Miniprojekt 1

Cały miniprojekt napisałem w pythonie wykorzystując biblioteki numpy oraz matplotlib do obliczeń i narysowania wykresów.

Początkowo stworzyłem moduł: **division_and_scaling.py**, w którym zaimplementowałem podział danych w losowy sposób na zbiory: treningowy(60%), walidacyjny(20%) oraz testowy(20%), a także możliwość wyboru skalowania danych metodami: **min-max, min-max2, standaryzacja**, a następnie podzielone i przeskalowane dane zapisywałem do odpowiednich plików odpowiadających zbiorom.

Następnie stworzyłem moduł **prepare_data.py**, w którym wczytuje wcześniej zapisane do plików zbiory, a potem przekształcam dane za pomocą różnych funkcji bazowych: **wielomiany, funkcje gaussowskie, sigmoid, sinusy,** a także **funkcji usuwających niektóre argumenty.** Na koniec uzupełniam stworzoną macierz kolumną wypełnioną jedynkami.

Stworzyłem również 3 foldery odpowiadające za odpowiednie regularyzacje:

- grzbietową
- lasso
- sieć elastyczną

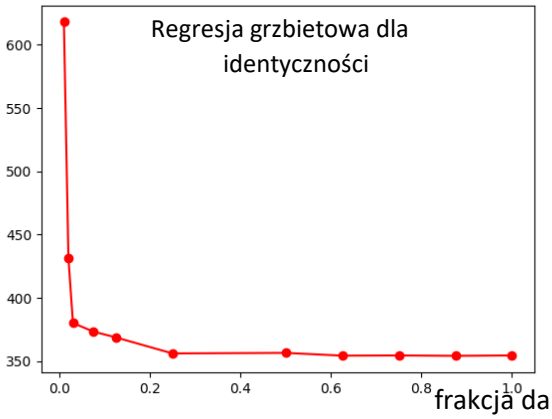
W folderze regresji grzbietowej zaimplementowałem **algorytm spadku w wzdłuż gradientu w wersji mini batch** z możliwością dostosowywania rozmiaru batcha, a także proces znajdowania najlepszej lambdy dla danego zbioru walidacyjnego i mechanizm szukający thety na podstawie zbioru treningowego minimalizując koszt na zbiorze testowym.

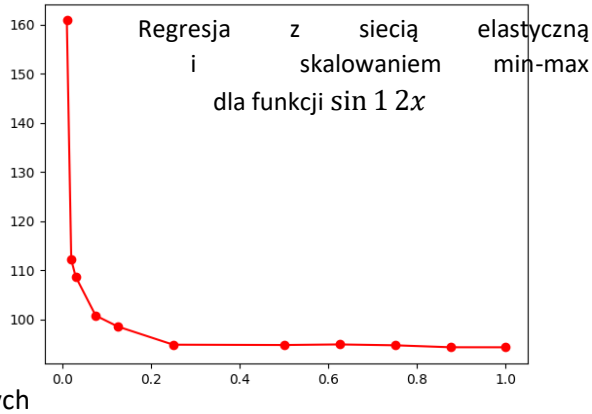
W folderach regresji lasso i sieci elastycznej zaimplementowałem **algorytm spadku w wzdłuż współrzędnych** odpowiedni dla danej regularyzacji, a także podobnie jak w przypadku regresji grzbietowej mechanizm szukania lambdy (i lambdy2) i thety minimalizującej koszt na zbiorze testowym.

W folderze resources stworzyłem także foldery odpowiadającym powyższym regularyzacjom, w których zapisywałem wyniki / metaparametry obliczone w procesie uczenia się / walidacji. Znajdują się tam też otrzymane wykresy zawierające uśrednione krzywe uczenia na różnych podziałach na zbiory.

Niezależnie od wybranego modelu, skalowania i ostatecznego kosztu otrzymanego na zbiorze testowym można zaobserwować, że największa poprawa kosztu w zależności od frakcji danych następuje na przedziale [0-0.125], po czym wykres jest praktycznie stały - tzn. dalej już nie ma nagłych, drastycznych spadków, polepszeń.

$\sqrt{koszt}$

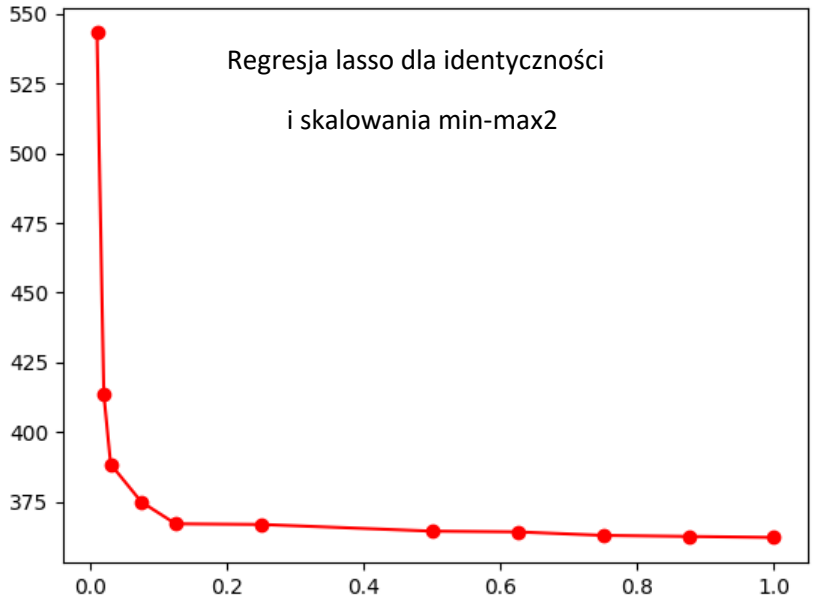




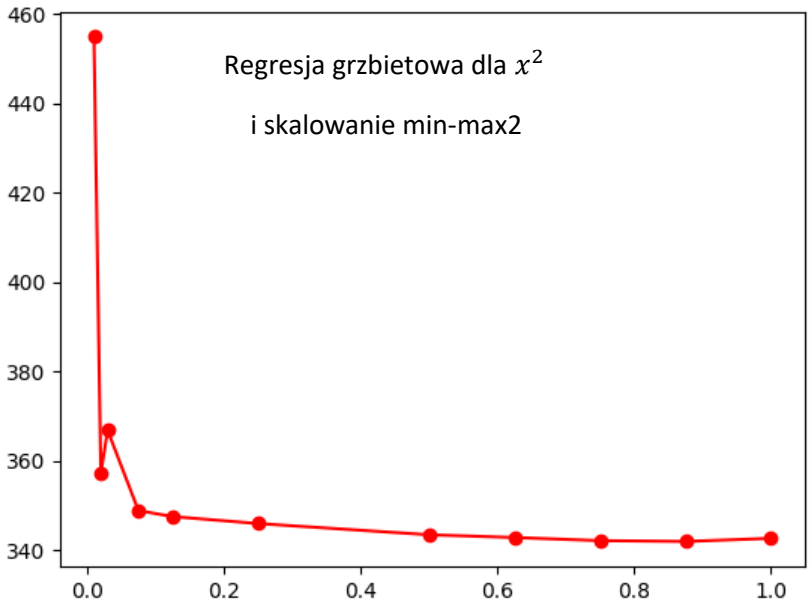
frakcja danych

Przeanalizujemy teraz wpływ funkcji bazowych na otrzymany koszt w zależności od skalowania danych i zastosowanej regularyzacji:

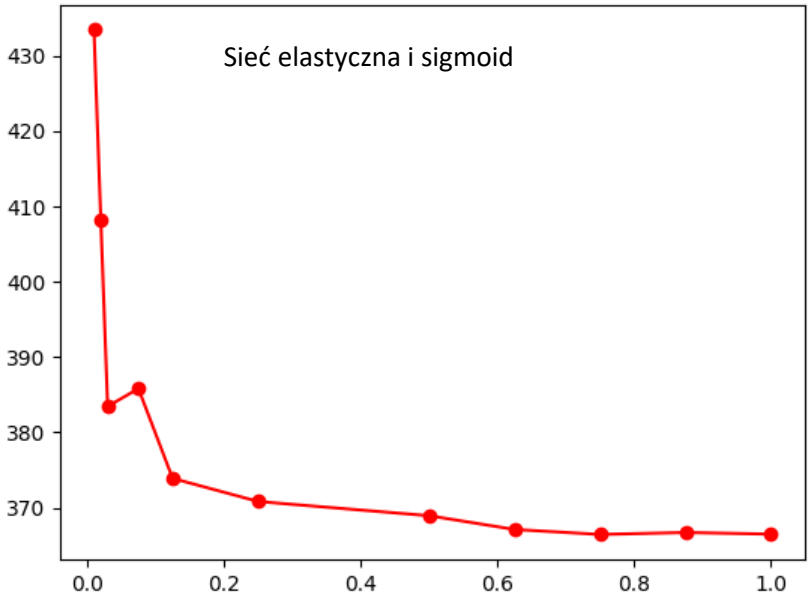
1. Identyczność
   Wartość kosztu dla tej funkcji oscylowała granicy 125 000 – 130 000 i była dosyć stabilna dla różnych zbiorów i skalowań. W wynikowej θ dla regresji lasso można zaobserwować w wielu przypadkach zerowanie 3 i 4 argumentu, co może świadczyć o małym wpływie tych danych na ostateczny wynik - np. θ = [457.75, -68.94, 25.60, 0, 0, -5.5, -22.84]



2. Wielomiany
   Zastosowanie różnych kombinacji mnożenia kolumn, a także funkcji f(x) = x^2, x^3, itd. nie spowodowało znaczącego spadku kosztu dla naszych danych. Koszt był zbliżony do kosztów identyczności. Jedynie dla funkcji f(x) =x^2 i skalowanie min-max2 średni koszt był trochę niższy i wynosił 117 000 - 125 000

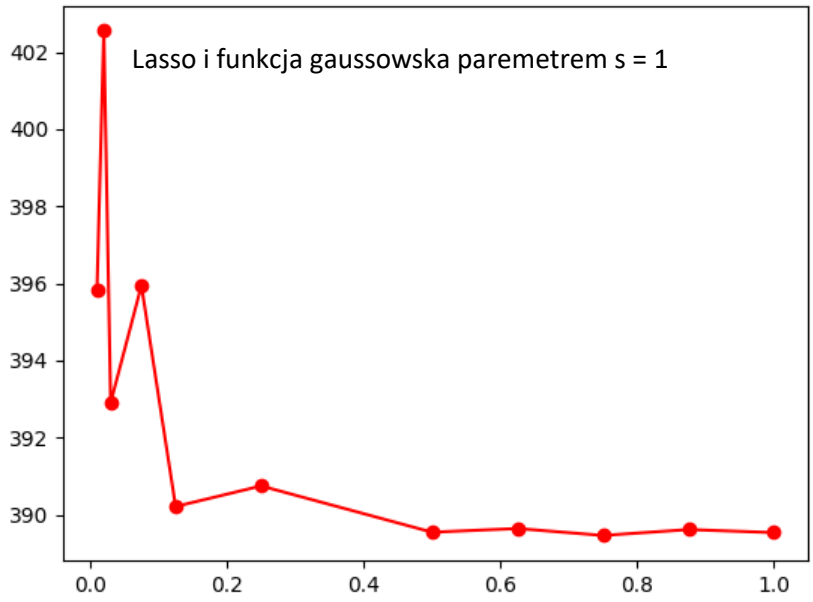


3. Sigmoid
   Średnia wartość kosztu dla tej funkcji nie była lepsza niż dla poprzednich modeli, a nawet była wyższa i wynosiła ponad 130 000 (130 000 – 135 000). Ponadto współczynniki w wektorze θ w wielu przypadkach były większe niż w poprzednich modelach i wynosiły np. w sieci elastycznej θ = [1531.21, -2948.17, 221.14, 12.16, 2345.62, -32.28, -1670.27]



4. Funkcje gaussowskie

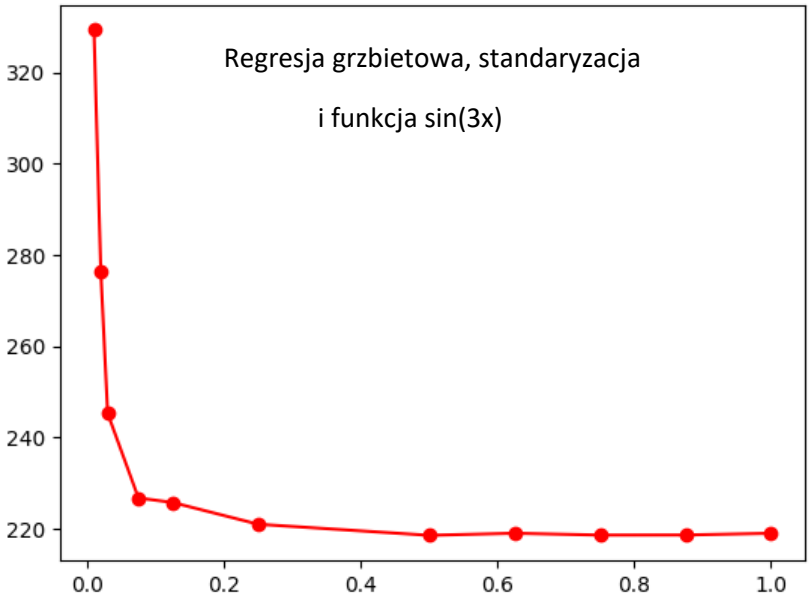
Funkcje gaussowskie dla każdej regularyzacji przeanalizowałem dla wielu różnych metaparametrów s. Tzn. przed przystąpieniem do szczegółowego szukania parametrów regularyzacji badałem funkcję kosztu dla **s** z przedziału [$\frac{1}{50}$, 50]. Koszt tych funkcji był już jednak wielokrotnie równy ponad 150 000 albo wektor θ miał wyzerowaną większość współrzędnych - dla sieci elastycznej θ = [324.83, 0, 0, 0, 0, 0, 172.93]. Ponadto często zaobserwowałem nienaturalny wygląd wykresów kosztu od frakcji danych treningowych – tzn. wykres potrafił mieć wzniesień i spadków, a wynik dla frakcji 1.0 był zbliżony do wyniku 0.01. Wynika z tego, że funkcje gaussowskie nie są dobrą funkcją bazową dla naszych danych.



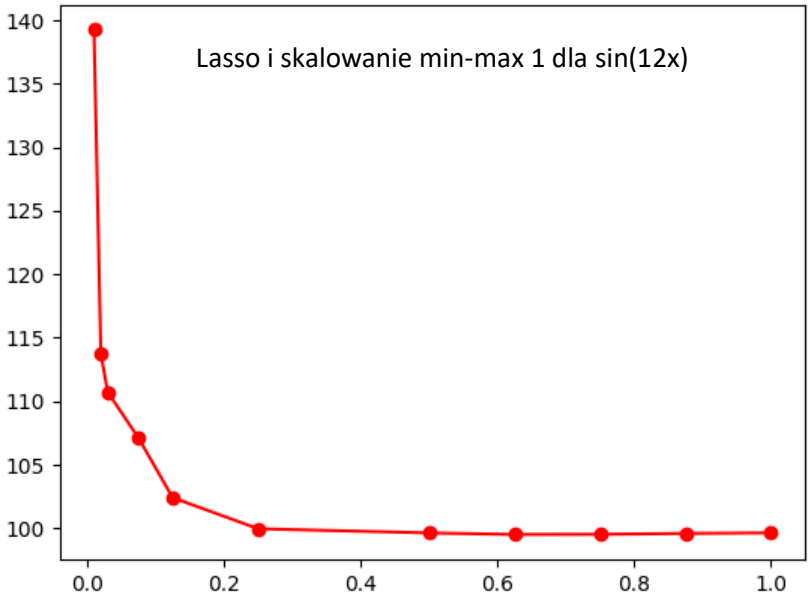
5. Sinusy – f(x) = sin(αx)

Podobnie jak w przypadku funkcji gaussowskich pracę rozpocząłem od wyznaczenia najlepszych parametrów α dla różnych skalowań i regularyzacji. Po sprawdzeniu pierwszych wyników funkcji kosztu od razu zobaczyłem, że sinusy są wreszcie funkcją bazową, która znacząco poprawia i przebija poprzednie wyniki. Dla standaryzacji, regularyzacji grzbietowej i funkcji sin(3x) otrzymałem średni koszt w granicach 45 000 – 47 000 i

θ = [443.81, -376.18, -4.54, -7.55, -6.22, -3.80, 166.04], po sprawdzeniu regresji lasso otrzymałem podobny koszt jednak wektor θ był w większości wyzerowany - np. θ = [442.06, -374.82, 0, 0, 0, 0, 157.92] a optymalny parametr λ średnio wynosił ponad 0 podczas gdy w innych modelach był znacząco poniżej 0 - np. λ = 6.16



Po znalezieniu funkcji sin(3x) dla standaryzacji zacząłem badać sinusy dla innych skalowań. Wówczas dla skalowania min-max 1 zauważyłem, że dla funkcji w okolicach sin10x, sin11x, sinus13x otrzymuję bardzo podobne wyniki co wcześniej dla sin3x albo lekko gorsze – granicach 55 000 – 70 000. Jednakże nie był to najlepszy wynik jaki otrzymałem, ponieważ dla funkcji f(x) = sin12x średni koszt spadł poniżej 10 000. Przykładowe średnie dla regresji grzbietowej: `9050.09, 10288.40,` lasso: `8179.36, 9458.82,` sieć elastyczna: `9621.00, 8891.61.` Współczynniki w θ dla argumentów 2,3,4 były we wszystkich regularyzacjach zbliżone do 0, a w lasso bardzo często po prostu równe - θ = [`442.90, 462.81, 0, 0, 0, 11.23, 189.04`] przy średniej λ = `3.87.` Potem gdy zbadałem dokładniej standaryzację to okolicach sin3x również znalazłem funkcję dające średni koszt ok. 10 000. Wynika z tego, że sinusy są najlepszą funkcją bazową dla naszego modelu.



6. Funkcje usuwające argumenty

Z ciekawości chciałem sprawdzić co stanie się z kosztem liczonym dla identyczności, gdy nie dostaniemy wszystkich argumentów. Okazało się, że usunięcie 1 argumentu zwiększyło koszt funkcji porównując z funkcją kosztu dla identyczności do ponad 140 000, a natomiast usunięcie 3 i 4 argumentu nie spowodowało zmiany funkcji kosztu, a nawet ją czasami polepszyło o 2000 – 3000