
Rysunek 1: Szkic Roberta Crumba

# Miniprojekt 1: Regresja liniowa

Metody Probabilistyczne w Uczeniu Maszynowym

Szymon Szulc

9 kwietnia 2025

# Spis treści

# 1 Wstęp

Niniejszy raport oparty jest na notatniku `main.ipynb`. Raport ma stanowić zwięzłe i czytelne podsumowanie mojej pracy nad problemem regresji liniowej.

# 2 Regresja liniowa

## 2.1 Definicja

Dostaliśmy zbiór obserwacji $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ – `dane.data`. Na jego poodstawie checemy oszacować $y$ dla nowego wejścia $x$.

## 2.2 Model

Ja przez większość czasu rozważałem prosty model, gdzie dla $x = [x_1, \ldots, x_k]^T$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_k x_k.$$

Na sam koniec rozszerzyłem model o różne funkcje bazowe (wielomiany oraz $\sigma()$) –

$$h_\theta(x) = \theta_0 + \theta_1 \phi_1(x) + \cdots + \theta_d \phi_d(x).$$

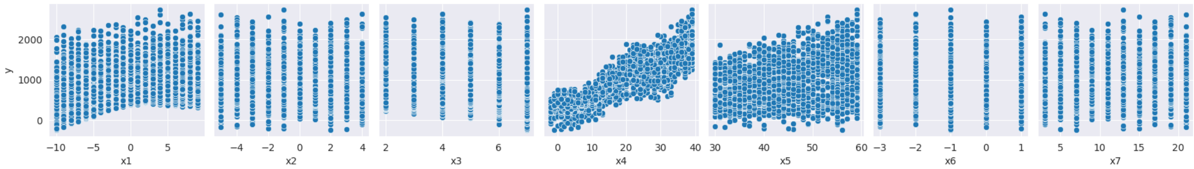
## 2.3 Funkcja straty

Ograniczyłem się do metody najmniejszych kwadratów jako funkcja straty –

$$\mathcal{J}(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2.$$

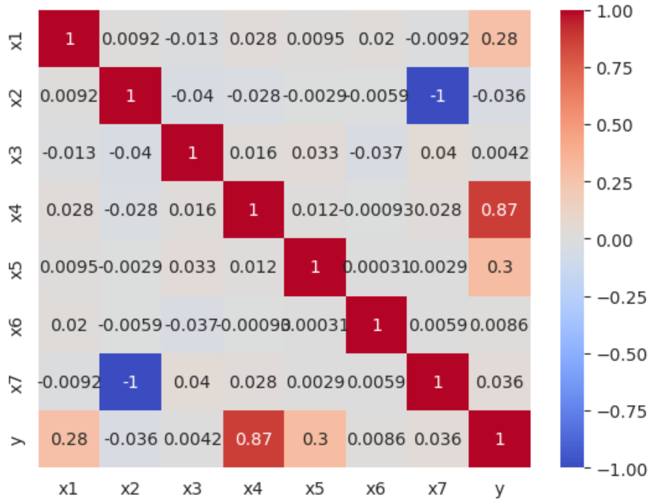
# 3 Pierwszy kontakt z danymi

W naszym przypadku mamy 7 cech – $x_1, \ldots, x_7$ i zmnienną objaśnianą $y$. Pierwsze, co zrobiłem, to 7 wykresów punktowych. Pojawiła się następująca obserwacja – $x_1, x_4, x_5$




Rysunek 2: Wykresy punktowe 7 cech vs zmienna objaśniana

powinny być istotne. Możemy w łatwy sposób sprawdzić tę hipotezę. Policzmy korelację tych zmiennych względem $y$. Biblioteka `pandas` udostępnia nam funkcję `corr()`, która pod spodem korzysta z korelacji Pearsona. Rzeczywiście $x_1, x_4, x_5$ są istotne.




Rysunek 3: Macierz korelacji Pearsona

# 4 Podział danych

Dane podzieliłem w sposób losowy na 3 zbiory – treningowy 60%, walidacyjny 20% oraz testowy 20% z pomocą funkcji `numpy.shuffle()`. Takich podziałów wykonałem 10, żeby uśrednić wyniki – coś na kształt walidacji krzyżowej.

# 5 Pierwsze modele

Celowo jeszcze nie skaluję danych.

Z wykładu wiemy, że dla prostej regresji możemy wyznaczyć $\theta$ analitycznie

$$\theta = (X^T X)^{-1} X^T y,$$

gdzie $X$ i $y$ to odpowiednio macierz planowania i wektor zmiennej objaśnianej. W przypadku bez $\theta_0$ nie dodajemy kolumny samych 1 do $X$. Jeśli $(X^T X)^{-1}$ nie istnieje to używamy pseudo-odwrotności Moore'a-Penrose'a – `numpy.pinv()`.

Poprawność została sprawdzona przy użyciu biblioteki `SciKit-Learn`.

## 5.1 Rozwiązanie analityczne bez $\theta_0$

$\theta = [23.6648, -101.5721, -6.4955, 39.0517, 18.7345, -0.8390, -49.5116]^T$

`Training loss:` 14120.0796

`Validation loss:` 15464.2807

`Test loss:` 14883.7154

## 5.2 Rozwiązanie analityczne z $\theta_0$

$\theta = [-22.0573, 23.6648, -97.5617, -6.4955, 39.0517, 18.7345, -0.8390, -47.5064]^T$

`Training loss:` 14120.0796

`Validation loss:` 15464.2807

`Test loss:` 14883.7154

## 5.3 Obserwacje

Niezależnie od $\theta_0$ dostajemy te same błędy, te same $\theta_1, \theta_4, \theta_5$. `SciKit` to potwierdza. Mam podejrzenia, że jest to związane z tym, że $x_1, x_4, x_5$ na wykresach przechodzą przez początek układu współrzędnych, a inne zmienne nie mają większego znaczenia.

# 6 Metoda spadku wzdłuż gradientu

Od teraz rozpatruję modele tylko z $\theta_0$.

## 6.1 Gradient Descent

$$\theta^{[k+1]} \leftarrow \theta^{[k]} - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

Ustaliłem z góry liczbę iteracji na `EPOCHS = 3000`.

I tutaj był mój pierwszy błąd. Nie potrafiłem ustawić odpowiednio małego $\alpha$, gradient eksplodował.

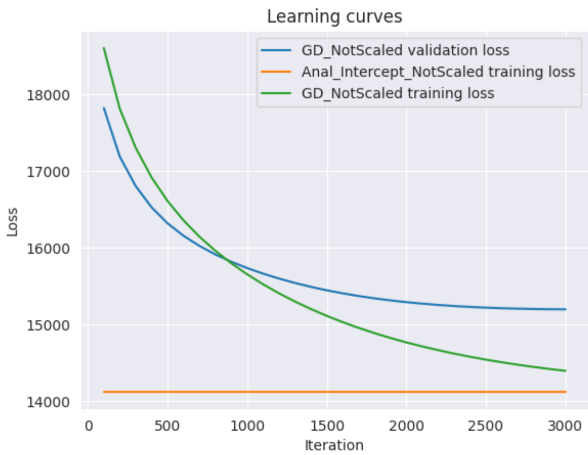
Ostatecznie skończyłem z takim wzorem

$$\theta^{[k+1]} \leftarrow \theta^{[k]} - \frac{\alpha}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x^{(i)}.$$

Jak się później okazało, wystarczyło $\alpha \leftarrow \frac{\alpha}{m}$, żeby sama suma działała poprawnie.

Przy pomocy zbioru walidacyjnego, wyszło mi, że $\alpha = 0.0007$.

Spójrzmy jak to zbiega do minimum. Jak widać na załączonym obrazku, zbiega powoli.




Rysunek 4: Zbieżność GD

Podejrzenia – cechy nie są przeskalowane i gradient nie ma jednolitych spadków.

**Drobna uwaga:** tym razem poprawność sprawdziłem korzystając z definicji pochodnej

$$\nabla \mathcal{J}(\theta) = \frac{\mathcal{J}(\theta + \epsilon) - \mathcal{J}(\theta - \epsilon)}{2\epsilon}.$$

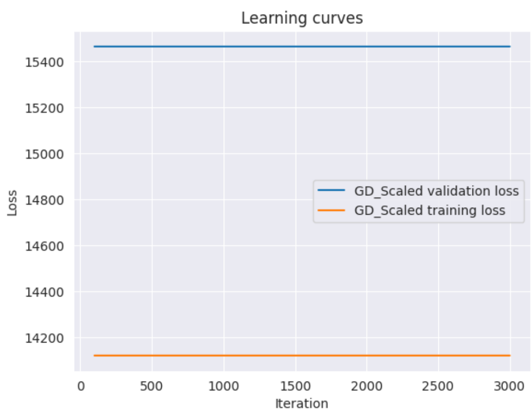
## 6.2 Standaryzacja cech

Zastosowałem znany i lubiany wzór

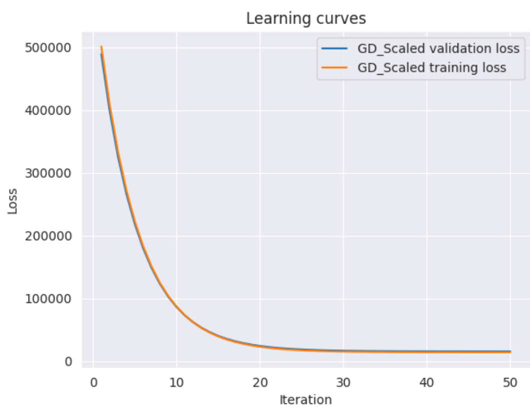
$$x^{(i)} \leftarrow \frac{x^{(i)} - \overline{x}}{\sigma(x)},$$

wszystko w obrębie kolumny. Średnia i odchylenie standardowe liczone są tylko na podstawie zbioru treningowego. $y$-ki nie zostały poddane standaryzacji.

Dla $\alpha = 0.1$ błąd na zbiorze walidacyjnym jest najmniejszy. Zobaczmy jak teraz GD zbiega. Standaryzowanie cech znacząco przyspiesza zbieżność, gradient wykonuje poprawny krok w wielu kierunkach.




(a) `EPOCHS=3000`




(b) `EPOCHS=50`

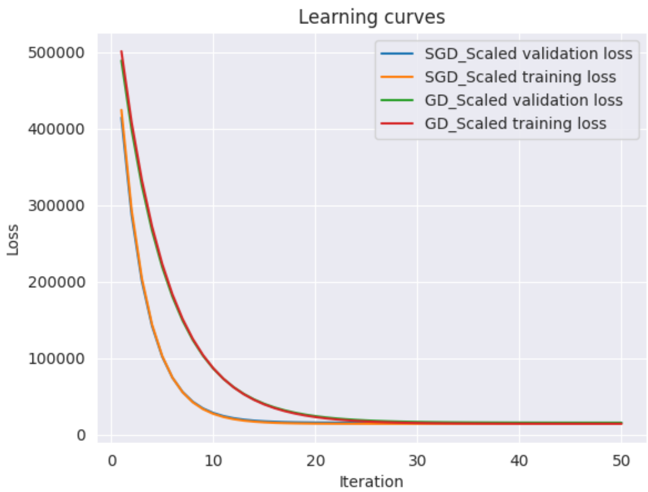
## 6.3 Stochastic Gradient Descent

Od zwykłego GD różni się tylko tym, że zamiast całego zbioru treningowego losujemy jego podzbiór. Dokładając trochę probabila liczymy na to, że to się będzie lepiej uogólniać.

Empirycznie wyznaczyłem $\alpha = 0.01$ oraz `batch_size` $= 64$. Zaciekawiła mnie mniejsza $\alpha$. Tłumaczę to sobie tym, że wylosowały się podzbiory z dużym krokiem gradientu i musieliśmy to zrównoważyć. Jak widać SGD zbiega szybciej. Obie metody gradientowe dają podobne wyniki na zbiorze testowym.

`Test loss [GD_Scaled]:` 14853.6827

`Test loss [SGD_Scaled]:` 14886.0377




Rysunek 6: GD vs SGD

# 7 Regularyzacje

Chcemy uniknąć zbytniego dopasowania do danych treningowych.

## 7.1 Grzbietowa (L2)

$$\mathcal{J}(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{k}\theta_j^2$$

Jak tak teraz patrzę na ten wzór to jest on inny niż na wykładzie i ja go świadomie zmieniłem, ponieważ analityczny nie brał średniej. Prawdopodobnie nie ma to znaczenia, wszystko nadrobią $\alpha, \lambda$.

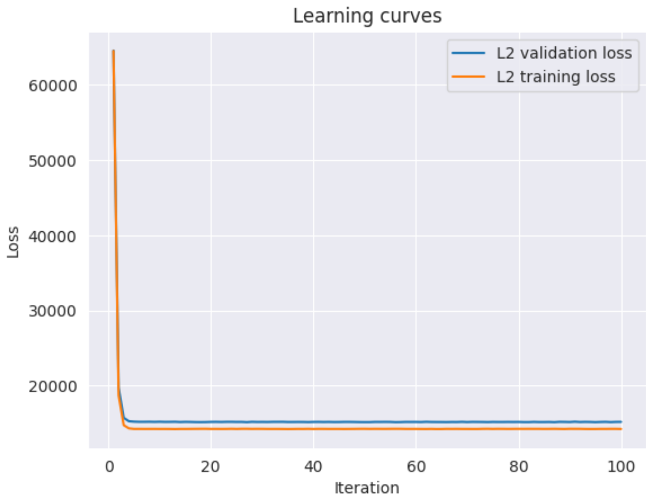
### 7.1.1 Rozwiązanie analityczne z $\theta_0$

$$J = I$$
$$J[0,0] = 0$$
$$\theta = (X^TX + \lambda J)^{-1}X^Ty$$

Stratę na jakimś zbiorze liczę już bez $||\theta_{-0}||_2^2$. Taki model daje nam najlepszy wynik dla $\lambda = 1$ (trochę skłamałem, ponieważ sprawdziłem tylko do $\lambda = 1$, a potem okazało się, że dla większych $\lambda$ jeszcze lepiej się uogólnia).

### 7.1.2 SGD

Hiperparametry: $\alpha = 0.001, \texttt{babtch\_size} = 64, \lambda = 1$




Rysunek 7: L2 SGD

Nieziemsko szybko zbiega do minimum.

### 7.1.3 L2 SGD vs SGD

Otrzymujemy mniejszą normę $\theta$ oraz mniejszy błąd na zbiorze testowym. Bardzo się cieszę, że tak wyszło.

`Norm [Analytic_Intercept_Scaled]:` 507.8657

`Norm [L2]:` 492.7032

`Test loss [L2]:` 14496.2205

`Test loss [SGD_Scaled]:` 14886.0377

## 7.2 Lasso (L1)

$$\mathcal{J}(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{k} |\theta_j|$$

Tutaj nie jesteśmy już w stanie użyć SGD.

### 7.2.1 Spadek względem współrzędnych

$$\hat{\theta}_j = \arg\min_{\theta_j \in R} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)}\right)^2 + \lambda \|\theta\|_1,$$

$$\hat{\theta}_j = \begin{cases} \frac{c_j+\lambda}{a_j} & \text{dla } c_j < -\lambda, \\ 0 & \text{dla } c_j \in [-\lambda, \lambda], \\ \frac{c_j-\lambda}{a_j} & \text{dla } c_j > \lambda, \end{cases}$$

$$a_j = 2\sum_{i=1}^{m} \left(x_j^{(i)}\right)^2, \quad c_j = 2\sum_{i=1}^{m} \left(x_j^{(i)} y^{(i)} - \theta_{-j}^T x_{-j}^{(i)}\right)$$

Ja pozwoliłem sobie ustawiać wszystkie $\theta_j$ w 1 iteracji.

### 7.2.2 Wyniki

Tutaj najbardziej oczekiwałem, że nieistotne cechy będą miały odpowiadające $\theta_i \approx 0$.
Hiperparametry: $\lambda = 256$

$$\theta = [970.2551, 134.6390, -5.9474, -10.9890, 461.6880, 162.7735, -1.2059, 1.3487]^T$$

Można powiedzieć, że efekt wyzerowania jest zadowalający.

## 7.3 Sieć elastyczna (Elastic Net)

$$\mathcal{J}(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \left(\alpha \sum_{j=1}^{k} \theta_j^2 + (1-\alpha) \sum_{j=1}^{k} |\theta_j|\right)$$

Korzystam z tego wzoru, ponieważ mogę rozszerzyć macierz planowania $X$ tak jak na ćwieczeniach i wrzucić do L1.

### 7.4 SGD vs L2 vs L1 vs Elastic Net

`Test loss [ElasticNet]:` 14869.5227

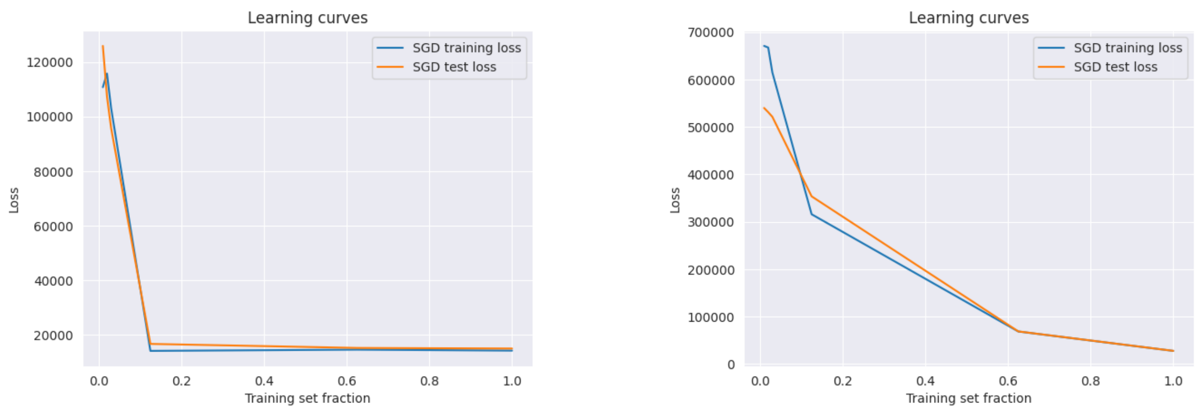
`Test loss [L2]:` 14496.2205

`Test loss [L1]:` 14876.4765

`Test loss [SGD]:` 14886.0377

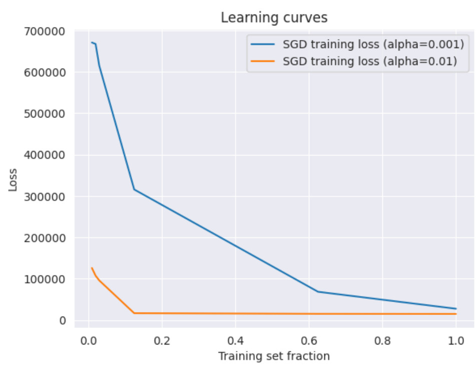
L2 króluje.

## 8 Strata na zbiorze testowym względem rozmiaru zbioru treningowego




(a) $\alpha = 0.01$ (b) $\alpha = 0.1$

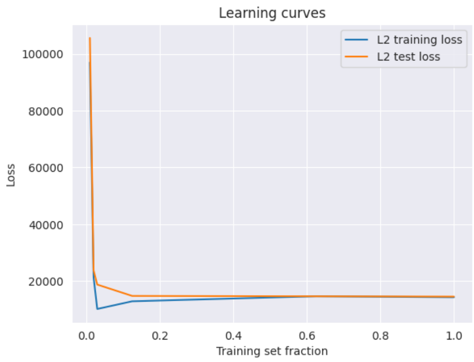
Rysunek 8: SGD




Rysunek 9: Zbieżność SGD

Kolejny wykres wyjątkowo mi się podoba.

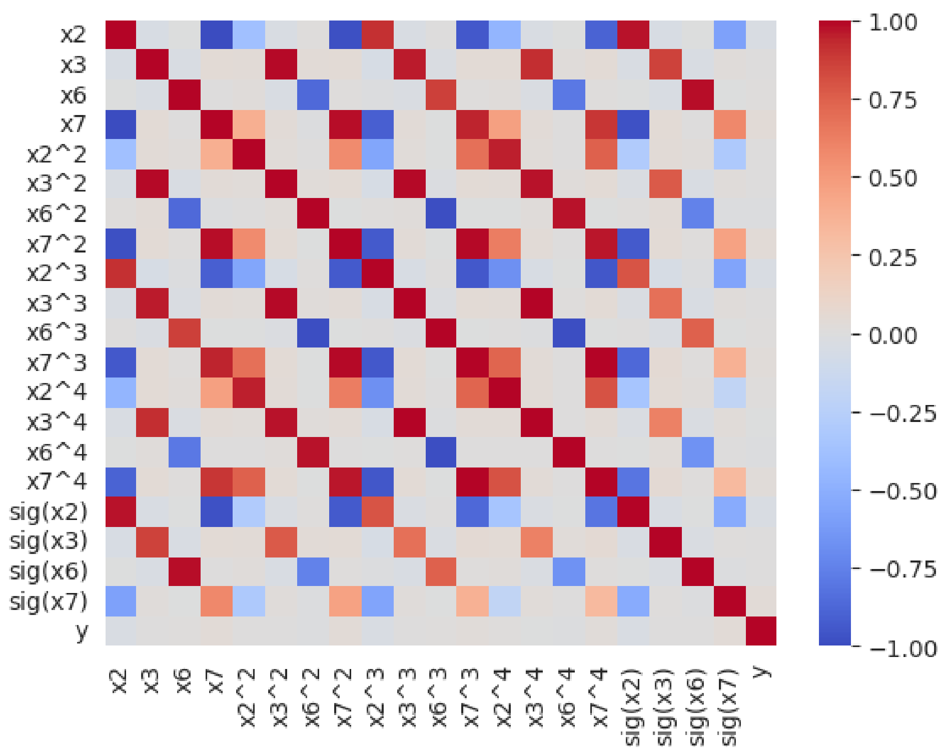
Pokazuje, że L2 najpierw się dopasowała a potem zaczęła regularyzację.




Rysunek 10: Regularyzacja L2

# 9 Dodatkowe funkcje bazowe

Niestety w tej sekcji nie mogę się pochwalić wielkimi osiągnięciami. Zostawiłem $x_1, x_4, x_5$ w spokoju, a resztę zmiennych rozszerzyłem o wielomiany $x^2, x^3, x^4$ oraz $\sigma(x)$.




Rysunek 11: Korelacja rozszerzonych zmiennych

Macierz korelacji nie wróżyła nic ciekawego.

Puściłem na tym SGD z $\alpha = 0.01$ i `batch_size` = 64. Ku mojemu zdziwieniu otrzymałem najlepszy wynik spośród wszystkich poprzednich.

`Test loss [Augmented dataset SGD]:` 13369.6066

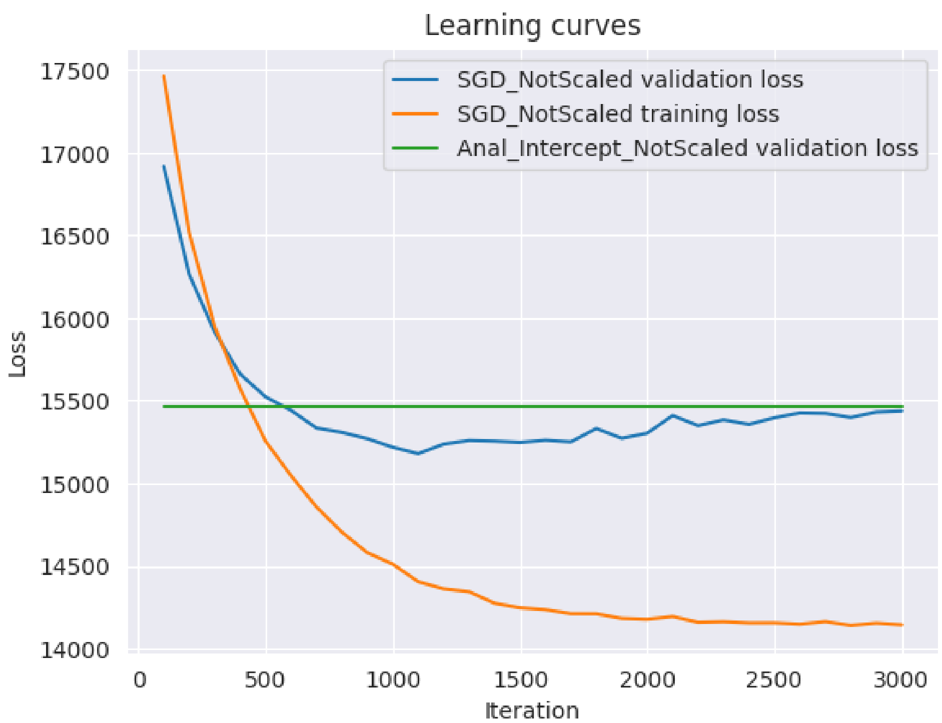
## 10 Podsumowanie projektu

Może nie otrzymałem satysfakcjonujących wyników na zbiorze testowym, ale zakochałem się bez pamięci w SGD i L2. To jest niesamowite jak dorzucenie odrobiny losowości poprawia wyniki.

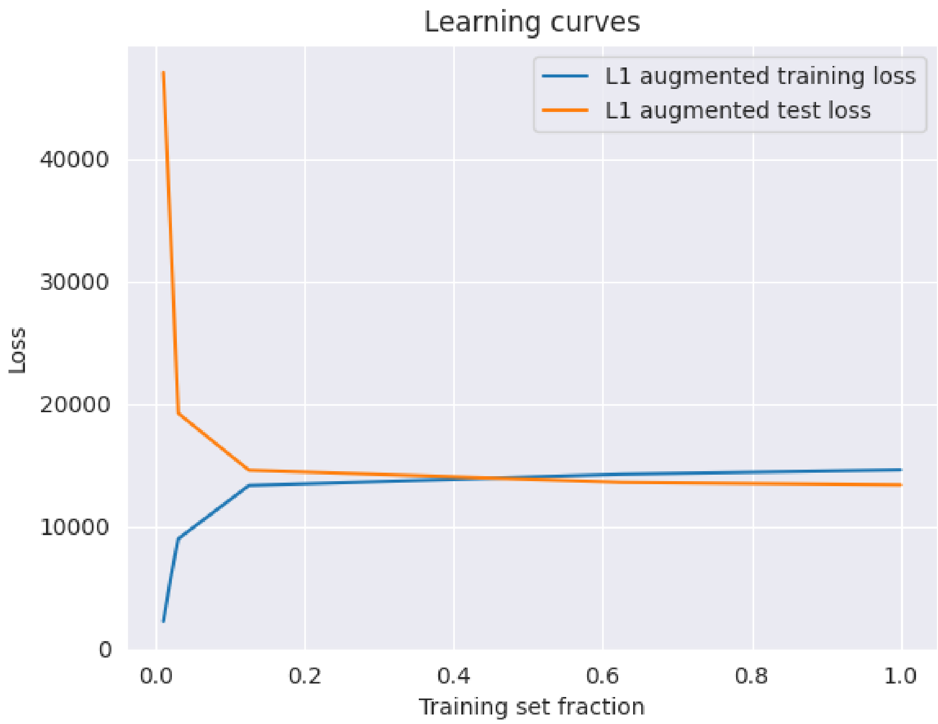
**PS**

Proszę się nie gniewać za obrazek ze strony tytułowej, świetnie się bawiłem przy tym projekcie.

## 11 BONUS




Rysunek 12: SGD schodzi poniżej analitycznej straty




Rysunek 13: L1 z $\lambda = 256$