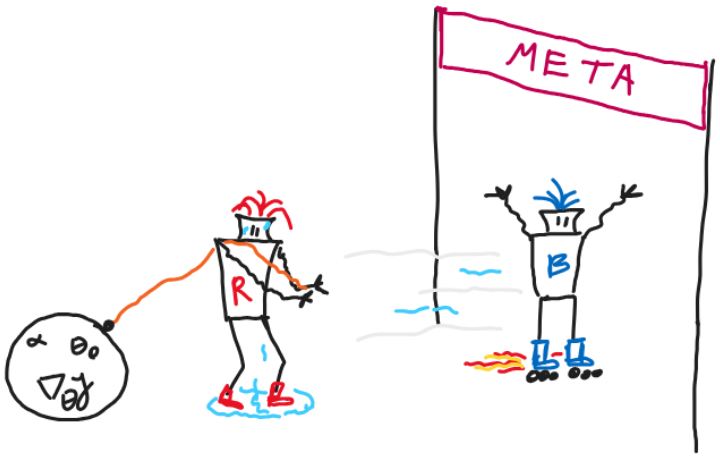


Rysunek 1: Tym razem to mój szkic

Miniprojekt 2: Klasyfikacja binarna

Metody Probabilistyczne w Uczeniu Maszynowym

Szymon Szulc

7 maja 2025

Spis treści

1 Wstęp

Niniejszy raport oparty jest na notatniku `main.ipynb`. Raport ma stanowić zwięzłe i czytelne podsumowanie mojej pracy nad problemem klasyfikacji binarnej korzystając z naiwnego klasyfikatora bayesowskiego oraz regresji logistycznej.

2 Badany problem

2.1 Definicja

Dane to wyniki badań diagnostycznych pozwalających stwierdzać, czy wykryty rak piersi jest łagodny czy złośliwy.

2.2 Założenia

Formalnie:

$y \in \{2, 4\}$, gdzie $y = 4$ to rak złośliwy

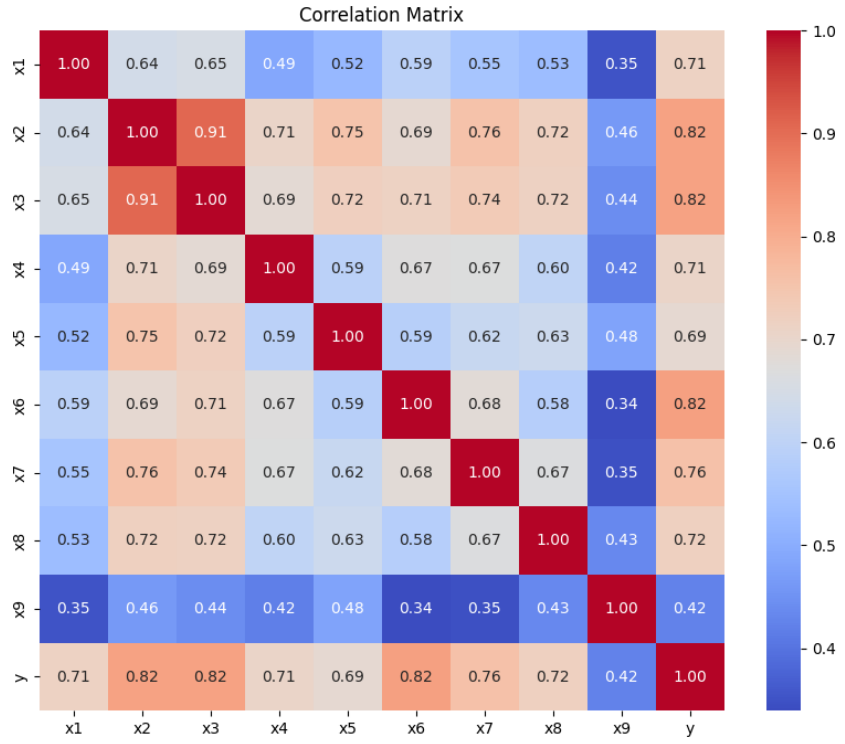
$\forall j \in \{1, \ldots, 9\}\ \ x_j \in \{1, \ldots, 10\}$, mamy do czynienia z cechami dyskretnymi

$p(x_j = d | y = c) = \phi^j_{c,d}$, gdzie $\sum_{d=1}^{10} \phi^j_{c,d} = 1$

czyli zmienne $x_j | y = c$ mają rozkład wielomianowy.

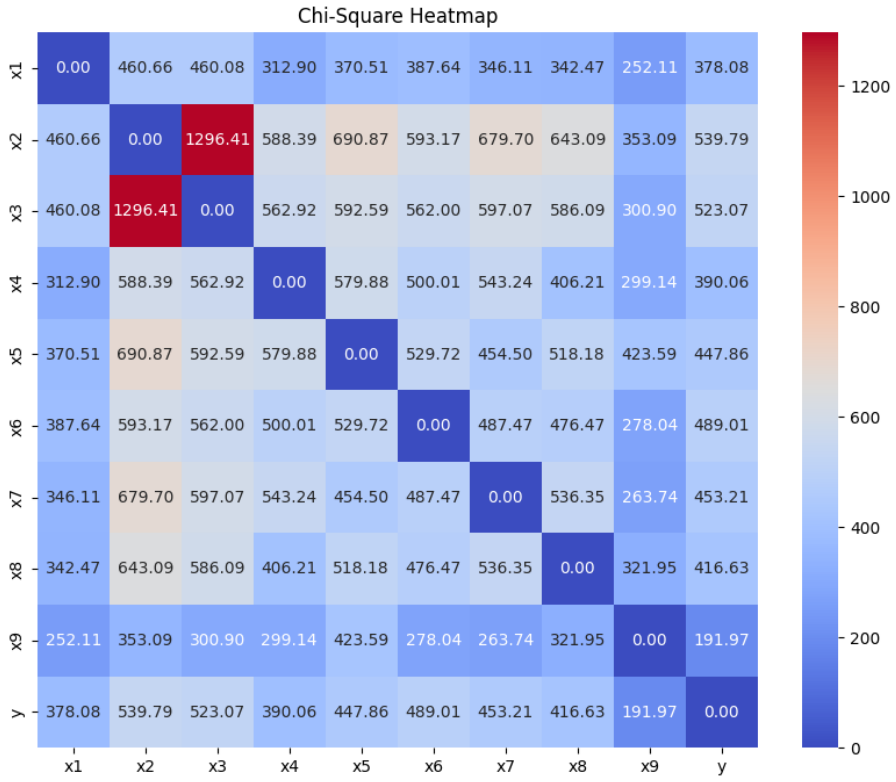
3 Pierwszy kontakt z danymi

Mamy 9 cech – $x_1, \ldots, x_9$ i zmienną binarną y. Pierwsze co zrobiłem to macierz korelacji, którą znamy już z poprzedniego miniprojektu.

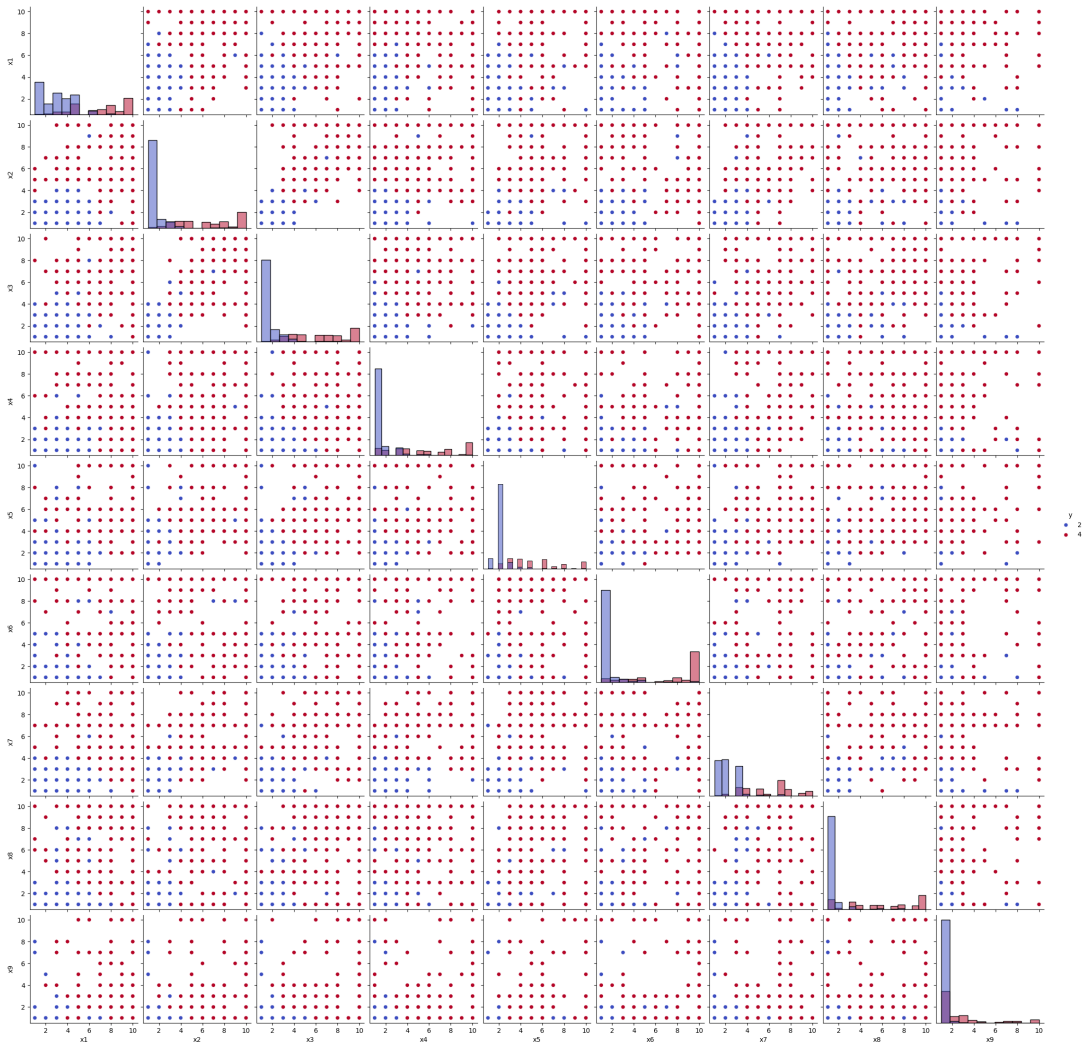


Rysunek 2: Macierz korelacji Pearsona

Ale zdałem sobie sprawę, że korelacja Pearsona może prowadzić do błędnych wniosków. Znalazłem test chi-kwadrat, który sprawdza zależność między zmiennymi.

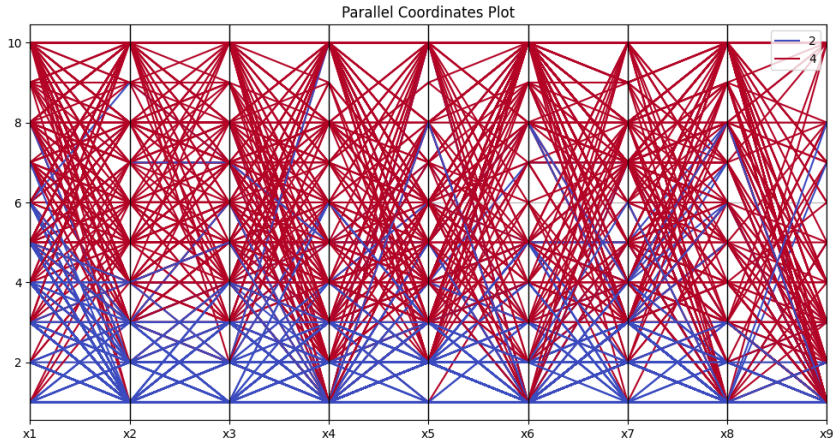


Rysunek 3: Test chi-kwadrat



Rysunek 4: Pary cech i y

I dostaliśmy te same wnioski – x_2 i x_3 są mocno zależne, a na y wysoki wpływ mają x_2, x_3 i x_6. Następnie chciałem sprawdzić czy dane są liniowo separowalne i na oko **SĄ**, co odzwierciedla kolejny wykres. Mam świadomość, że nie jest on do końca czytelny przez rozmiar, ale po przybliżeniu widać, że w każdym kwadraciku jesteśmy w stanie narysować granicę. Jeśli popatrzymy tylko na przekątną to możemy dojść do wniosku, że jeśli $y = 2$ to $\forall j \in \{1, \ldots, 9\}\ x_j < 5$. Możemy to poprzeć już ostatnim wykresem.



Rysunek 5: Wykres równoległych cech

4 Podział danych

Dokonałem losowego podziału danych (67% – zbiór treningowy, 33% zbiór testowy, zachowując również taki stosunek w obrębie klas y) 10 razy, żeby uśrednić wyniki. Sam podział danych wygląda następująco:

1: $i \leftarrow \frac{2}{3}m$
2: $XY_2 \leftarrow$ `np.random.shuffle`$(XY[y == 2])$
3: $XY_4 \leftarrow$ `np.random.shuffle`$(XY[y == 4])$
4: $\text{train}_2, \text{test}_2 \leftarrow XY_2[:i], XY_2[i:]$
5: $\text{train}_4, \text{test}_4 \leftarrow XY_4[:i], XY_4[i:]$
6: $\text{train} \leftarrow \text{train}_2 \cup \text{train}_4$
7: $\text{test} \leftarrow \text{test}_2 \cup \text{test}_4$

5 Naiwny klasyfikator bayesowski

Będę korzystał z wariantu z wygładzeniem Laplace'a.

$$p(y = c) = \frac{1 + \sum_{i=1}^{m} \mathbb{1}[y^{(i)} = c]}{|C| + m}$$

$$p(x_j = d | y = c) = \frac{1 + \sum_{i=1}^{m} \mathbb{1}[y^{(i)} = c \wedge x_j^{(i)} = d]}{|D| + \sum_{i=1}^{m} \mathbb{1}[y^{(i)} = c]}$$

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}, \qquad \text{co umiemy policzyć}$$

W tym przypadku zwracamy 4, jeśli $p(y = 4|x) > \frac{1}{2}$. W implementacji nie ma żadnych trudności. Możemy usprawnić zliczanie korzystając z `np.bincount()` oraz `np.apply_along_axis()`.

Błąd będę liczył tak jak w artykule:

$$\texttt{error} = \frac{\sum_{i=1}^{m} \mathbb{1}[\hat{y^{(i)}} \neq y^{(i)}]}{m}$$

Taki naiwny klasyfikator bayesowki daje:

`Naive Bayes average error:` 0.0274

`Naive Bayes average training error:` 0.0214

Co wydaje się bardzo dobrym wynikiem. Zanim spojrzymy na dokładność, precyzję i czułość pare słów o problemach z SciKitLearnem. Biblioteka udostępnia `MultinomialNB`, który wypada gorzej niż nasz klasyfikator.

`Sklearn Naive Bayes average error:` 0.0796

`Sklearn Naive Bayes average training error:` 0.1028

Czemu tak się dzieje? Otóż `MultinomialNB` zlicza cechy jakby one były z $\{0, 1\}$. My wiemy więcej o cechach, stąd lepszy wynik.
Z racji tego, że badamy złośliwość raka zależy nam na wysokiej czułości.

`Naive Bayes average accuracy:` 0.9726

`Naive Bayes average precision:` 0.9413

`Naive Bayes average sensitivity:` 0.9835

Bazowy model jest naprawdę dobry, ale co jeśli przesunęli byśmy granicę decyzyjną do $\hat{y} = 4$ jeśli $p(y = 4|x) > \frac{1}{5}$.

`Naive Bayes average accuracy:` 0.9739

`Naive Bayes average precision:` 0.9394

`Naive Bayes average sensitivity:` 0.9899

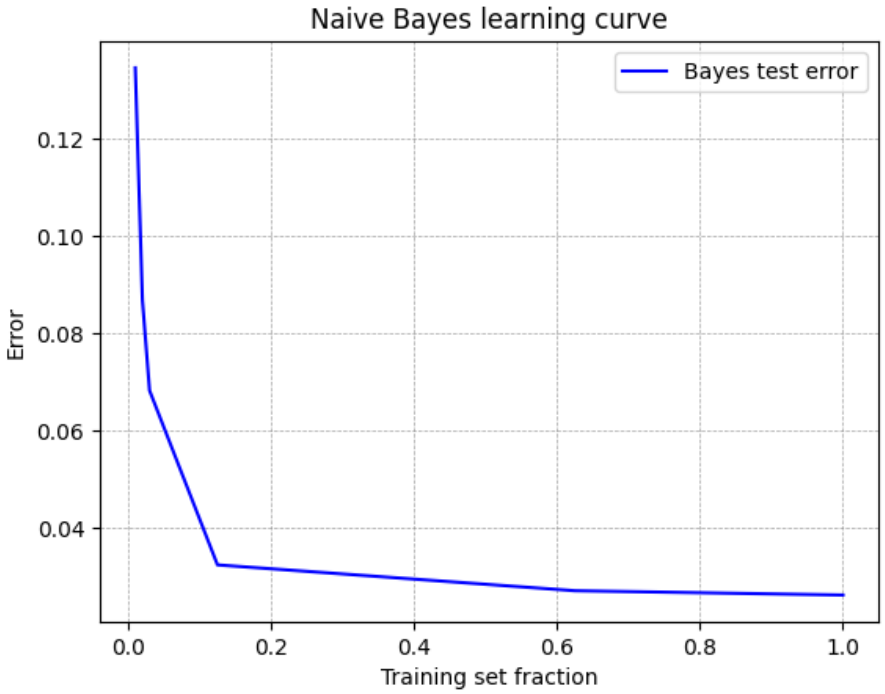
To nie jest duża poprawa, ale być może moglibyśmy wprowadzić taki hiperparametr.
Poniżej zamieszczam wycinek $p(y = 4|x)$ dla, losowego z 10 modeli:

```
0.0002501727910187334
0.9999955799911749
0.9999999999910805
0.999999999724131
3.95165834865358e-09
0.0002159627418231111
0.9999999988067703
3.95165834865358e-09
0.9999999995439866
1.7258835065009394e-08
```

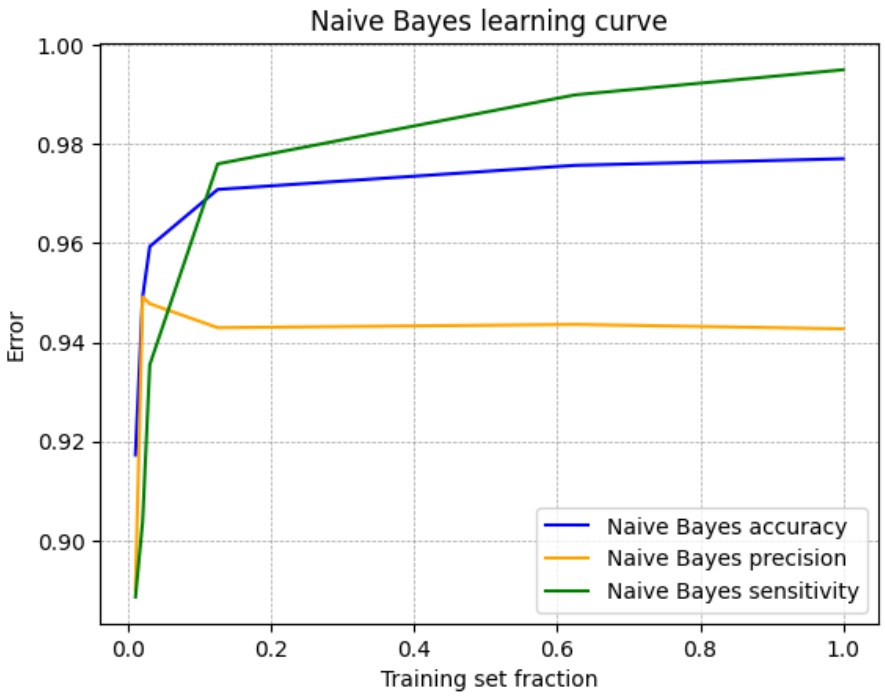
Klasyfikator jest naprawdę pewny swoich wyborów.
Dodatkowo uczy się bardzo szybko, na moim komputerze średni czas uczenia:

`Naive Bayes average fit time:` 0.0005753 `seconds`

Na zakończenie z Bayesem spójrzmy na wykresy uczenia z przesuniętą granicą decyzyjną.



Rysunek 6: Krzywa uczenia Bayesa 1



Rysunek 7: Krzywa uczenia Bayesa 2

6 Regresja logistyczna

To nic innego jak obłożenie regresji liniowej `sigmoidem`.

$$p(y = 4|x) = \frac{1}{1 + \exp(-(\theta^T x + \theta_0))}$$

Wynik zwracamy w taki sam sposób jak klasyfikator bayesowski.
Będziemy korzystać z następującej funkcji straty.

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} \log(\sigma(\theta^T x^{(i)} + \theta_0)) - (1 - y^{(i)}) \log(1 - \sigma(\theta^T x^{(i)} + \theta_0))$$

Wyszedłem z założeniem, że chce użyć następującego modelu: θ_0 + standaryzacja + L2 + GD, ale po drodze napotkałem okropne problemy.

1. Przepisując kod na klasy (za radą Mai) pomieszałem $\hat{y}$ i y, co kosztowało mnie $\approx$ 1h szukania hiperparametrów gradientu na marne.

2. Po rozwiązaniu tego problemu działała mi regresja bez θ_0, bez standaryzacji i bez L2. Ona dawała kiepskie wyniki, ale takie same jak SciKitLearn.

   `Logistic regression binary cross entropy test:` 0.3863

   `Logistic regression binary cross entropy train:` 0.4197

3. Jako następny rozwiązałem problem θ_0. Gradient nie mógł wejść w minimum, przez to, że pochodna po θ_0 była bardzo mała w stosunku do innych pochodnych (co ma sens, ponieważ jest ona postaci $\frac{1}{m}\sum_{i=1}^{m}\sigma(\theta^T x^{(i)} + \theta_0) - y^{(i)}$. Najpierw dałem jej wyższą wagę – przemnożyłem przez m – to już znacznie poprawiło zbieżność. Potem spróbowałem znormalizować gradient i to też poprawiało zbieżność. Z tych 2 pomysłów drugi wydaje się poprawniejszy formalnie, więc zostałem przy nim na chwilę.

   `Logistic regression binary cross entropy test:` 0.0657
   `Logistic regression binary cross entropy train:` 0.1035
   `Logistic regression error:` 0.0310
   `Logistic regression training error:` 0.0350
   `Logistic regression accuracy:` 0.9690
   `Logistic regression precision:` 0.9500
   `Logistic regression sensitivity:` 0.9620

4. Teraz nastąpił przełom, ponieważ naprawiłem standaryzację – zapomniałem zastosować skalowania w `predict()`. Otrzymujemy zawrotnie szybką zbieżność, zszedłem z 500 epok do 10.

5. Odpuściłem już L2 patrząc na wyniki, normalizacja gradientu również przestała być potrzebna.

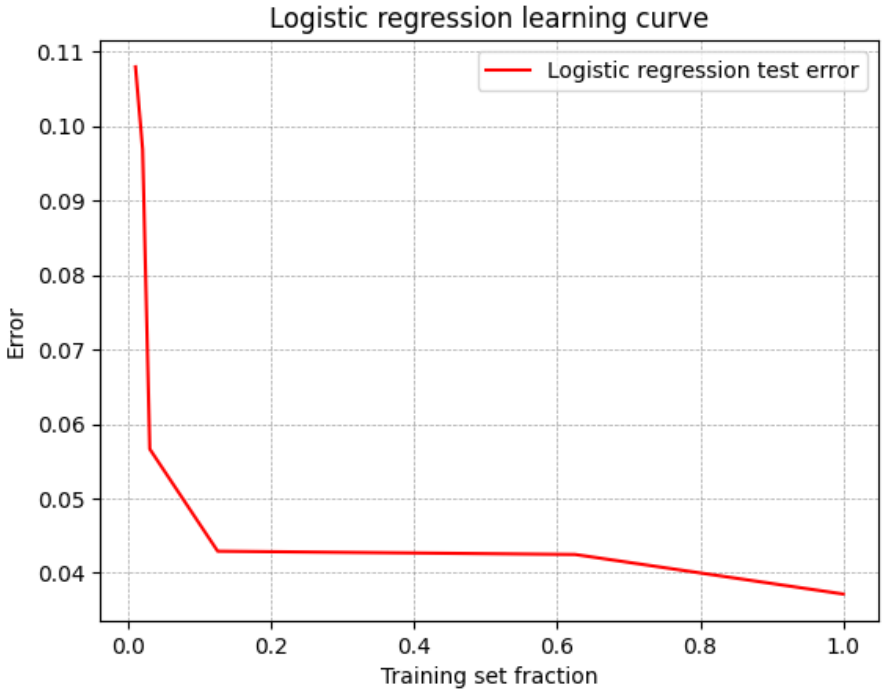
Podsumowując kończymy z modelem: GD($\alpha = 0.01, \text{epochs} = 10$) + standaryzacja (ta ze średnią i odchyleniem) + θ_0.
Teraz czas na wyniki.

`Logistic regression average error:` 0.0385
`Logistic regression average training error:` 0.0298
`Logistic regression average accuracy:` 0.9615
`Logistic regression average precision:` 0.9551
`Logistic regression average sensitivity:` 0.9342

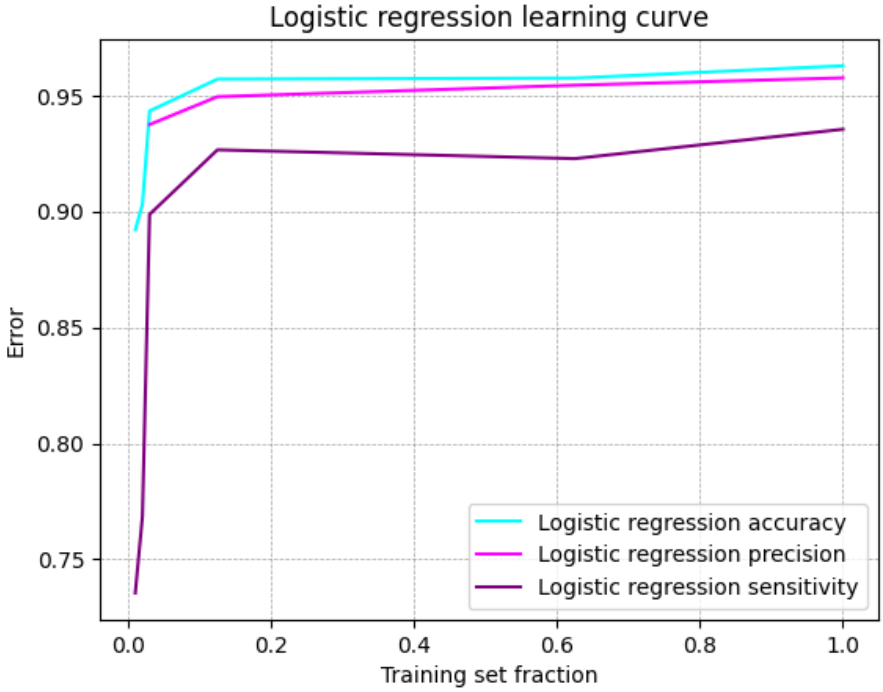
Mimo zejścia do 10 epok, regresja uczy się rząd wolniej niż naiwny klasyfikator bayesowski.

`Logistic regression average fit time:` 0.0016249 `seconds`

Na zakończenie z regresją spójrzmy na wykresy uczenia.



Rysunek 8: Krzywa uczenia regresji logistycznej 1



Rysunek 9: Krzywa uczenia regresji logistycznej 2

7 Regresja logistyczna vs naiwny Bayes

Naiwny klasyfikator bayesowski uczy się szybciej. Ponadto działa lepiej, co nie do końca umiem wytłumaczyć. Cechy są zależne od siebie (tak przynajmniej mówi test chi-kwadrat), ale łamią też założenie normalności, które zakłada regresja, być może jest ono silniejsze albo to ja nie umiem dobrać hiperparametrów.

`Logistic regression average error:` 0.0385

`Naive Bayes average error:` 0.0274

`Logistic regression average accuracy:` 0.9615

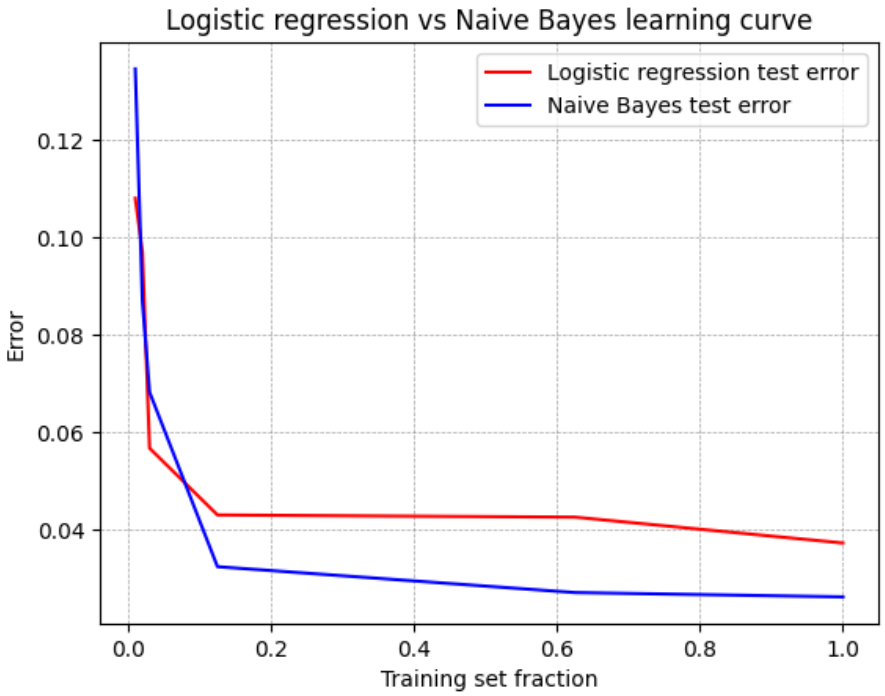
`Naive Bayes average accuracy:` 0.9739

`Logistic regression average precision:` 0.9551

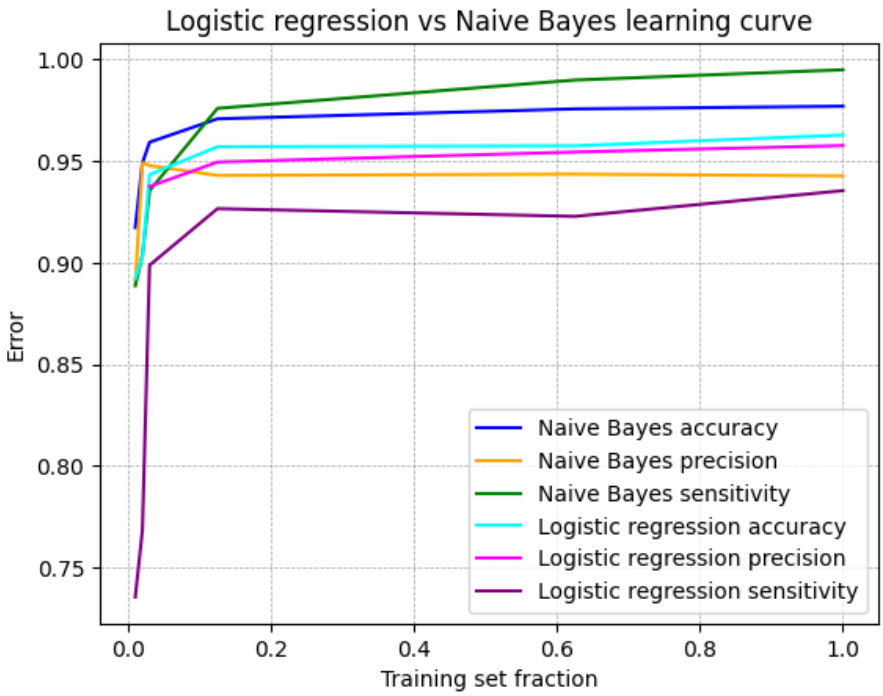
`Naive Bayes average precision:` 0.9394

`Logistic regression average sensitivity:` 0.9342

`Naive Bayes average sensitivity:` 0.9899



Rysunek 10: Regresja logistyczna vs naiwny Bayes 1

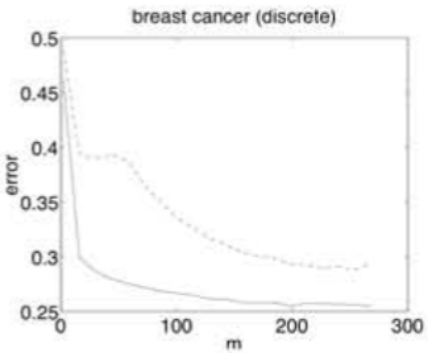


Rysunek 11: Regresja logistyczna vs naiwny Bayes 2

8 Jak to się ma do artykułu?

Artykuł *On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes* Andrew Nga i Michaela Jordana wskazuje na 2 sposoby oceny modelu. Autorzy zwracają uwagę, że modele generatywne mogą być lepsze, wbrew powszechnym przekonaniom, jeśli dobierzemy odpowiednią miarę. Osiągają one swoje, co prawda wyższe, minimum znacznie szybciej.

U mnie tego nie widać, nie dość, że naiwny klasyfikator bayesowski okazał się lepszy, to jescze te modele zbiegają w miarę równo. Ale jest nadzieja . . . ich wykres o raku piersi również nie popiera teoretycznych wyników.



Rysunek 12: Wykres autorów – przerywana linia to regresja logistyczna