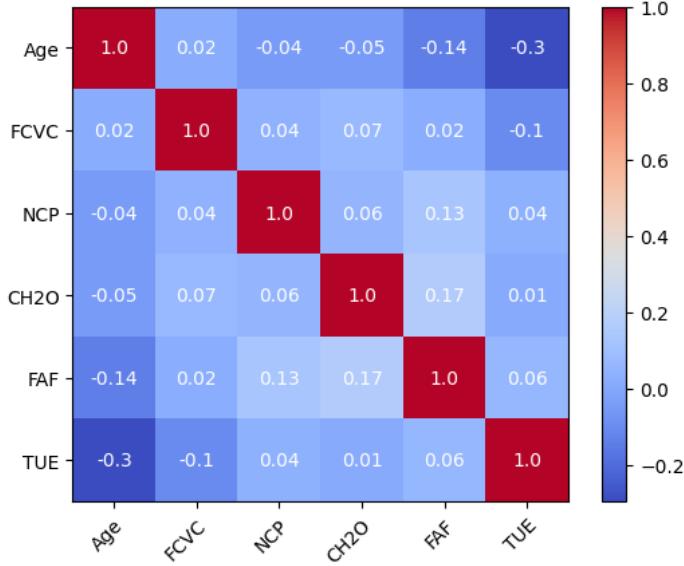


Figure 5: Macierz korelacji między zmiennymi numerycznymi

Z macierzy korelacji zaprezentowanej na rysunku 5. wynika, że nie ma dużych korelacji między zmiennymi numerycznymi. Największa jest negatywna korelacja pomiędzy zmiennymi TUE a Age, która wynosi -0.3 .

1.4 Podział danych

Dane zostały podzielone na trzy zbiory: treningowy, walidacyjny i testowy, z wykorzystaniem stratyfikacji.

Podzbiór	Część całości	Ilość próbek
Treningowy	70%	1477
Walidacyjny	15%	317
Testowy	15%	317

Table 3: Podział zbioru danych

1.5 Preprocessing

Ponieważ wartość otyłości jest wyliczana na podstawie wskaźnika BMI, który z kolei jest bezpośrednio wyliczany ze wzrostu i wagi, kolumny *Height*, *Weight* i *BodyMassIndex(BMI)* zostały usunięte już na początku.

W ramach preprocessingu danych umożliwiono wykonanie dwóch operacji: konwersja danych kategorycznych do numerycznych, oraz dyskretyzacja danych numerycznych do zadanych zakresów. Każdą transformację wykonano na zbiorze treningowym, a następnie nałożono na zbiory walidacyjny i testowy, aby uniknąć przecieku danych.

Dane kategoryczne (niech liczba kategorii w kolumnie będzie oznaczona liczbą N) mogły zostać zakodowane na dwa sposoby:

- *One-Hot* - gdzie dla każdej kolumny powstawało N kolumn z wartościami $\{0, 1\}$ reprezentującymi przynależność do kategorii
- *Ordinal* - gdzie każdej kategorii przypisana została liczba całkowita z zakresu $\{0, \dots, N - 1\}$

Po konwersji kolumn kategorycznych, dodatkowo istnieje opcja dyskretyzacji danych do zadanych zakresów, np kolumnę z zakresem $0-100$ można by przekształcić na pięć komórek: $[0-20)$, $[20-40)$, $[40-60)$, $[60-80)$, $[80-100]$.

2 Modele

2.1 Gaussian Processes (GP)

2.1.1 Wyjaśnienie doboru modelu

Wybraliśmy ten model do tego zadania, ponieważ GP są modelami nieparametrycznymi, czyli nie zakładają konkretnej formy funkcji modelującej relację w danych. Dzięki temu model może zaadaptować się do skomplikowanej struktury danych, która mogłaby nie być dobrze odwzorowana przez modele parametryczne. Poza tym modele GP mogą korzystać z wcześniejszej wiedzy (prior), co pozwala na polepszenie predykcji, gdy danych jest mało.

2.1.2 Zasada działania

Proces Gaussa to zbiór nieskończoność wielu zmiennych losowych, z których każda reprezentuje wartość funkcji w określonym punkcie przestrzeni cech. Każdy punkt danych reprezentowany jest jako próbka z pewnego rozkładu prawdopodobieństwa. Funkcje w ramach procesu Gaussa są ze sobą powiązane poprzez kowariancję, która określa podobieństwo między różnymi punktami przestrzeni cech.

2.1.3 Uczenie

Model jest oparty na estymacji średnich wartości funkcji oraz ich kowariancji na podstawie dostępnych danych. Potrzebujemy najpierw priora rozkładu funkcji, który w GP jest modelowany jako funkcja m zwracająca średnią dla danego punktu danych oraz funkcja zwracająca macierz kowariancji k . Nosi ona nazwę kernela. Kernel można dobierać w zależności od zadania, ale ja używam kernela RBF (radial basis). Następnie obliczamy rozkład posterior funkcji, który jest rozkładem normalnym wielu zmiennych, ale warunkowym na danych zaobserwowanych. Po obliczeniu posteriora następuje aktualizacja funkcji m zwracającej średnie oraz funkcji k zwracającej macierze kowariancji. Parametry funkcji k specyficzne dla danego kernela i są optymalizowane w trakcie uczenia. Np. w kernelu RBF mamy parametr scale oraz noise. W naszym modelu te dwa parametry na początku będziemy ustawać jako konkretne wartości priora i traktować je jako hiperparametry.

2.1.4 Predykcja

Dla nowego wektora X obliczamy jego predykcję wywołując funkcję m , co daje estymowaną wartość y .

2.1.5 Wady

GP wspiera natywnie tylko klasyfikację binarną, zatem musialem zaimplementować klasyfikację wieloklasową używając techniki OVR (one vs rest).

2.2 Bayesowska Sieć Neuronowa (BNN)

2.2.1 Wyjaśnienie doboru modelu

Wybraliśmy Bayesowską sieć neuronową ze względu na jej wyjaśnialność i większe możliwości dopasowania się do problemu niż inne metody oparte na Bayesian probability. W przypadku dobrze wytrenowanego modelu jesteśmy w stanie uzyskać dużo informacji o naszym zbiorze danych.

2.2.2 Zasada działania

Bayesowska sieć neuronowa działa zbliżenie do innych sieci neuronowych z tą różnicą, że w przypadku standardowych sieci neuronowych bezpośrednio ustalamy wagi i bias modelu. W przypadku Bayesowskich sieci neuronowych naszym zadaniem będzie aproksymacja parametrów rozkładów opisujących wagi i bias modelu. Różnicę tę pokazuje poniższe zdjęcie.

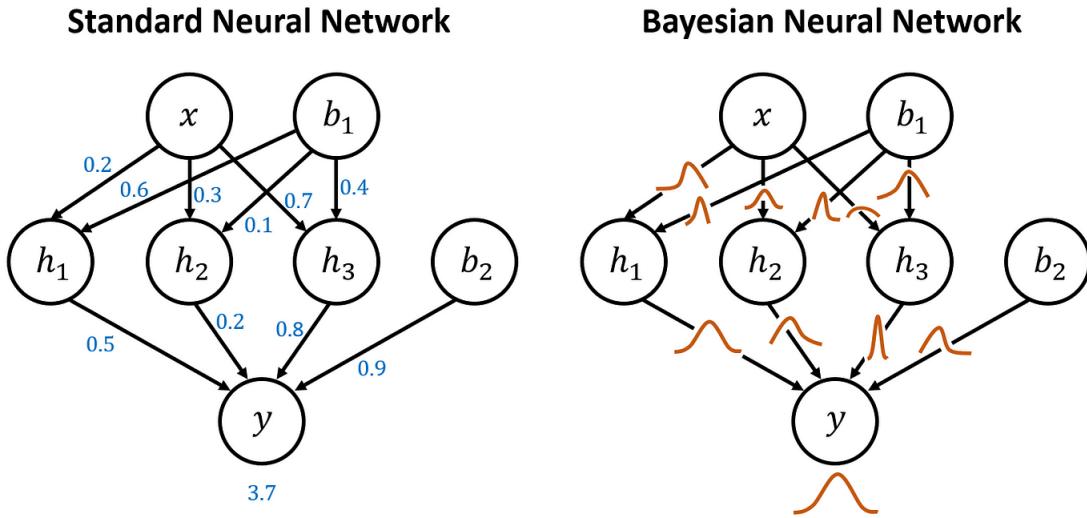


Figure 6: Różnica pomiędzy standardową siecią neuronową, a Bayesowską siecią neuronową

Żeby móc aproksymować parametry rozkładu musimy wpierw ustalić typ naszego rozkładu z którego będziemy próbować wagi i bias modelu. Nie ma łatwej metody ustalenia optymalnego typu rozkładu. Trzeba więc próbować różne typy rozkładów aż nie uzyskamy optymalnych wyników.

Mając ustalony typ rozkładu naszym zadaniem będzie ustalenie takich parametrów tego rozkładu, który pozwoli najlepiej zamodelować zadanie.

2.2.3 Uczenie

Uczenie modelu przebiega w następujący sposób:

1. Robimy forward pass przez nasz model uzyskując predykcje
2. Wyliczamy funkcję kosztu
3. Na postawie funkcji kosztu aktualizujemy parametry naszego modelu (czyli parametry rozkładu opisującego weights i parametry rozkładu opisującego bias) robiąc tzw. backpropagation. Wpierw liczymy gradienty w następujący sposób:

$$\Delta_\mu = \frac{\partial f}{\partial \theta} + \frac{\partial f}{\partial \mu}$$

$$\Delta_\sigma = \frac{\partial f}{\partial \theta} \frac{\epsilon}{\sigma} + \frac{\partial f}{\partial \sigma}$$

A potem aktualizujemy parametry modelu:

$$\mu^{(t+1)} = \mu^t - \alpha \Delta_\mu$$

$$\sigma^{(t+1)} = \sigma^t - \alpha \Delta_\sigma$$

4. Powyższe trzy kroki powtarzamy przez określoną liczbę epok albo dopóki nie został spełniony inny warunek np. koszt walidacyjny nie malał przez 7 epok

2.2.4 Predykcja

Mając parametry rozkładu czyli $\theta = (\mu, \sigma^2)$ jesteśmy w stanie obliczyć wartość weights i bias samplując z odpowiedniego rozkładu.

Powysze podejście ma jeden problem. Otóż nie przewiduje ono sytuacji gdzie wartość σ^2 jest ujemna. W takim przypadku nie uzyskamy predykcji w standardowy sposób. Źeby rozwiązać problem możemy użyć sztuczki z reparametryzacją, która pozwala nam uzyskać wagi i bias modelu w następujący sposób

$$f(\epsilon) = \theta = \mu + \sigma \cdot \epsilon$$

gdzie ϵ jest losowany z rozkładu normalnego standardowego:

$$\epsilon \sim \mathcal{N}(0, 1)$$

Mając ustalone wagi i bias liczymy wyjście naszego modelu w następujący sposób:

$$X * \text{wagi} + \text{bias}$$

W przypadku gdy mamy więcej niż jedną warstwę w sieci neuronowej to wynik powyższego kroku jest przekazywany do wejścia kolejnej warstwy aż nie dojdziemy do ostatniej warstwy.

2.2.5 Wady

Bayesowskie sieci neuronowe mają następujące wady:

1. Czas trenowania modelu jest długi. W przypadku dużych zbiorów danych trudno jest wytrenować optymalną Bayesowską sieć neuronową
2. Narzędzia do Bayesowskich sieci neuronowych nie są aż tak proste w użyciu jak inne prostsze modele oparte na Bayesian probability
3. Musimy zrobić założenie dotyczące tego jak wygląda prior w naszym modelu. Nie jest to zawsze łatwe zadanie, a wybór złego priora może znacząco pogorszyć wyniki modelu.

2.3 Naive Bayes (NB)

2.3.1 Wyjaśnienie doboru modelu

Model został wybrany, ponieważ ze względu na swoją prostotę jest szybki, a cechy wejściowe w większości wykazują niską korelację między sobą - niezależność cech jest jednym z założeń działania tego modelu. Ponadto model naiwnego bayesa powinien dobrze sobie poradzić z wysoką wymiarowością i małą ilością danych treningowych.

2.3.2 Zasada działania

Naive Bayes to algorytm klasyfikacyjny bazujący na twierdzeniu Bayesa. Zakłada on, że wszyskie obserwowane zmienne X_1, X_2, \dots, X_N są warunkowo niezależne względem zmiennej Y , oraz że jedyna zależność istnieje między zmiennej Y a zmiennymi X .

Dla konkretnej instancji $X^{(i)} = (x_1, x_2, \dots, x_N)$ klasyfikację uzyskano maksymalizując prawdopodobieństwo warunkowe klasy Y_k pod warunkiem danych $X^{(i)}$.

$$\hat{y} = \operatorname{argmax}_{k \in \{1, 2, \dots, K\}} P(y_k | x_1, x_2, \dots, x_N)$$

Stosując regułę Bayesa i *naiwne* podejście, że $P(x_i | x_{i+1}, x_{i+2}, \dots, x_N, y_k) = P(x_i | y_k)$:

$$P(y_k | x_1, x_2, \dots, x_N) = \frac{P(y_k) P(x_1, x_2, \dots, x_N | y_k)}{P(x_1, x_2, \dots, x_N)}$$

$$\begin{aligned}
P(y_k|x_1, x_2, \dots, x_N) &\propto P(y_k)P(x_1, x_2, \dots, x_N|y_k) \\
P(y_k|x_1, x_2, \dots, x_N) &\propto P(y_k, x_1, x_2, \dots, x_N) \\
P(y_k|x_1, x_2, \dots, x_N) &\propto P(y_k) \prod_{i=1}^N P(x_i|y_k) \\
\hat{y} = \operatorname{argmax}_{k \in \{1, 2, \dots, K\}} P(y_k) \prod_{i=1}^N P(x_i|y_k)
\end{aligned}$$

2.3.3 Uczenie

Uczenie polega na obliczeniu a priori prawdopodobieństw każdej klasy oraz parametrów rozkładów normalnych dla każdej cechy przy danej klasie. Obliczenie a priori prawdopodobieństwo każdej klasy $P(y_i)$ jako częstość występowania tej klasy w zbiorze treningowym $P(y_i) = N_{y_i}/N$. Następnie dla każdej cechy x_j i klasy y_i obliczamy średnią $\mu_{y_i,j}$ i wariancję $\sigma_{y_i,j}^2$. Te parametry są następnie wykorzystywane podczas predykcji, aby obliczyć prawdopodobieństwa $P(x_j|y_i)$ za pomocą funkcji gęstości rozkładu normalnego i łączne prawdopodobieństwa klas $P(y_i|X^{(j)})$ przy danych cechach $X^{(j)}$.

2.3.4 Predykcja

Dla nowego wektora cech $X^{(j)} = (x_1, x_2, \dots, x_N)$, obliczane są prawdopodobieństwa przynależności do każdej klasy y_i . Aby przypisać nowy punkt danych $X^{(j)}$ do jednej z klas, wybieramy klasę y o najwyższym prawdopodobieństwie.

Obliczane jest a priori prawdopodobieństwa każdej klasy $P(y_i)$. Następnie obliczane jest prawdopodobieństwa każdej cechy x_j przy danej klasie $P(x_j|y_i)$. Finalnie wyliczono łączne prawdopodobieństwo $P(y_i|X)$ dla każdej klasy y_i . Finalnie wybrana jest klasa y_i o najwyższym łącznym prawdopodobieństwie jako przewidywaną klasę \hat{y} .

2.3.5 Wady

Naive Bayes zakłada warunkową niezależność cech, co rzadko jest spełnione w praktyce, przez co model może być mniej dokładny, jeśli cechy są skorelowane. Ponadto, model może mieć trudności z obsługą bardzo złożonych danych, gdzie struktura zależności między cechami jest bardziej skomplikowana. W przypadku małych próbek danych, estymowane prawdopodobieństwa mogą być niestabilne, co może prowadzić do błędnych predykcji.

3 Eksperymenty

3.1 Metryki do porównywania różnych typów modeli

Do porównywania wyników modeli użyliśmy następujących metryk. Niech:

TP – przypadki prawdziwie pozytywne

FP – przypadki faszywne pozytywne

TN – przypadki prawdziwie negatywne

FN – przypadki faszywne negatywne

Metryka	Definicja
Dokładność (Accuracy)	$\frac{TP+TN}{TP+TN+FP+FN}$
Precyzja (Precision)	$\frac{TP}{TP+FP}$
Czułość (Recall)	$\frac{T}{TP+FN}$
F1	$2 \times \frac{Precision \times Recall}{Precision + Recall}$
AUC	Obszar pod krzywą ROC pokazujący zdolność modelu do rozróżniania między klasami pozytywnymi i negatywnymi. Dla problemu klasyfikacji wieloklasowej wykorzystano podejście <i>One-vs-Rest</i> uśredniające wyniki pomiędzy klasami.

3.2 Gaussian Processes (GP)

Aby uruchomić eksperymenty dla modelu GP należy wykonać instrukcje w pliku README aby pobrać zbiór danych, a następnie wykonać polecenie ‘dvc repro’, które wykona notebook z implementacją modelu, eksperymentami oraz tuningiem hiperparametrów z narzędziem Optuna i zapisze wyniki do pliku html w folderze results/

3.2.1 Ewaluacja modelu

Do ewaluacji modelu Gaussian Process wykonano następujący scenariusz eksperimentalny: Podzielono zbiór danych na zbiór uczący o wielkości 70% całego zbioru, walidacyjny o wielkości 15% całego zbioru oraz testowy o wielkości 15% całego zbioru.

Na początku wykonano prostą ewaluację modelu z domyślnymi parametrami. Poniżej znajdują się wyniki modelu z domyślnymi parametrami ($scale=1.0$, $noise=1e-6$), trenowanego na zbiorze treningowym i ewaluowanego na zbiorze walidacyjnym, powtórzzone 20 razy i uśrednione.

Metryka	Wartość
Accuracy	0.823
F1 Score	0.821
Precision	0.825
Recall	0.823
AUC	0.959

Table 4: Ewaluacja GP z domyślnymi parametrami na zbiorze val

Następnie, dla kombinacji hiperparametrów modelu, dokonano 10-krotnej ewaluacji w postaci trenowania modelu na zbiorze treningowym i ewaluacji na zbiorze walidacyjnym. Następnie zaraportowano metryki klasyfikacji (F1-score, precision, recall, accuracy, RoC-AUC) w formie uśrednionej oraz z klamrami przedstawiającymi odchylenie standardowe dla metryk. Poniżej metryki w relacji do wartości zmieniającego się parametru scale:

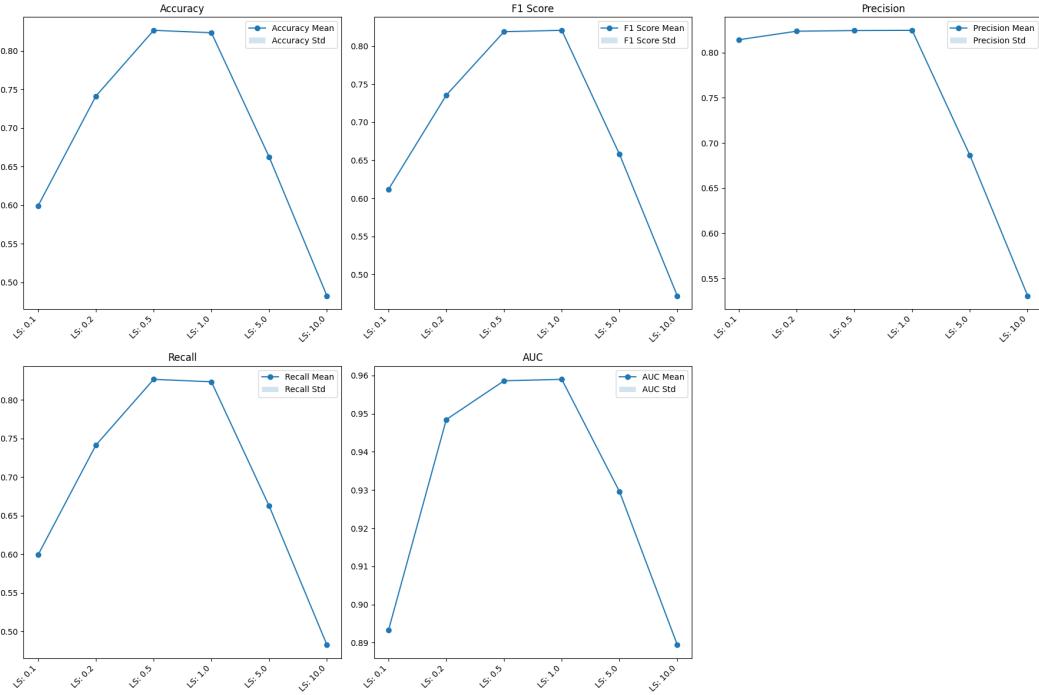


Figure 7: Metryki dokładności modelu w zależności od hiperparametru scale

Poniżej metryki w relacji do wartości zmieniającego się parametru noise:

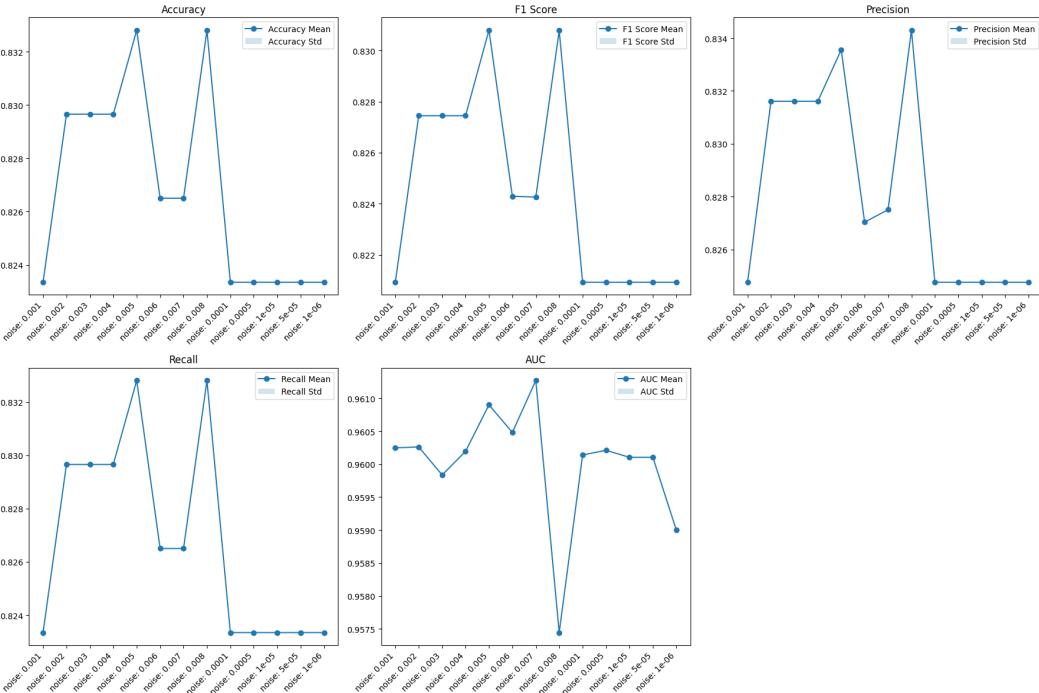


Figure 8: Metryki dokładności modelu w zależności od hiperparametru noise

3.2.2 Przeszukiwanie hiperparametrów (Optuna)

Do przeszukiwania hiperparametrów w modelu Gaussian Process użyty został pakiet Optuna wykorzystujący optymalizację Bayesowską. Wizualizacje procesu optymalizacji hiperparametrów znajdują się na rysunkach 9, 10, 11.

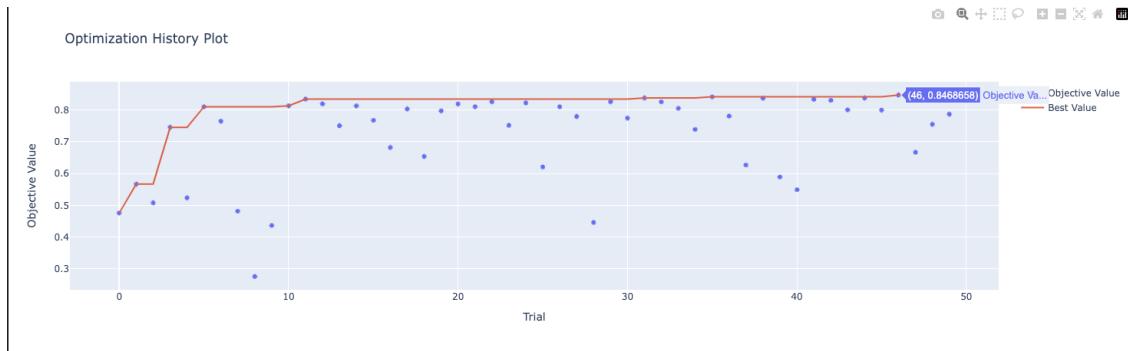


Figure 9: Historia optymalizacji Optuny

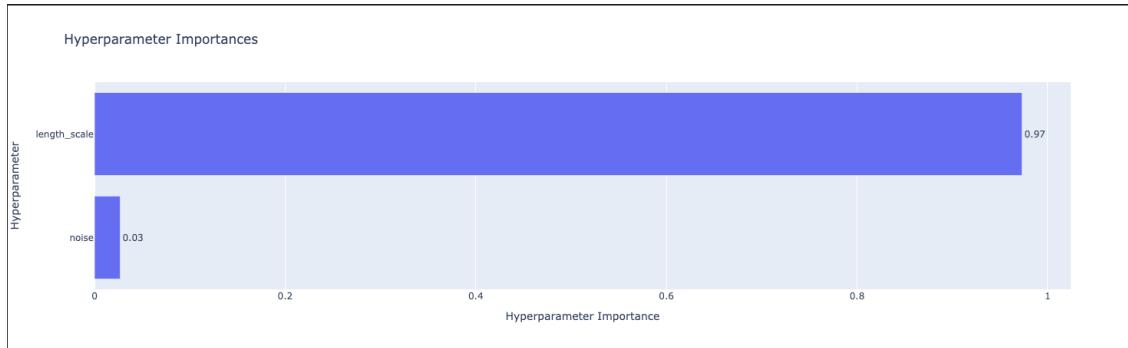


Figure 10: Ważność hiperparametrów

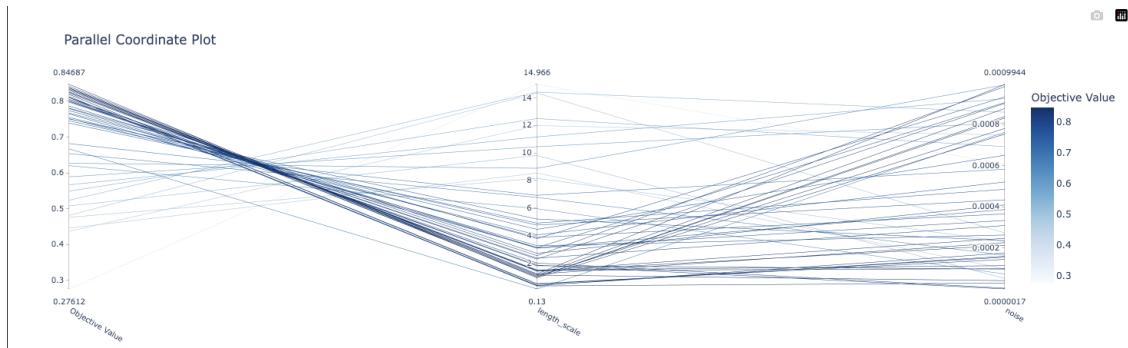


Figure 11: Łączna zależność parametrów od metryki F-1 score

Okazało się, że najlepsze hiperparametry to $\text{scale}=0.955$ oraz $\text{noise}=0.00094$.

3.2.3 Finalna ewaluacja

Finalnie wytrenowano model z najlepszymi hiperparametrami na zbiorze treningowym i walidacyjnym. Wytrenowany model został zwalutowany na zbiorze testowym. Uśredniono metryki ewaluacji na 20 powtórzeniach trenowania. Rezultaty finalnej ewaluacji znajdują się w Tabeli 2.

Metryka	Wartość
Accuracy	0.811
F1 Score	0.803
Precision	0.802
Recall	0.811
AUC	0.952

Table 5: Ewaluacja GP na zbiorze testowym

3.2.4 Wnioski

Na podstawie powyższej analizy można wyciągnąć wiele wniosków.

1. Model osiąga bardzo dobre wyniki dla powyższego zbioru nawet na domyślnych parametrach (0.821 F-Score).
2. Jeśli chodzi o parametry modelu to ze strojenia wynika, że najlepsze wyniki ewaluacji dla parametru scale są od jego wartości 0.5 do 1.0
3. Parametr noise mało wpływa na spadek metryk klasyfikacji. Jego zmiany wpływają na accuracy w zakresie 8 punktów procentowych.
4. W ogólności model sprawuje się bardzo dobrze, osiągając z najlepszymi hiperparametrami 0.803 accuracy na danych testowych.

3.3 Bayesowska Sieć Neuronowa (BNN)

3.3.1 Ustalenie scenariusza eksperymentalnego

Zbiór danych został podzielony na:

1. Zbiór treningowy - 70% zbioru danych, używany do trenowania modelu
2. Zbiór walidacyjny - 15% zbioru danych, używany do walidacji wyników modelu podczas przeszukiwania hiperparametrów
3. Zbiór testowy - 15% zbioru danych, używany do liczenia finalnych metryk

Do danych kategorycznych użyliśmy ordinal encoding.

3.3.2 Wybór funkcji kosztu do ewaluacji modelu

Podczas przeglądu hiperparametrów użyliśmy ELBO (Evidence Lower Bound). Tym mniejsza wartość ELBO tym lepszy model.

3.3.3 Przegląd hiperparametrów modelu

Do przeglądu hiperparametrów modelu użyliśmy ręcznie zaimplementowanego grid search. Poniższa tabela przedstawia wszystkie porównywane hiperparametry wraz z ich możliwymi wartościami.

Nazwa	Opis	Przeszukiwane wartości
<code>lr</code>	Współczynnik uczenia	0.01, 0.001, 0.0001
<code>num_hidden_features</code>	Liczba neuronów w warstwie ukrytej	64, 128, 256
<code>sigma_1</code>	Wartość odchylenia standardowego pierwszego rozkładu normalnego w Gaussian Mixture Model	0.5, 1
<code>sigma_2</code>	Wartość odchylenia standardowego drugiego rozkładu normalnego w Gaussian Mixture Model	0.000001, 0.001
<code>mixing</code>	Tym wyższa wartość tego parametru tym bardziej czerpiemy z pierwszego rozkładu normalnego w Gaussian Mixture Model. Dla wartości 1 Gaussian Mixture Model staje się rozkładem normalnym o odchyleniu standardowym równym <code>sigma_1</code>	0.2, 0.5, 0.8, 1

Table 6: Analizowane hiperparametry

Każdy model był trenowany przez 30 epok na seedzie równym 0. Rozmiar batcha wynosił 32. Po wytrenowaniu modeli porównaliśmy wartość ELBO dla zbioru walidacyjnego i wybraliśmy najlepszy model. Poniżej przedstawiamy wartości hiperparametrów najlepszego modelu

Nazwa hiperparametru	Wartość hiperparametru
<code>lr</code>	0.01
<code>num_hidden_features</code>	256
<code>sigma_1</code>	0.5
<code>sigma_2</code>	0.000001
<code>mixing</code>	1

Table 7: Wartości hiperparametrów najlepszego modelu

Warto zauważyć, że dla wartości `mixing` równej 1 wartość `sigma_2` nie ma żadnego znaczenia gdyż nie wpływa ona w żaden sposób na model.

3.3.4 Wizualizacja wyników

Poniżej znajdują się wyniki trenowania najlepszego modelu na zbiorze treningowym i walidacyjnym.

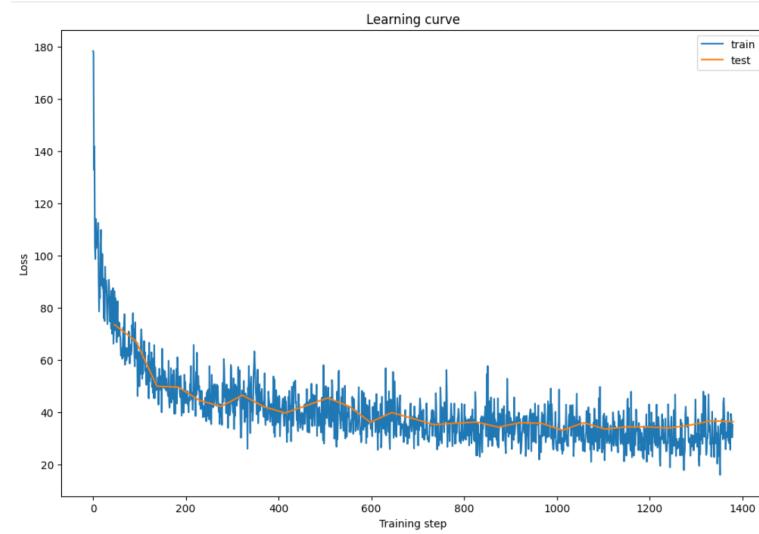


Figure 12: Funkcja kosztu od kroku trenowania

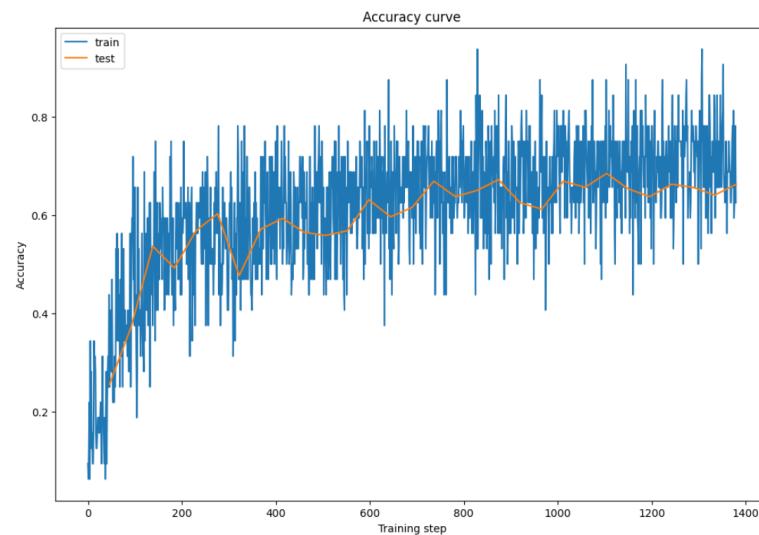


Figure 13: Dokładność od kroku trenowania

Obliczyliśmy też standardowe wyniki klasyfikacyjne dla zbioru testowego. Wyniki są przedstawione w poniższej tabeli.

Metryka	Wartość
Accuracy	0.653
F1 Score	0.653
Precision	0.653
Recall	0.653
AUC	0.923

Table 8: Ewaluacja BNN na zbiorze testowym

Sprawdziliśmy najbardziej pewne i niepewne predykcje. Analizowaliśmy poprawne i niepoprawne predykcje.

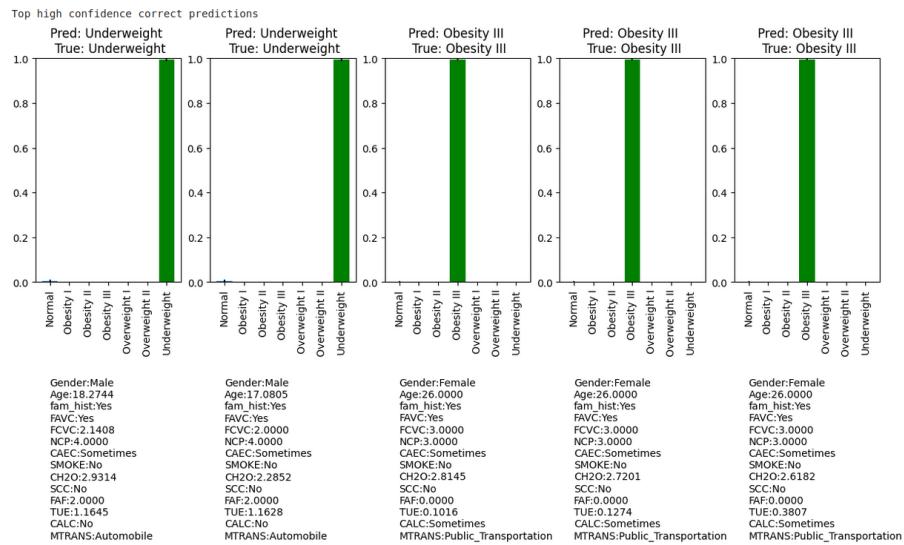


Figure 14: Najbardziej pewne poprawne predykcje

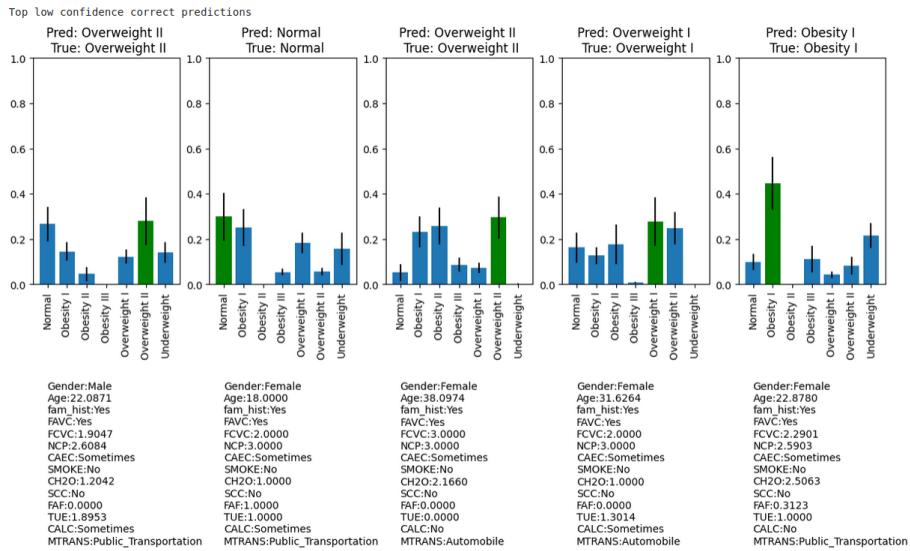


Figure 15: Najmniej pewne poprawne predykcje

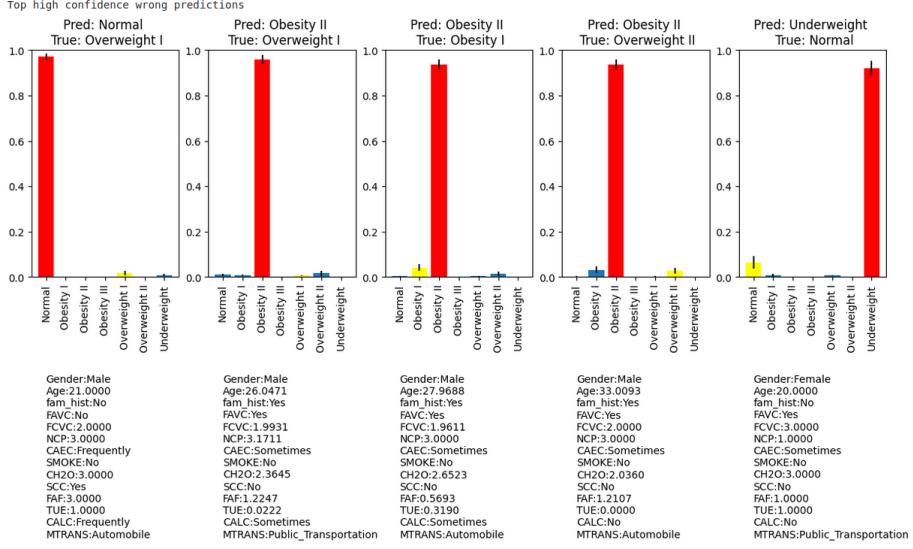


Figure 16: Najbardziej pewne niepoprawne predykcje

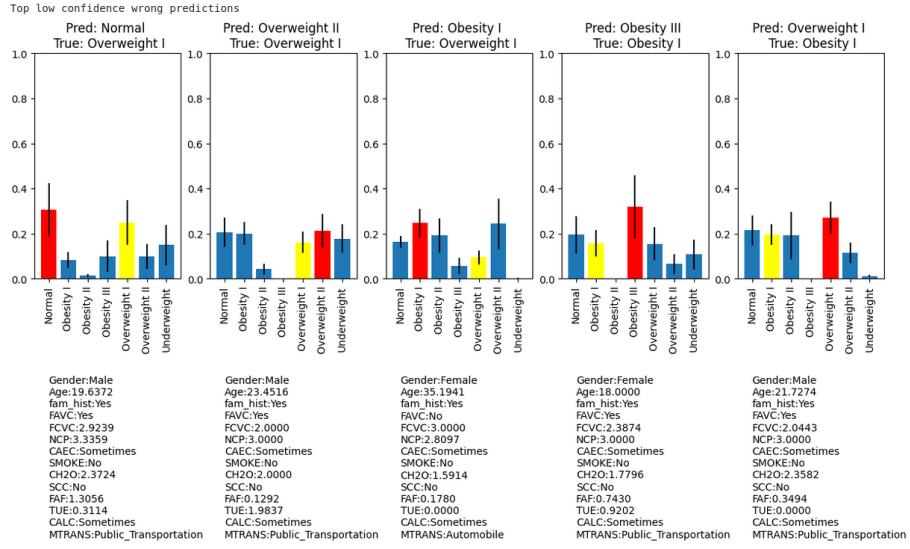


Figure 17: Najmniej pewne niepoprawne predykcje

Wykonaliśmy też analizę credible intervals na poziomie 90%. Głównymi problemami podczas dokonywania tej analizy było:

1. Zbyt duża wymiarowość danych. Żeby uwzględnić całość naszych danych z predykcjami musielibyśmy zrobić wykres 15-wymiarowy
2. Wyjście modelu to wartość przypisana do każdej klasy. Tym większa wartość tym większe prawdopodobieństwo, że to jest prawidłowa klasa. Trudno będzie na jednym wykresie uwzględnić wszystkie 7 klas

Mając na uwadze powyższe postanowiliśmy w następujący sposób przeprowadzić analizę credible intervals:

1. Każda klasa będzie analizowana oddzielnie.
2. Zmniejszymy wymiarowość danych wejściowych do jednego wymiaru z użyciem algorytmu PCA (Principal Component Analysis), żeby wykres końcowy był dwuwymiarowy. Niestety musimy pogodzić się z tym, że stracimy dużo informacji z naszych danych.

Poniżej znajdują się wykresy credible intervals:

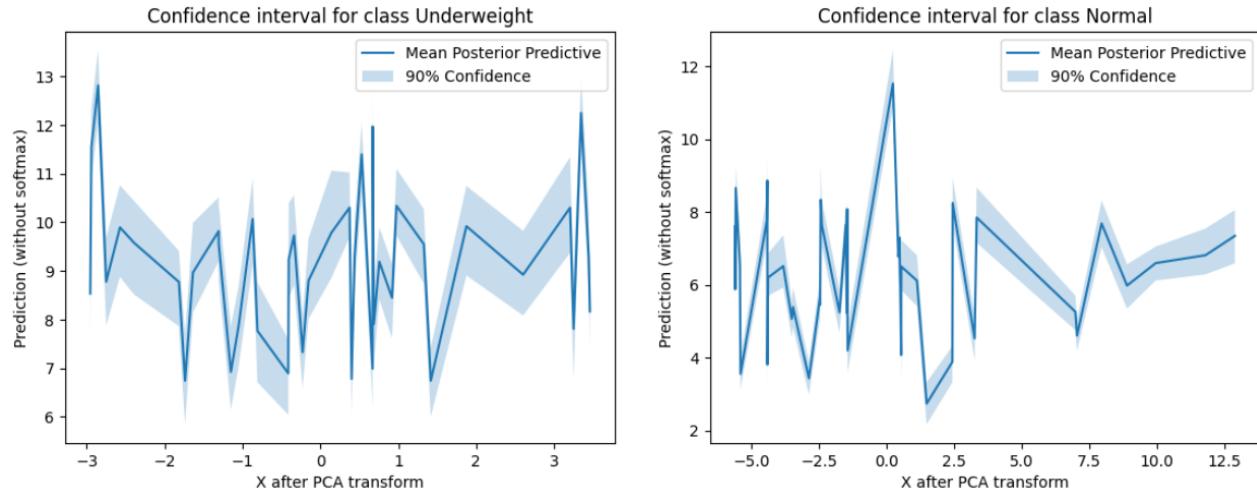


Figure 18: Confidence intervals dla klasy "un-
derweight"

Figure 19: Confidence intervals dla klasy "nor-
mal"

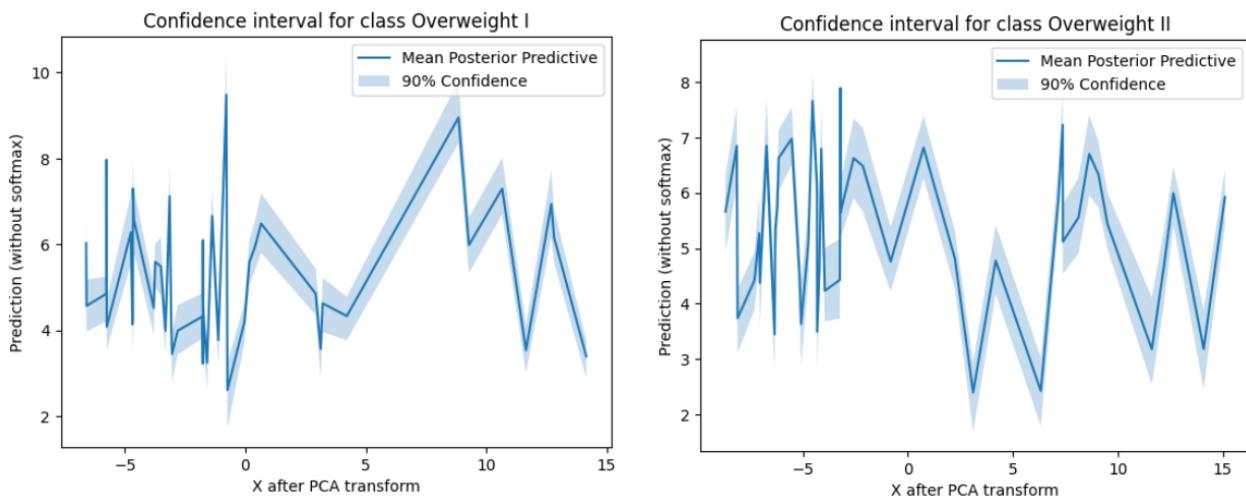


Figure 20: Confidence intervals dla klasy "over-
weight I"

Figure 21: Confidence intervals dla klasy "over-
weight II"

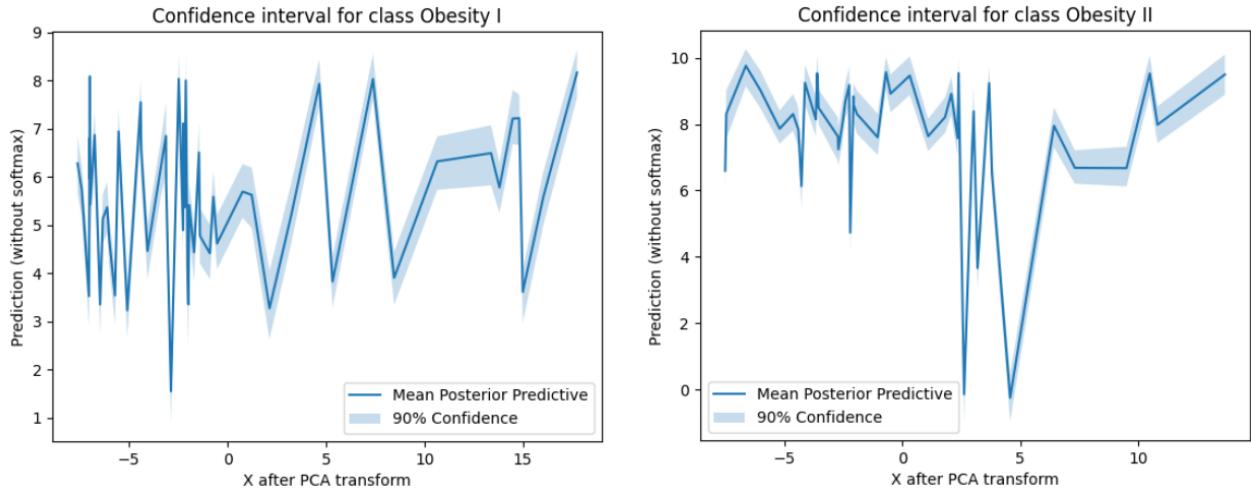


Figure 22: Confidence intervals dla klasy "obesity I"

Figure 23: Confidence intervals dla klasy "obesity II"



Figure 24: Confidence intervals dla klasy "obesity III"

3.3.5 Wnioski

Patrząc na wyniki modelu możemy stwierdzić, że:

1. mamy lepsze wyniki niż losowe zgadywanie klasy jednak nie są to zadowalające wyniki. Wynik F1 równy 0.653 nie pozwala na praktyczne zastosowanie tego modelu
2. model ma równe wyniki precision, recall, accuracy i F1. Oznacza to, że model jednakowo się myli w przypadku każdej klasy.

Patrząc na analizę niepewności i credible intervals możemy zauważyc, że:

1. model często myli się pomiędzy zbliżonymi klasami np. predykuje "Obesity I" zamiast "Obesity II"
2. model dla większości klas ma stabilne predykcje. Najbardziej niepewne predykcje są dla klasy "Underweight" i "Obesity III". Jednak nie zauważymy, żeby to wpływało na osiągnięcia modelu. Może to wynikać z tego, że musieliśmy znaczco zmniejszyć wymiarowość danych i przez utratę informacji nasze wykresy nie reprezentują dokładnie zbioru danych.

3. mimo w większości wypadków stabilnych predykcji model wciąż się myli. Możliwe że nasz model jest zbyt mało skomplikowany, żeby w pełni nauczyć się postawionego problemu.

3.4 Naive Bayes (NB)

3.4.1 Wstępny wybór architektury

Wstępny wybór rodzaju modelu, oraz swojego rodzaju baseline, zostanie wykonany na implementacji dostępnej w `scikit-learn`. Biblioteka ta udostępnia 4 rodzaje modeli Naiwnego Bayesa: `GaussianNB`, `MultinomialNB`, `BernoulliNB`, `CategoricalNB`. Dodatkowo trzy rodzaje preprocessingu zostaną sprawdzone: dwa sposoby kategoryzacji oraz binaryzacja.

Preprocessing	Model	Accuracy	Precision	Recall	F1	AUC
one hot	GaussianNB	0.498	0.576	0.498	0.455	0.656
one hot	MultinomialNB	0.539	0.553	0.539	0.519	0.685
one hot	BernoulliNB	0.536	0.570	0.536	0.532	0.681
one hot	CategoricalNB	0.570	0.568	0.570	0.563	0.705
ordinal	GaussianNB	0.539	0.589	0.539	0.518	0.683
ordinal	MultinomialNB	0.356	0.326	0.356	0.313	0.572
ordinal	BernoulliNB	0.479	0.527	0.479	0.436	0.648
ordinal	CategoricalNB	0.596	0.583	0.596	0.584	0.722
ordinal 10 bins	GaussianNB	0.552	0.604	0.552	0.533	0.692
ordinal 10 bins	MultinomialNB	0.406	0.434	0.406	0.356	0.614
ordinal 10 bins	BernoulliNB	0.479	0.482	0.479	0.453	0.645
ordinal 10 bins	CategoricalNB	0.627	0.620	0.627	0.608	0.744

Table 9: Wstępne eksperymenty na bibliotece `scikit-learn`

Najlepsze wyniki $F1$ uzyskano dla modelu `CategoricalNB`, który zostanie zaimplementowany. Deskrytywacja kodowania `Ordinal` przyniosła najlepsze rezultaty.

Ilość przedziałów	Accuracy	Precision	Recall	F1	AUC
50	0.606	0.601	0.606	0.587	0.730
20	0.640	0.631	0.640	0.618	0.753
10	0.627	0.620	0.627	0.608	0.744
5	0.615	0.608	0.615	0.600	0.735

Table 10: Dobór ilości przedziałów dyskretyzacji

Dyskretyzacja 20-przedziałowa zostanie zastosowana.

3.4.2 Ewaluacja dystrybucji

Model zaimplementowano wykorzystując `pyro`. Sprawdzono trzy wersje użytych rozkładów:

1. C - Wszystkie parametry z rozkładu `Categorical` (tak jak pokazały eksperymenty ze `scikit-learn`).
2. $C + G$ - Parametry zdefiniowane jako kategoryczne w danych korzystają z `Categorical`, natomiast parametry numeryczne z `Gaussian` (zgodnie z intuicją).
3. G - Wszystkie parametry z rozkładu `Gaussian` (dla porównania).

Podejście	Accuracy	Precision	Recall	F1	AUC
C	0.603	0.589	0.603	0.586	0.727
C+G	0.533	0.539	0.533	0.502	0.681
G	0.539	0.591	0.539	0.514	0.687

Table 11: Porównanie zaimplementowanych modeli w `pyro`

Wykorzystanie samych rozkładów kategorycznych przyniosło najlepsze wyniki.

3.4.3 Analiza hiperparametrów

Loss

Jako funkcję straty wykorzystano `Trace_ELBO`. Domyślnie funkcja ma m.in. parametry `num_particles=1` i `vectorize_particles=False`. Przetestowano zwiększenie liczby próbek oraz włączenie wektoryzacji. Nie spowodowało to żadnych zmian w wynikach.

Learning rate

Przeanalizowano wartości pomiędzy 0.1 a 0.00001. Najlepsze wyniki osiągnięto dla najwyższej wartości.

lr	Accuracy	Precision	Recall	F1	AUC
1e-5	0.426	0.420	0.426	0.417	0.618
1e-4	0.467	0.448	0.467	0.451	0.644
1e-3	0.568	0.555	0.568	0.555	0.708
1e-2	0.603	0.589	0.603	0.586	0.727
1e-1	0.603	0.595	0.603	0.589	0.731

Table 12: Porównanie learning rate

Weight decay

Sprawdzono czy dodanie weight decay jest w stanie poprawić wyniki i osiągnięto nieznaczną poprawę.

wd	Accuracy	Precision	Recall	F1	AUC
0	0.603	0.595	0.603	0.589	0.731
1e-3	0.606	0.596	0.606	0.591	0.731
1e-4	0.603	0.595	0.603	0.589	0.731

Table 13: Porównanie weight decay

3.4.4 Podsumowanie

Finalnie osiągnięto accuracy rzędu 60%. Tak wyglądały krzywe uczenia.

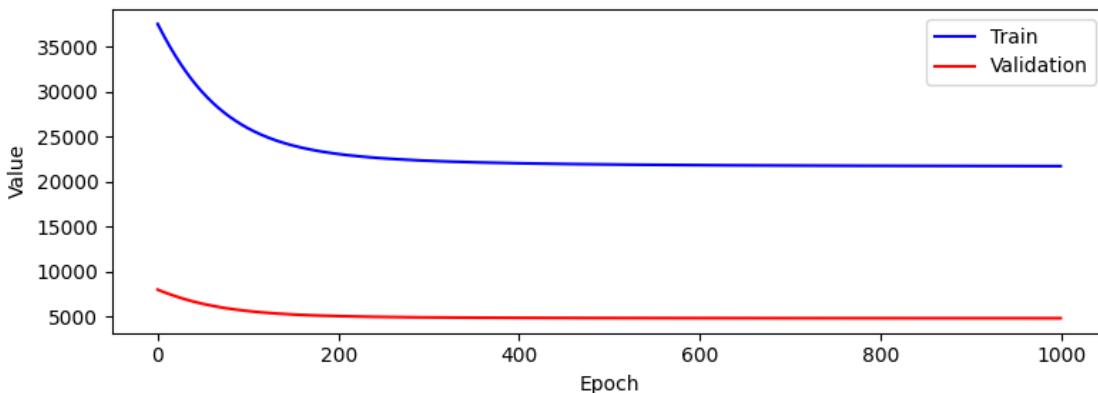


Figure 25: Wartość funkcji straty podczas uczenia dla zbioru treningowego i walidacyjnego

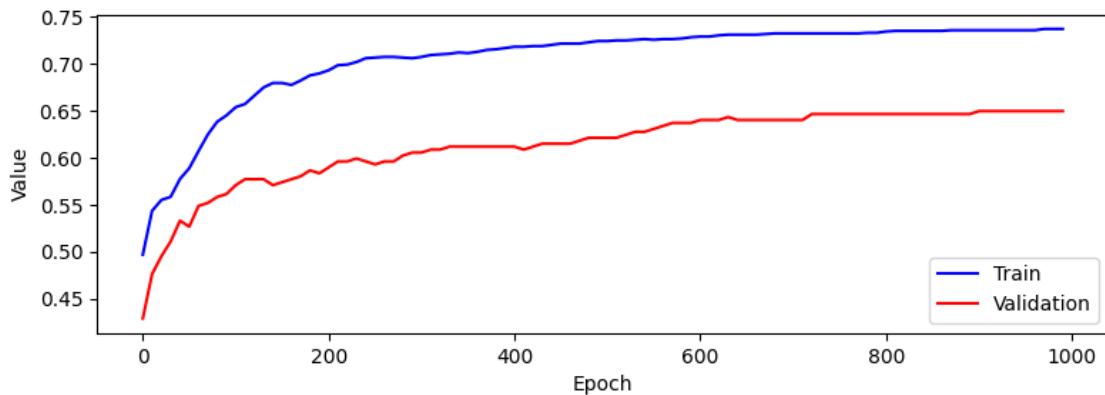


Figure 26: Wartość accuracy podczas uczenia dla zbioru treningowego i walidacyjnego

Niestety model *Naive Bayes* okazał się zbyt naiwny aby osiągnąć lepsze wyniki. Niemniej jednak 60% accuracy przy klasyfikacji 7-klasowej jest całkiem dobrym wynikiem.

4 Porównanie wyników modeli

Model	Accuracy	F1 Score	Precision	Recall	AUC
GP	0.811	0.803	0.802	0.811	0.952
Naive Bayes	0.606	0.591	0.596	0.606	0.731
Bayesian NN	0.653	0.653	0.653	0.653	0.923

Table 14: Metryki jakości modeli z najlepszymi hiperparametrami na zbiorze testowym

Widzimy, że najlepsze wyniki osiągnął Gaussian Processes. Widzimy więc, że nie zawsze najbardziej zawiły model osiąga najlepsze rezultaty.