Figure 1: Logo Politechniki

Politechnika Krakowska

Wydział Inżynierii Elektrycznej i Komputerowej

Studia Niestacjonarne, Informatyka Sprawozdanie z przedmiotu:

Sztuczna Inteligencja

Prowadzący: mgr. inż. Kazimierz Kiełkowicz

Temat Projektu:

Klasyfikacja - Przewidywanie stabilności systemu symulującego zmiany klimatyczne (Climate Model Simulation Crashes) Wykonali:

Jakub Gwiazda

Wojciech Windak

Wstęp:

Celem naszego projektu było wykonanie klasyfikatora dla określonych danych systemu symulującego zmiany klimatyczne. Symulacje klimatu są bardzo złożonym

zadaniem, obecne projekty pozwalające na częściowe modelowanie zmian klimatycznych składają się z wielu modułów, przeważnie zbierających dane niezależnie od siebie. W naszym projekcie wykorzystamy dane zebrane podczas symulacji Parallel Ocean Program (POP2), komponentu modelu klimatycznego Community Climate System Model (CCSM4).

Wśród 540 wykonanych symulacji, 46 zakończyło się niepowodzeniem ze względu na wartości liczbowe, co daje 8,5% całości.

Każda symulacja posiadała 18 parametrów(kolumny 3-20), które w różnym stopniu wpływały na możliwe zawieszenia działania programu. Data set zawierał dodatkowo 3 kolumny, numer grupy testowej(1), id symulacji(2) oraz wynik symulacji(21, porażka/sukces).

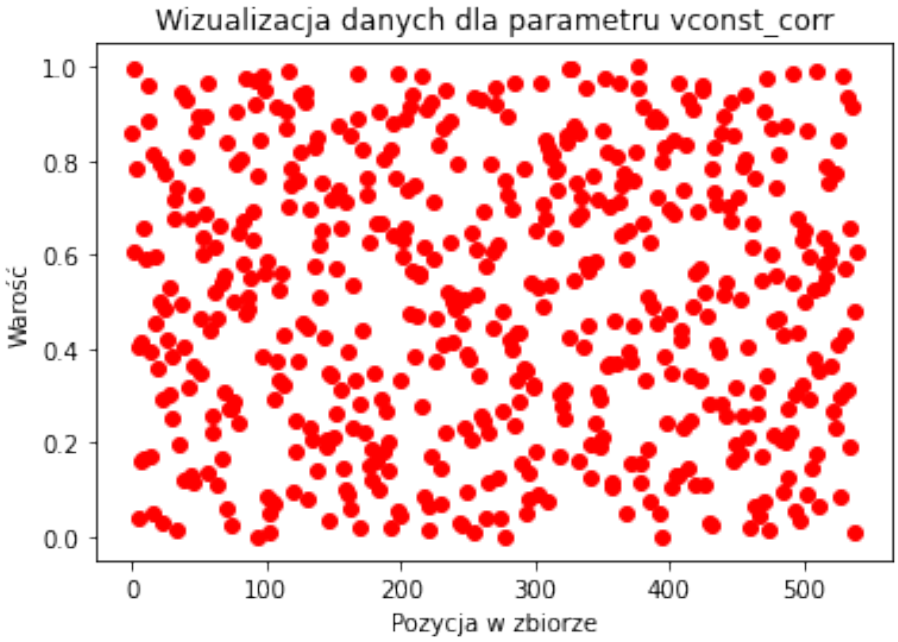
Celem jest wykorzystanie klasyfikacji w celu przewidywania powodzenia symulacji na podstawie wprowadzonych do niej wartości parametrów, a także odnalezienie parametrów, które w głównym stopniu wpływają na możliwe niepowodzenia.

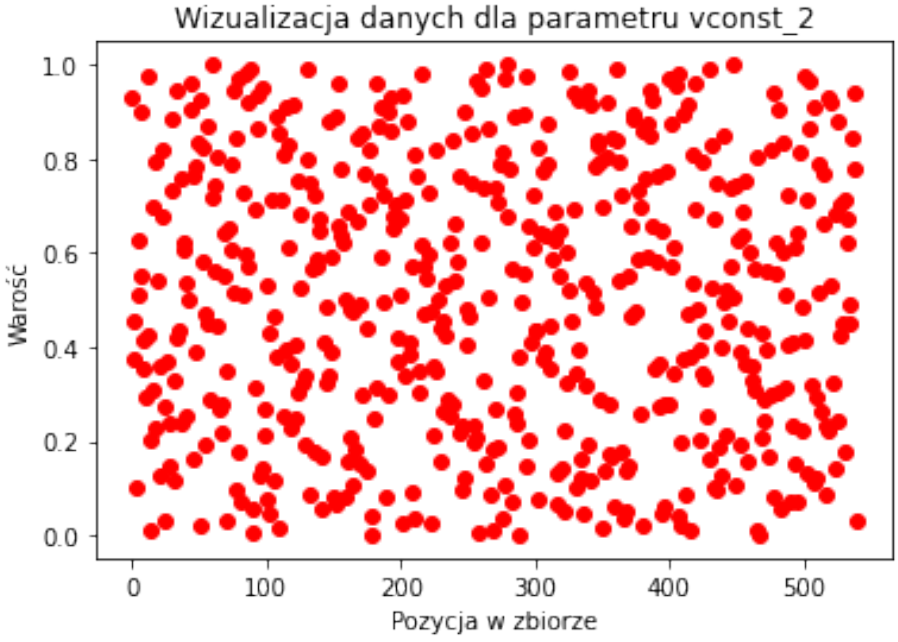
Przedstawienie danych: Nasz data set zawierał 540 wierszy z podziałem na 21 kolumn. Kolumny 1 i 2 zawierały kolejno numer grupy testowej(1-3) i numer kolejnej symulacji w niej(1-180). Kolumny 3-20 zawierają parametry symulacji POP2 o wartościach rzeczywistych w zakresie 0-1. Kolumna 21 natomiast zawiera wynik symulacji(porażka/sukces). Parametry w kolumnach 3-8 wykorzystywane były do uchwycenia poziomego mieszania pędu o przestrzennie anizotropowej lepkości. Parametry 9–11 są wykorzystywane do poziomego mieszania znaczników poprzez izopikalny transport indukowany wirami. Parametry 12–14 są wykorzystywane w niedawno opracowanych schematach do symulacji wirów submezoskalowych i mieszanych warstw oraz mieszania pływów głębinowych. Parametry 15–20 są używane do wyznaczania konwekcji pionowej i mieszania pionowego za pomocą schematu parametryzacji profilu K (KPP).

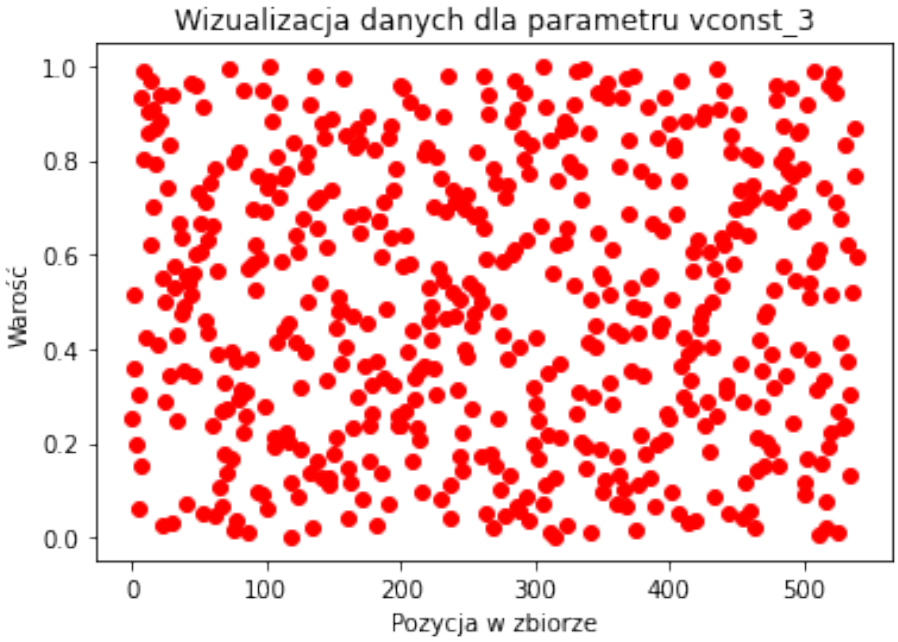
Dane przed normalizacją

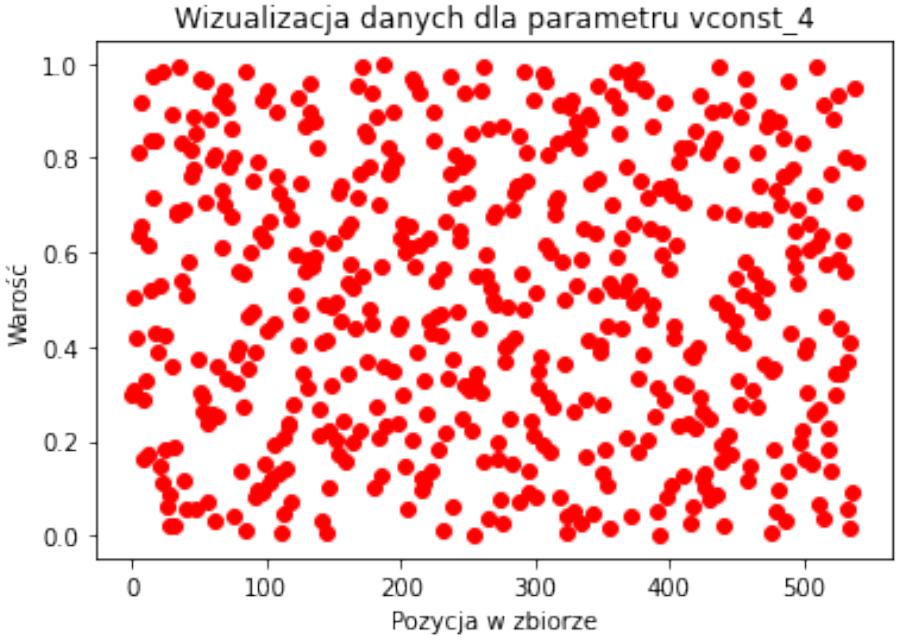
Wykonaliśmy wizualizację danych w celu zobrazowania ich w odpowiedni sposób dla każdego parametru.

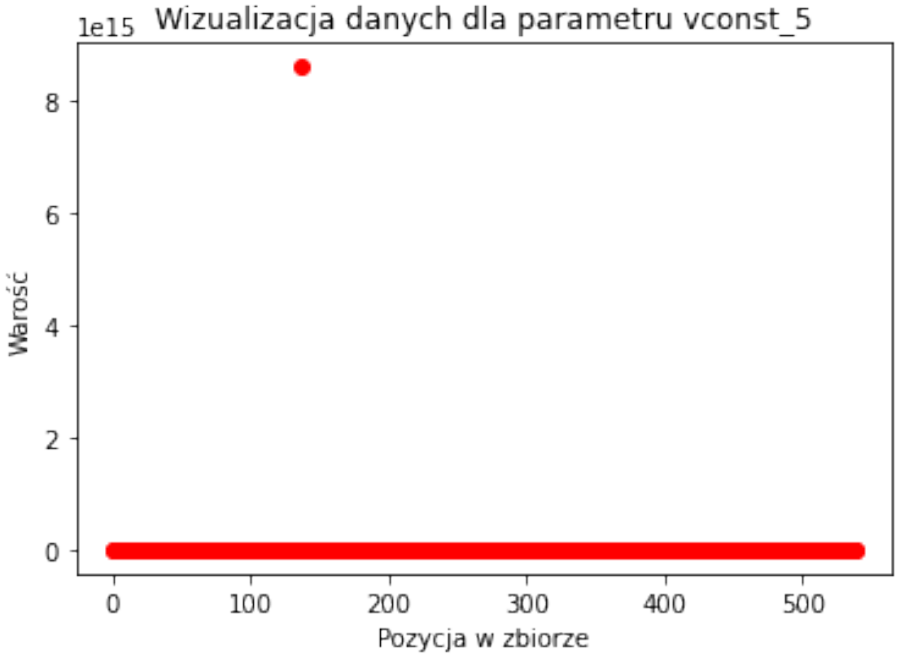
	Study	Run	vconst_corr	vconst_2	vconst_3
count	540.000000	540.000000	540.000000	540.000000	540.000000
mean	2.000000	90.500000	0.500026	0.500097	0.500027
std	0.817254	52.008901	0.288939	0.288922	0.289067
min	1.000000	1.000000	0.000414	0.001922	0.001181
25%	1.000000	45.750000	0.249650	0.251597	0.251540
50%	2.000000	90.500000	0.499998	0.499595	0.500104
75%	3.000000	135.250000	0.750042	0.750011	0.749180
max	3.000000	180.000000	0.999194	0.998815	0.998263
	vconst_4	vconst_5	vconst_7	ah_corr	ah_bolus
count	540.000000	5.400000e+02	540.000000	540.000000	540.000000
mean	0.500119	1.589598e+13	0.499913	0.500059	0.500076
std	0.288993	3.693892e+14	0.288852	0.289010	0.288909
min	0.001972	1.889182e-03	0.000476	0.004590	0.000296
25%	0.250158	2.534061e-01	0.251325	0.253048	0.250402
50%	0.500456	5.019133e-01	0.499174	0.499070	0.500074
75%	0.750348	7.513031e-01	0.748166	0.750109	0.749091
max	0.997673	8.583829e+15	0.997142	0.998930	0.998506
	efficiency_factor	tidal_mix_max	vertical_decay_scale	convect_corr	bckgrnd_vdc1
count	540.000000	540.000000	540.000000	540.000000	540.000000
mean	0.500111	0.499984	0.500032	0.499933	0.499944
std	0.288966	0.289127	0.289014	0.288822	0.288949
min	0.002015	0.000419	0.001188	0.001312	0.002509
25%	0.250758	0.251676	0.249669	0.249988	0.249586
50%	0.500393	0.500322	0.500151	0.500625	0.499080
75%	0.749447	0.749346	0.749164	0.749569	0.750012
max	0.999536	0.999942	0.997718	0.997518	0.999795
	bckgrnd_vdc_ban	bckgrnd_vdc_eq	bckgrnd_vdc_psim	Prandtl	outcome
count	540.000000	5.400000e+02	540.000000	540.000000	540.000000
mean	0.499946	1.649366e+13	0.500020	0.500021	0.914815
std	0.288923	3.832780e+14	0.288936	0.289013	0.279416
min	0.000732	2.748962e-03	0.000219	0.000263	0.000000
25%	0.249974	2.528092e-01	0.252739	0.249723	1.000000
50%	0.499959	5.023113e-01	0.498955	0.499431	1.000000
75%	0.747978	7.510987e-01	0.748539	0.749792	1.000000
max	0.999155	8.906575e+15	0.999306	0.999655	1.000000

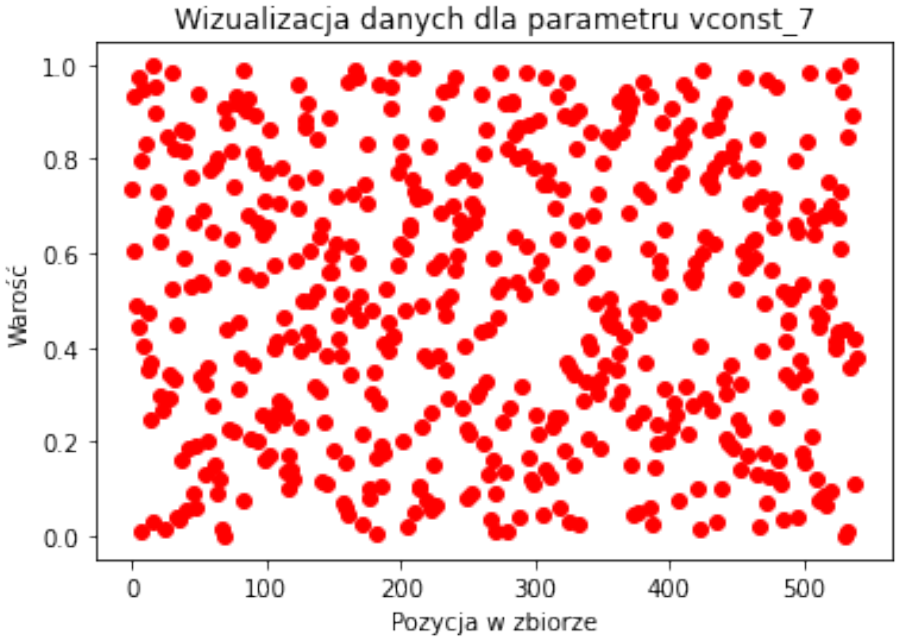

Wizualizacja danych dla parametru vconst_corr
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

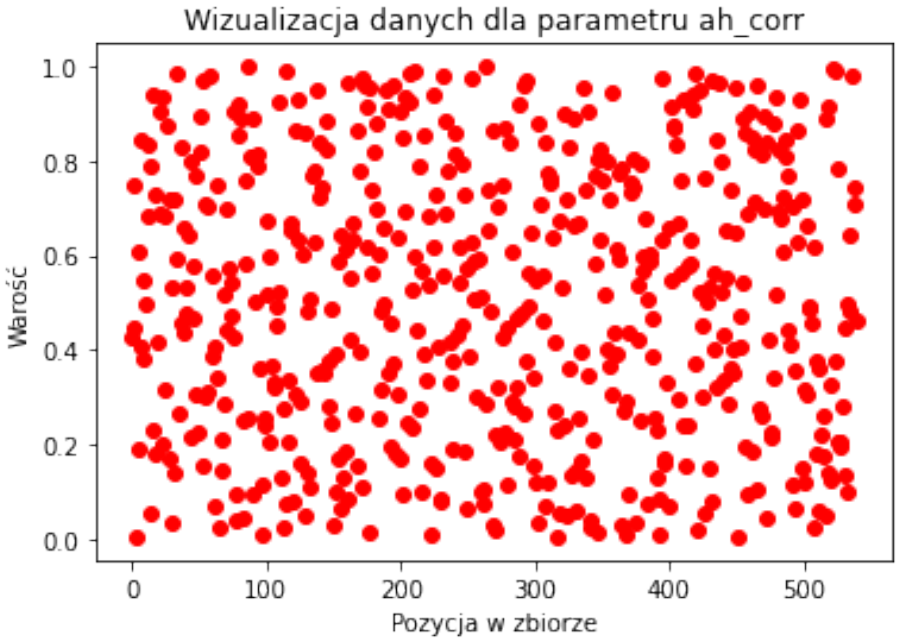

Wizualizacja danych dla parametru vconst_2
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

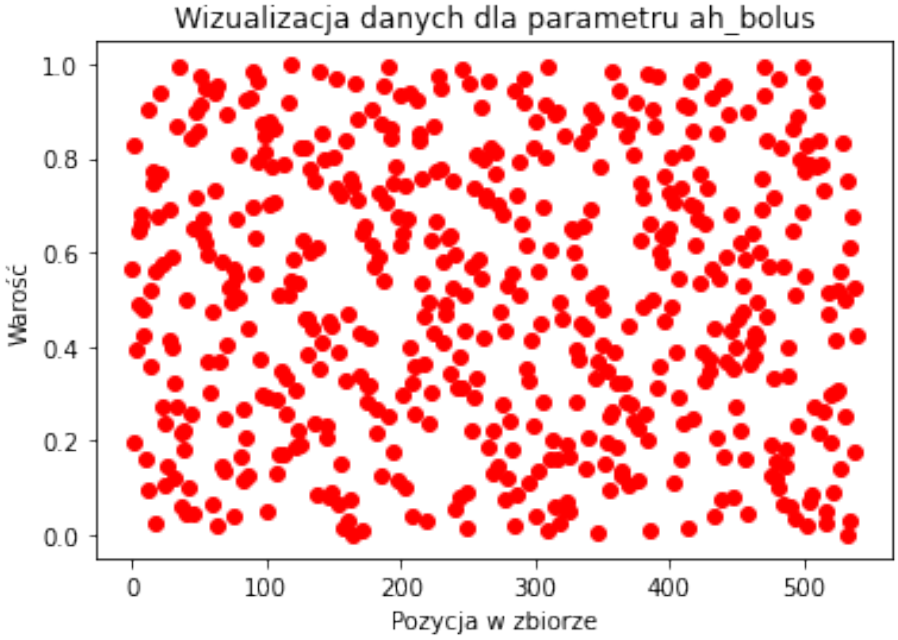

Wizualizacja danych dla parametru vconst_3
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

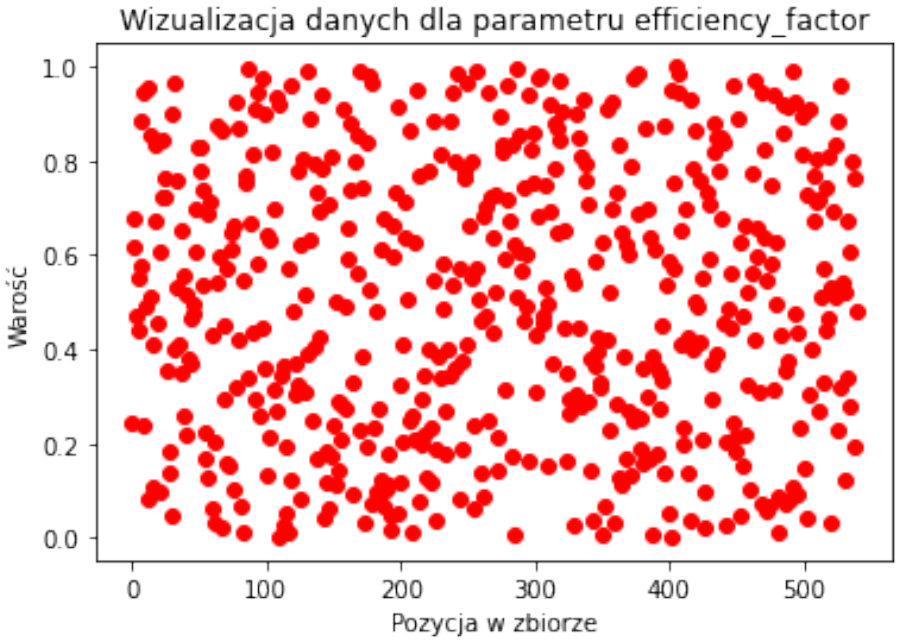

Wizualizacja danych dla parametru vconst_4
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

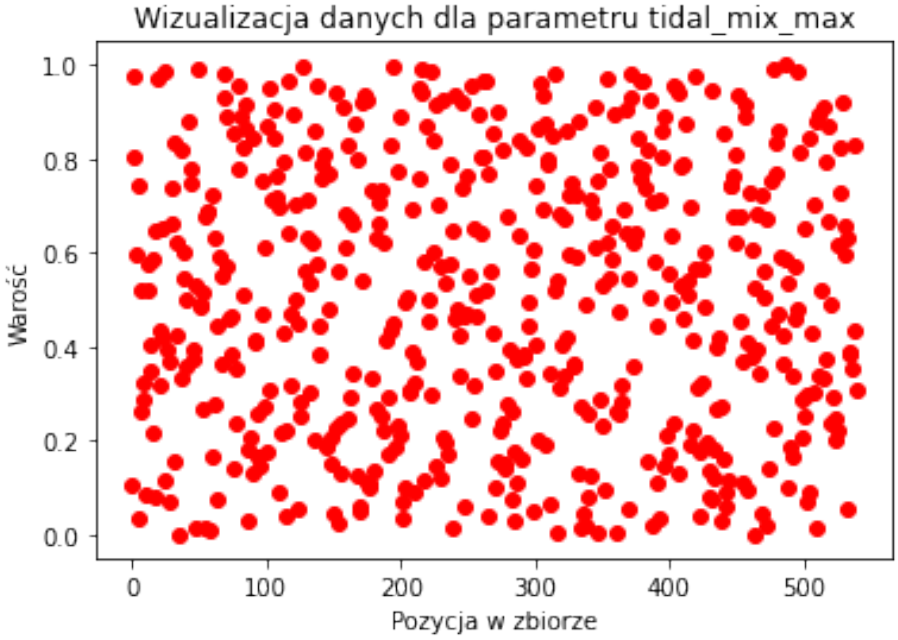

Wizualizacja danych dla parametru vconst_5
1e15
8
6
4
2
0
Warość
0
100
200
300
400
500
Pozycja w zbiorze

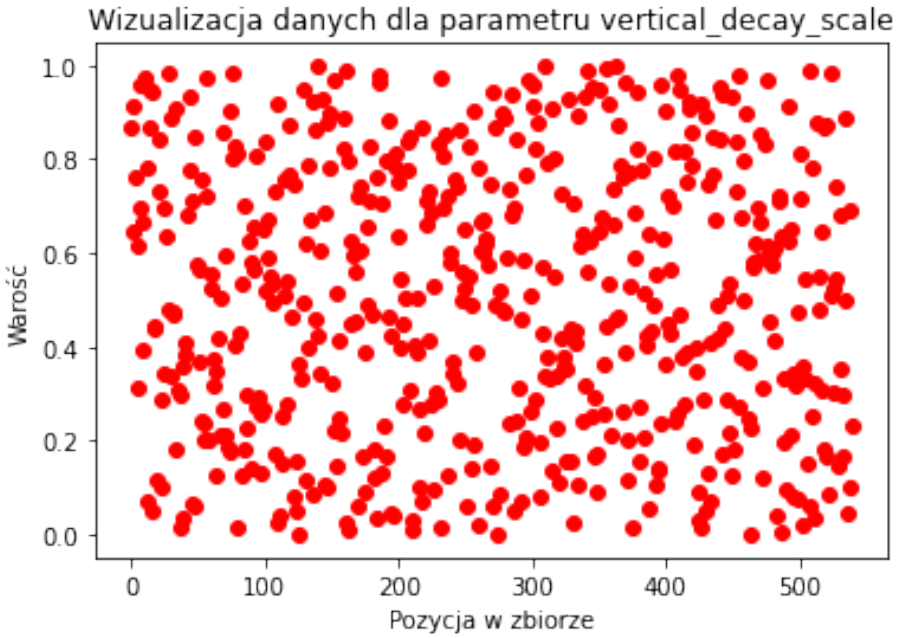

Wizualizacja danych dla parametru vconst_7
1.0
0.8
0.6
0.4
0.2
0.0
Warość
0
100
200
300
400
500
Pozycja w zbiorze

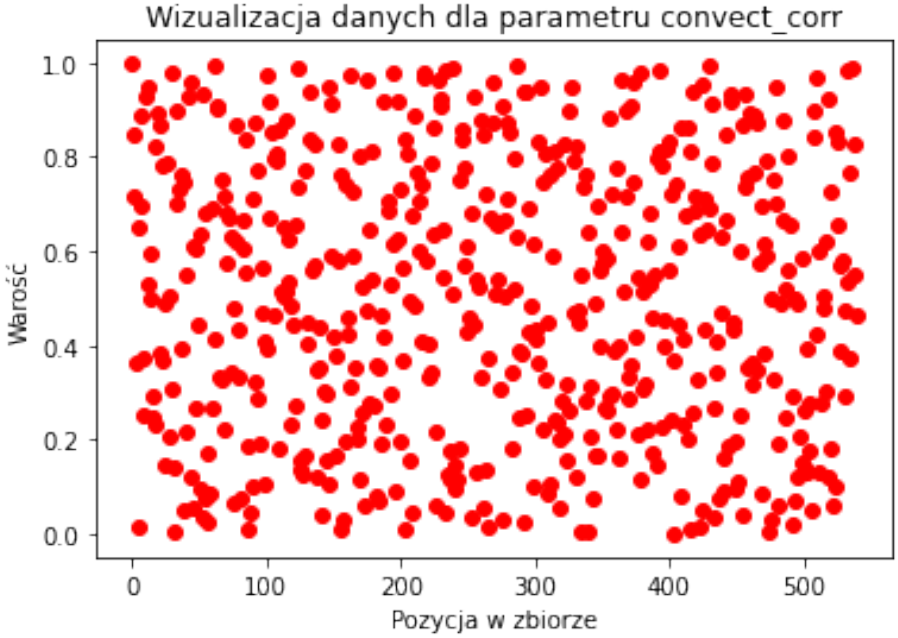

Wizualizacja danych dla parametru ah_corr
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

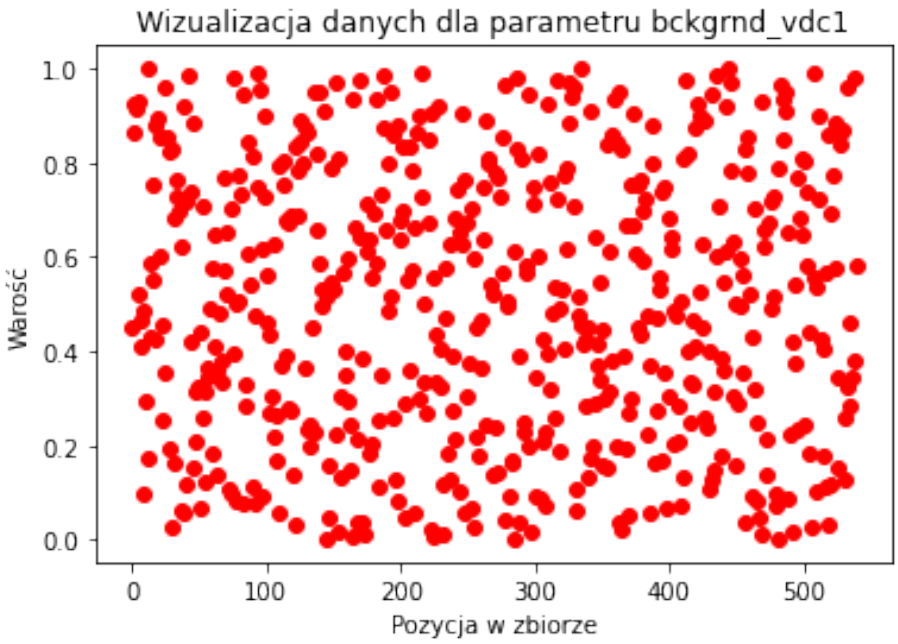

Wizualizacja danych dla parametru ah_bolus
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

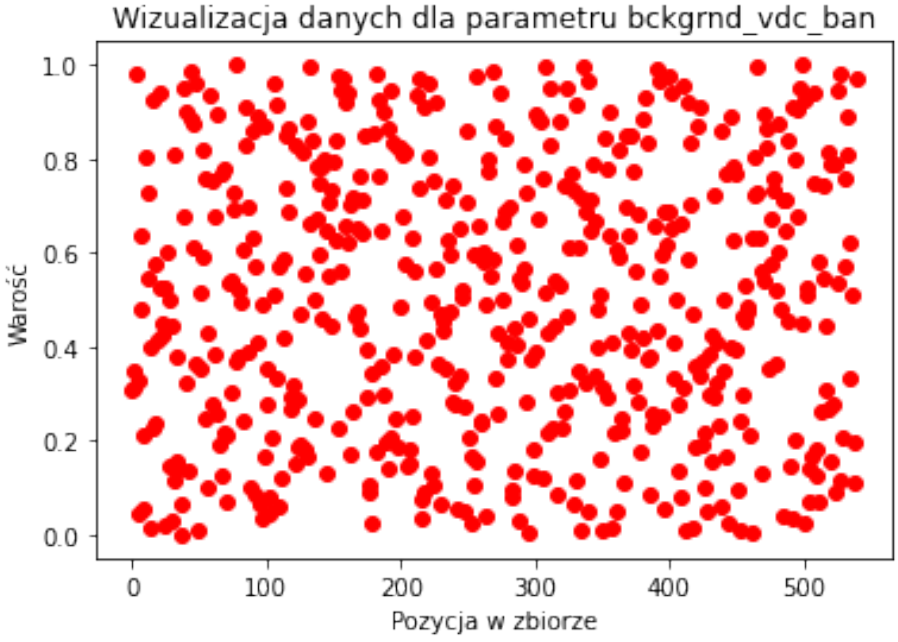

Wizualizacja danych dla parametru efficiency_factor
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

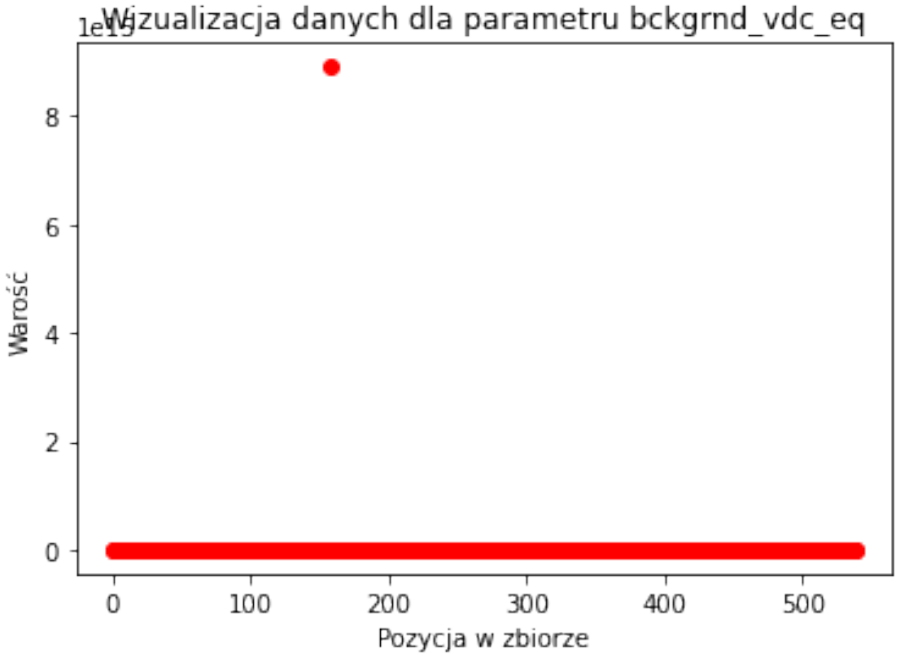

Wizualizacja danych dla parametru tidal_mix_max
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

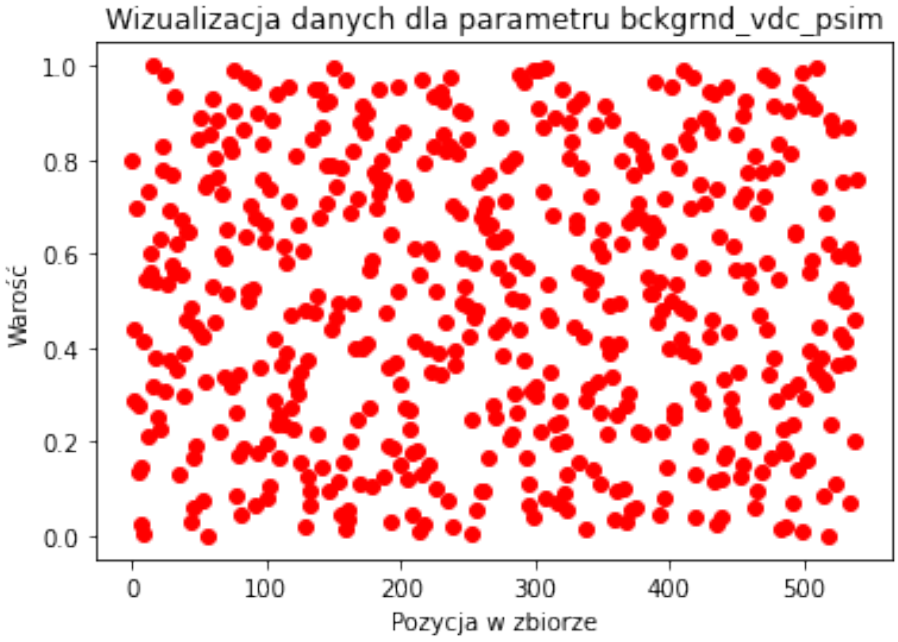

Wizualizacja danych dla parametru vertical_decay_scale
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500

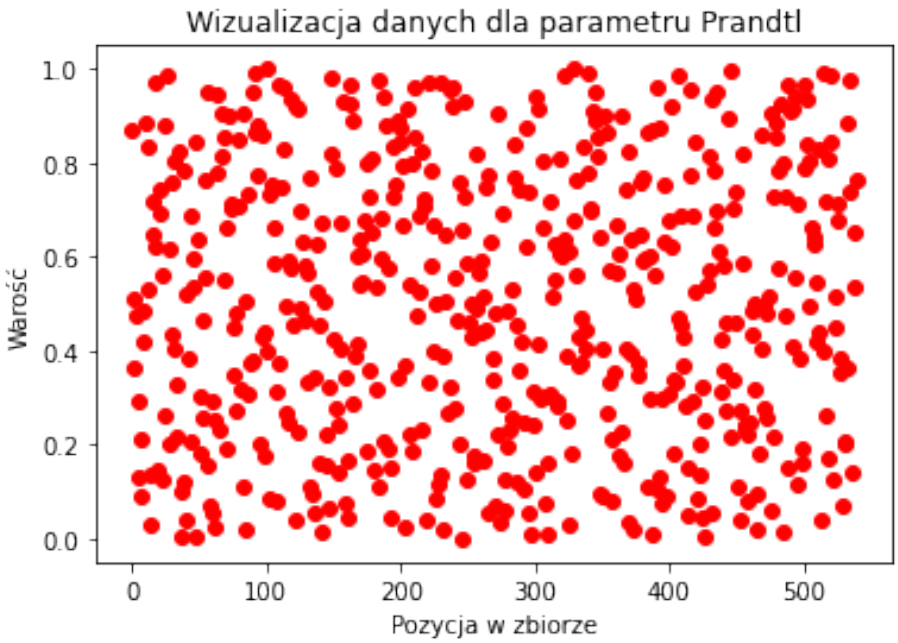

Wizualizacja danych dla parametru convect_corr
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500


Wizualizacja danych dla parametru bckgrnd_vdc1
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500


Wizualizacja danych dla parametru bckgrnd_vdc_ban
Warość
Pozycja w zbiorze
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500


Wizualizacja danych dla parametru bckgrnd_vdc_eq
1e15
8
6
4
2
0
Warość
0
100
200
300
400
500
Pozycja w zbiorze


Wizualizacja danych dla parametru bckgrnd_vdc_psim
1.0
0.8
0.6
0.4
0.2
0.0
Warość
0
100
200
300
400
500
Pozycja w zbiorze



A także ostatni parametr outcome wyznaczający które symulacje zakończone zostały sukcesem bądź porażką.

Analiza danych Przedstawione wizualizacje pozwoliły nam potwierdzić, które dane zawierają się w przedziale wartości rzeczywistych 0-1 i są losowo rozłożone na pełnej skali. W dwóch przypadkach natrafiliśmy na anomalie, które mogłyby wpłynąć na dokładność naszego klasyfikatora. Dla parametrów z kolumn nr. 7 (vconst_5) i nr. 17 (bckgrnd_vdc_eq) znaleźliśmy nieliczne wartości, rzędy wielkości odbiegające od reszty tzw. outlinery. Dla zwiększenia dokładności dla większości przypadków, zdecydowaliśmy się na pozbycie się tych wartości ze zbioru, gdyż ich brak nie wpłynie na poprawność klasyfikacji, a pozwoli znacznie zwiększyć dokładność dla ogólnych przypadków.

Zgodnie z analizą z poprzedniego punktu kolumny “bckgrnd_vdc_eq” i “vconst_5” zawierają po jednym wierszu, który znacznie wykracza poza trend pozostałych danych oraz wykracza poza założenie podane na stronie zbioru danych które mówi że dane w kolumnach 3-20 są wyskalowane od wartości zero do jeden.

Z-score Z-score (nazywany również standardowym wynikiem) - jest to wyobrażenie o tym, jak daleko dany parametr jest od średniej wartości punktu danych. Bardziej technicznie jest to miara tego, ile standardowych odchyleń poniżej lub powyżej danej populacji oznacza wynik surowy.

Wynik z-score można umieścić na krzywej rozkładu normalnego. Punktacja z

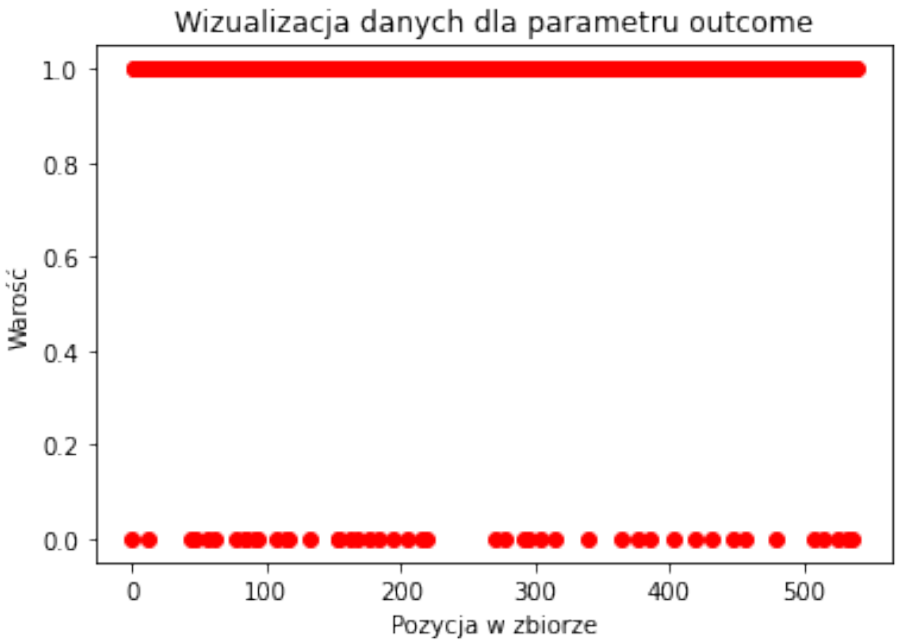


Figure 2: outcome

przechodzi od -5 odchyleń standardowych (które spadłyby do skrajnej lewej strony krzywej rozkładu normalnego) do +5 odchyleń standardowych (które spadłyby do skrajnej prawej strony krzywej rozkładu normalnego). Aby wykorzystać wynik z-score, musimy znać średniąμ i populacyjne odchylenie standardowe σ.

Podstawowy wzór wyniku z dla próbki jest następujący:

$z = (x \breve{} \mu)/\sigma$

Jeśli masz wiele próbek i chcesz opisać odchylenie standardowe tych próbek oznacza (błąd standardowy), użyjesz tej formuły z-score:

$z = (x \breve{} \mu)/(\sigma/\sqrt{n})$

Wykorzystanie z-score do odrzucenia outsiderów W sekcji 2.1 w kodzie zastosowana została implementacja z-score z biblioteki scipy. Dla całości zmiennej df wykonywana jest operacja wyliczenia z-score względem każdego z wierszy osobno. Następnie każdy z wierszy jest sprawdzamy czy jego wynik wykracza poza wartość 5 zarówno dodatnią jak i ujemną. Jeżeli tak się stanie to wiersz ten jest wyrzucany z tabeli, jako zawierający dane nie pasujące do trendu.

Wyłączenie kilku outlinerów pozwoli faktycznie uzyskać informacje z tego parametru bez szumu pochodzącego z pojedynczych absolutnie skrajnych przypadków.

Po wyrzuceniu ich ze zbioru uzyskaliśmy nowe Wizualizacje dla parametrów vconst_5 i bckgrnd_vdc_eq:

Figure 3: vconst_5

Na podstawie powyższych wizualizacji możemy wywnioskować, że wszyscy outlinerzy zostali usunięci z problematycznych kolumn.

W celu weryfikacji całości danych wygenerowaliśmy jeszcze raz tabelę opisującą danę i zweryfikowaliśmy wartości maksymalne i minimalne zaczynając od kolumny drugiej aż do ostatniej.

Klasyfikacja Zadaniem klasyfikacji jest przydzielenie Dla celów klasyfikacji, podstawowym narzędziem, którego możemy użyć jest regresja liniowa i logistyczna. Jako że dla danych wartości parametrów mamy tylko 2 możliwe wyniki, tj. symulacja powiedzie się lub nie, logicznym wyborem jest wykorzystanie regresji logistycznej, która pozwala przewidzieć wynik binarny.

Regresja Logistyczna Nazwa regresji logistycznej pochodzi od wykorzystywanej w niej funkcji logistycznej, która przyjmuje dowolną wartość z zakresu (0-1) i odwzorowuje je dokładnie na 1 lub 0. Ogólnym wzorem funkcji logistycznej jest:

$$y(x) = \frac{1}{1 + e^{-x}}$$

	Study	Run	vconst_corr	vconst_2	vconst_3
count	538.000000	538.000000	538.000000	538.000000	538.000000
mean	2.003717	90.286245	0.500628	0.499685	0.498483
std	0.816488	51.982591	0.288947	0.289300	0.288463
min	1.000000	1.000000	0.000414	0.001922	0.001181
25%	1.000000	45.250000	0.250620	0.250288	0.248889
50%	2.000000	90.000000	0.499998	0.498645	0.498887
75%	3.000000	135.000000	0.750795	0.750730	0.747663
max	3.000000	180.000000	0.999194	0.998815	0.998263
	vconst_4	vconst_5	vconst_7	ah_corr	ah_bolus
count	538.000000	538.000000	538.000000	538.000000	538.000000
mean	0.500054	0.501345	0.501065	0.500126	0.499939
std	0.288701	0.288396	0.288665	0.288985	0.288861
min	0.001972	0.001889	0.000476	0.004590	0.000296
25%	0.251064	0.251983	0.252380	0.254232	0.251270
50%	0.500456	0.501913	0.501757	0.499070	0.500074
75%	0.749041	0.750208	0.750201	0.749003	0.747758
max	0.997673	0.998944	0.997142	0.998930	0.998506
	efficiency_factor	tidal_mix_max	vertical_decay_scale	convect_corr	bckgrnd_vdc1
count	538.000000	538.000000	538.000000	538.000000	538.000000
mean	0.500709	0.499109	0.500082	0.500369	0.499388
std	0.289308	0.289208	0.288520	0.289049	0.288762
min	0.002015	0.000419	0.001188	0.001312	0.002509
25%	0.250250	0.249863	0.250759	0.251371	0.248744
50%	0.503133	0.498295	0.500151	0.500625	0.499080
75%	0.751126	0.747833	0.747511	0.751128	0.749161
max	0.999536	0.999942	0.997718	0.997518	0.999795
	bckgrnd_vdc_ban	bckgrnd_vdc_eq	bckgrnd_vdc_psim	Prandtl	outcome
count	538.000000	538.000000	538.000000	538.000000	538.000000
mean	0.500122	0.501868	0.499197	0.500602	0.914498
std	0.289170	0.287792	0.288761	0.289393	0.279887
min	0.000732	0.002749	0.000219	0.000263	0.000000
25%	0.250402	0.253663	0.250790	0.248217	1.000000
50%	0.499959	0.502311	0.498955	0.500931	1.000000
75%	0.749893	0.749923	0.747720	0.751430	1.000000
max	0.999155	0.997265	0.999306	0.999655	1.000000

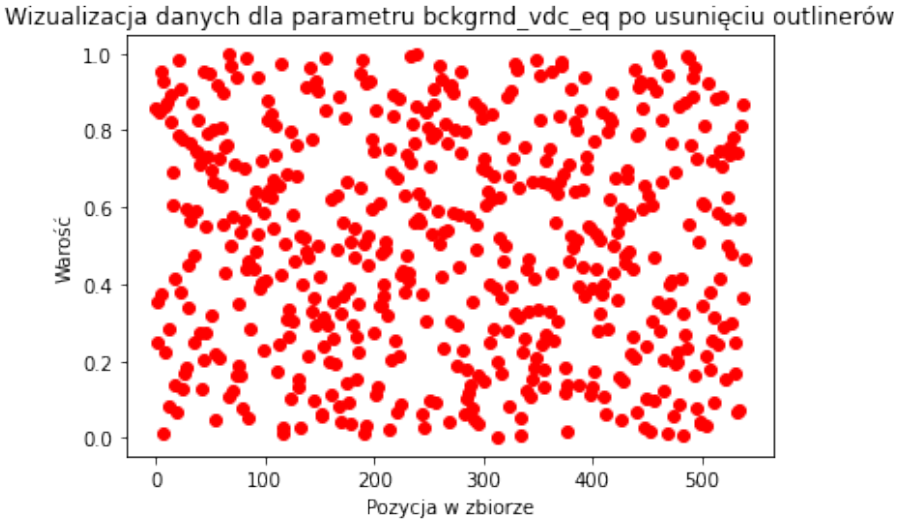


Figure 4: bckgrnd_vdc_eq

Gdzie naszym x będzie odpowiednia wartość rzeczywista, co można też zapisać zgodnie z naszym przypadkiem:

$$y(x) = \frac{1}{1 + e^{-\left(\sum_{n=0}^{17} \beta_n * x_n\right)}}$$

Gdzie β_n to współczynnik ważności danej zmiennej wyuczony przez algorytm a x_n kolejny parametr symulacji.

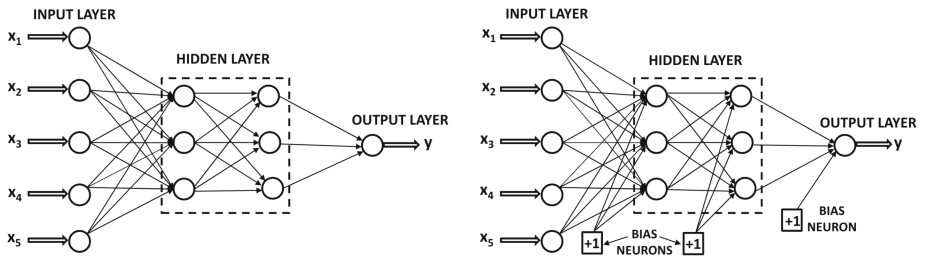
Klasyfikacja z wykorzystaniem sieci neuronowych MLPClassifier Jako kolejną metodę klasyfikacji wybraliśmy wykorzystanie sieci neuronowych i klasyfikatora MLP(Multi-layer Perceptron). Klasyfikacja ta wykorzystuje Perceptrony, proste sieci neuronowe oparte na neuronie McCullocha-Pittsa, który posiada wiele wejść i jedno wyjście, na które podawana jest wartość funkcji aktywacji dla wartości podanej wzorem:

$$s = \omega_0 + \sum_{i=1}^{n} x_i \omega_i$$

gdzie x_i to wartości podane na wejścia i ω_i to wagi wejść.

Sieć MLP składa się z jednej warstwy wejściowej, pewnej ilości warstw ukrytych i jednej warstwy wyjściowej.

Gdzie dla danej warstwy wejściowej podawane są wartości wejściowe, dla nich



(a) No bias neurons (b) With bias neurons

Figure 5: schematMLP

wykonywana funkcja aktywacji i przekazywana do warstw ukrytych, aż do warstwy wyjściowej która przekazuje ostateczny wynik.

Perceptron MLP można zapisać jako funkcję

$$y = f(x, \theta)$$

Gdzie y będzie tradycyjnym wyjściem sieci, x - wejściem, a θ parametrami wyznaczonymi podczas uczenia sieci. Funkcja f jest złożeniem wielu funkcji

$$f(x) = f^{(m)}\left(...f^{(3)}(f^{(2)}(f^{(1)}(x)))\right)$$

gdzie m jest numerem warstwy sieci

Warstwy ukryte z nieliniowymi funkcjami aktywacji przekształcają wejściowy wektor cech do ukrytej przestrzeni (latent space) Przekształcenie jest nieliniowe (następuje "wygięcie" przestrzeni) Dane w przestrzeni ukrytej są liniowo separowalne Neurony w warstwie wyjściowej dokonują liniowej klasyfikacji (przecinają przestrzeń ukrytą hiperpłaszczyznami)

Klasyfikacja z wykorzystaniem jeden przeciwko reszcie Kolejną metodę klasyfikacji jaką zdecydowaliśmy się sprawdzić była strategia jeden przeciwko reszcie(One vs Rest, OvR) z zastosowaniem regresji logistycznej.

Dzięki tej strategii, znacznie łatwiej przeprowadzić klasyfikację wieloklasową. Zakłada ona przekształcenie zbioru danych w wiele mniejszych problemów binarnych, które porównywane na zasadzie: Pasuje do zbioru A/Nie pasuje do zbioru A, pasuje do zbioru B/Nie pasuje do zbioru B Pozwala na rozwiązanie problemu wielu możliwych rozwiązań. Wykorzystanie tej klasyfikacji w przypadku rozwiązania binarnego nie wnosi większych zysków względem zwykłej klasyfikacji za pomocą regresji logistycznej.

Probabilistyczna klasyfikacja porażek

Jako że dla danych wartości parametrów mamy tylko 2 możliwe wyniki, tj. symulacja powiedzie się lub nie, możemy to sprowadzić do dwuklasowej zmiennej

w której porażki należeć będą do klasy C_p, a sukcesy do C_s. Naszym zadaniem jest wyznaczenie szansy na to że symulacja POP2 się nie powiedzie. Korzystając z warunkowego prawdopodobieństwa $P\left(C_p|x\right)$ i zasady Bayesa możemy je zapisać jako:

$$P\left(C_p|x\right) = \frac{P\left(x|C_p\right)P\left(C_p\right)}{P\left(x|C_p\right)P\left(C_p\right) + P\left(x|C_s\right)P\left(C_s\right)}$$

Gdzie $P\left(x|C_i\right)$ i $P\left(C_i\right)$ odpowiadają odpowiednio gęstościom warunkowym klas i poprzedzającym klasę. Wprowadzając zmienną reprezentującą logarytm naturalny ilorazu wiarogodności i szans:

$$\lambda = ln\left[\frac{P(x|C_p)}{P(x|C_s)} \quad \frac{P(C_p)}{P(C_s)}\right]$$

Możemy to zapisać jako funkcję sigmoidalną, tj. sprowadzić to do funkcji logistycznej.

$$P\left(C_p|x\right) = \frac{1}{1 + exp(-\lambda)}$$

Klasyfikacja SVM Klasyfikacja maszyn wektorów nośnych (SVM) z dziedzin rozpoznawania wzorców i nadzorowanego uczenia maszynowego służy do przypisania symulacji do klasy Cf lub Cs dla wektora wejściowego x. Krótko mówiąc, metoda SVM opiera się na maksymalizacji odległości między hiperpłaszczyznami, które dzielą klasy (tj. marginesie), przy jednoczesnym uwzględnieniu błędnej klasyfikacji z nakładających się punktów danych podczas treningu (tj. miękkiego marginesu). W przypadku klas nieliniowo separowalnych hiperpłaszczyzny są określane przez przekształcenie przestrzeni wejściowej do przestrzeni cech o wyższym wymiarze przy użyciu funkcji jądra. Celem przekształcenia jest ułatwienie rozdzielenia klas. Wektory wsparcia to punkty treningowe, które leżą na hiperpłaszczyznach zoptymalizowanego marginesu. Nowe wektory wejściowe x są przypisywane do klasy za pomocą znaku predykcyjnej funkcji decyzyjnej

$$f\left(x\right) = \sum_{i=1}^{N_s} y_i \alpha_i K(x_i, x) + b,$$

gdzie f (x) $>$ 0 i f (x) $<$ 0 są przypisane odpowiednio do klas C_p i C_s. Suma w równaniu jest ponad N wektorów wsparcia ze zbioru uczącego, y_i 2 {-1,1} jest zmienną binarnego wskaźnika wyniku, $K(x_i, x)$ jest funkcją jądra, a b i a_i są odpowiednio odchyleniem i Warunki mnożnika Lagrange'a określone poprzez ograniczoną optymalizację marginesu. Funkcja decyzji przypisuje dane wejściowe do klasy, ale nie podaje prawdopodobieństwa przynależności do klasy. W związku z tym opracowano rozszerzenie standardowego podejścia SVM (Platt, 1999), które wyprowadza prawdopodobieństwa klas poprzez dopasowanie do równania 2 do funkcji dwuparametrowej z wykorzystaniem danych uczących i walidacji krzyżowej.

Predykcja porażek bazuje na klasyfikatorach:

$$\mu_c = \frac{1}{N_b} \sum_{i=1}^{N_b} P_i \left(C_p|x\right)$$

i

$$\sigma_c^2 = \frac{1}{N_b} \sum_{i=1}^{N_b} [P_i \left(C_p|x\right) - \mu_c]^2$$

Gdzie μ_c i σ_c^2 są średnią i odchyleniem standardowym.

Przygotowanie i wykonanie klasyfikacji

Podział danych na zestaw treningowy i testowy

W celu przygotowania danych do klasyfikacji wykonaliśmy podział danych na zestaw treningowy i testowy przy pomocy funkcji train_test_split z biblioteki sklearn.model_selection. Zdecydowaliśmy się przeznaczyć 35% danych do testowania a posostałe 65% na uczenie klasyfikacji tak, aby osiągnąć jak najlepsze wyniki. Dodatkowo do danych wczytaliśmy wszytkie kolumny poza "Study" i "Run", aby zapobiec sytuacji w której klasyfikator wykorzystywał by którąś z nich zamiast parametrów które nie miały rzeczywistego wpływu na stabilność modelu pogodowego.

Klasyfikacja z wykorzystaniem regresji logistycznej

W ramach pierwszego klasyfikatora wykorzystaliśmy klasyfikator oparty na regresji logistycznej.

Po jego wytrenowaniu otrzymaliśmy model o celności 95% co było bliskie średniej 94% wyliczonej z waldiacji krzyżowej poniższych wartości:

[0.94736842 0.94736842 0.94736842 0.92105263 0.94594595]

Macierz konfuzji:

[[2 9] [0 178]]

Macierz konfuzji w reprezentacji graficznej:

Klasyfikacja z wykorzystaniem sieci neuronowych MLPClassifier

Kolejną klasyfikacją którą zastosowaliśmy na zbiorze danych jest MLPClassifier który działa z wykorzystaniem sieci neuronowych. Do zautomatyzoawania szukania najlepszych parametrów wykorzystaliśmy GridSearchCV którego zadaniem jest wykonanie wszytkich możliwych kombinacji parametów zadanych w punkci numer 5.1 w kodzie tak aby znaleść najlepsze rozwiązanie. Do znalezienia najleszego rozwiązania do wykorzystania daliśmy parametry takie jak ilość neuronów w warstwie ukrytej, ilość iteracji, metodę aktywacji neuronów, funkcję

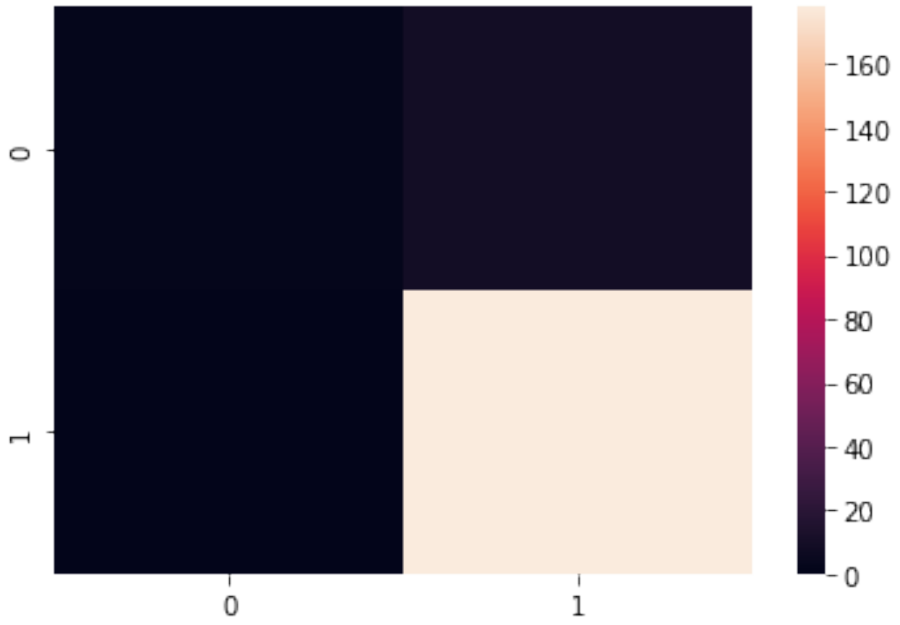


Figure 6: macierz_konfuzji_logistycznej

aktywacji neuronów. Dodatkowo tykorzystaliśmy parametr “early_stopping” aby przerwać przetwarzanie tych modeli które przestają się polepszać po 10 iteracjach co znacznie przyśpieszyło obliczanie modeli.

Dla naszego zbioru danych najlepszymi parametrami są

{‘activation’: ‘tanh’, ‘early_stopping’: True, ‘hidden_layer_sizes’: (60, 40, 20), ‘max_iter’: 300, ‘solver’: ‘lbfgs’}

I dają one wynik na poziomie 96% celności w klasyfikowaniu.

Macierz konfuzji: [[9 2] [6 172]]

Macierz konfuzji w reprezentacji graficznej:

Dodatkowo wygenerowaliśmy wykres wyników wszystkich 108 kandydatów z podziałem na wynik w danych treningowych i testowych

Dla każdej z warstw zwizualizowaliśmy również elementy które aktywują zawarte w niej neurony. Poniżej graficzna reprezentacja aktywacji neuronów dla pierwszej warstwy ukrytej

Aktywacja neuronów dla drugiej warstwy ukrytej

Aktywacja neuronów dla trzeciej warstwy ukrytej

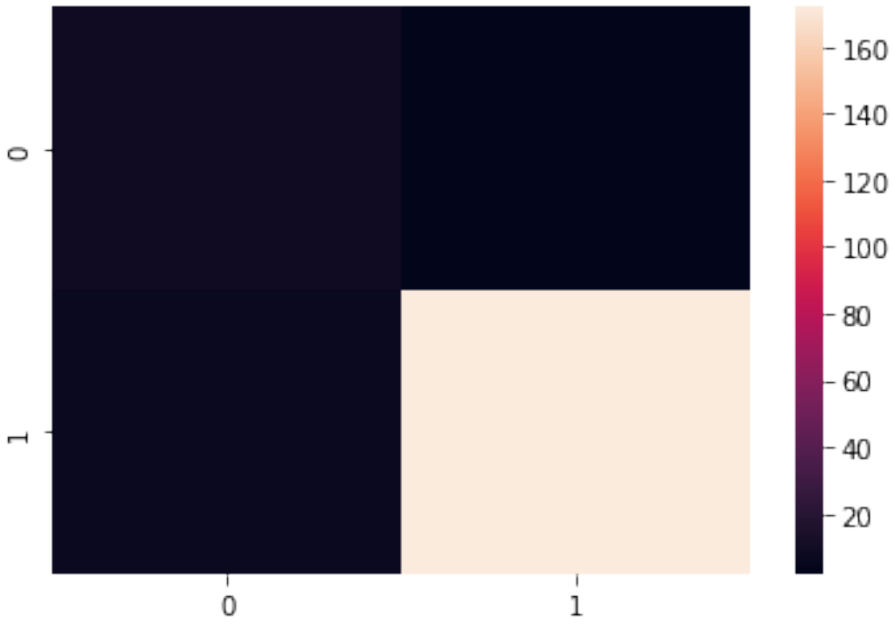


Figure 7: macierz_konfuzji_MLP

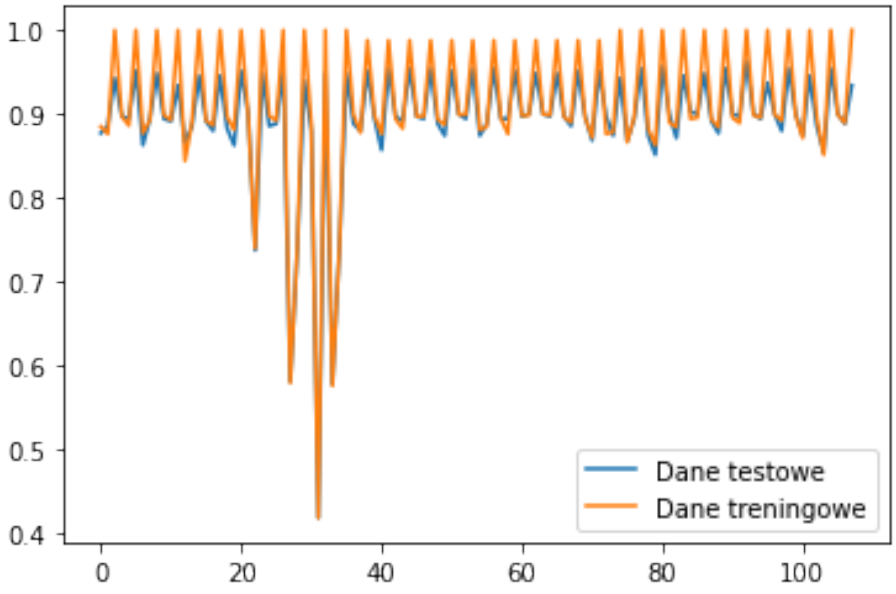


Figure 8: wyniki_MLP

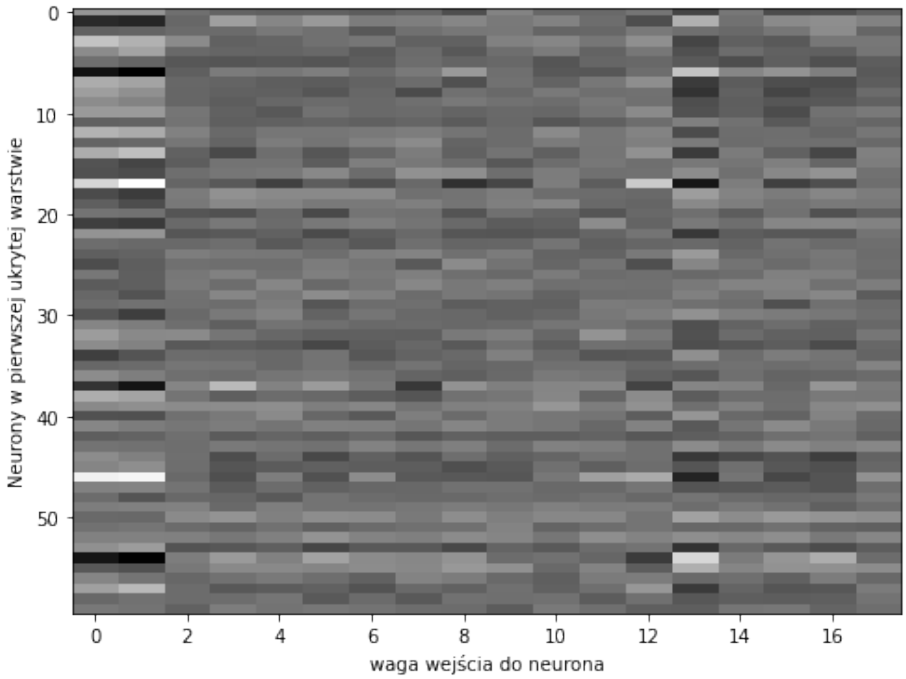


Figure 9: warstwa1_neuron

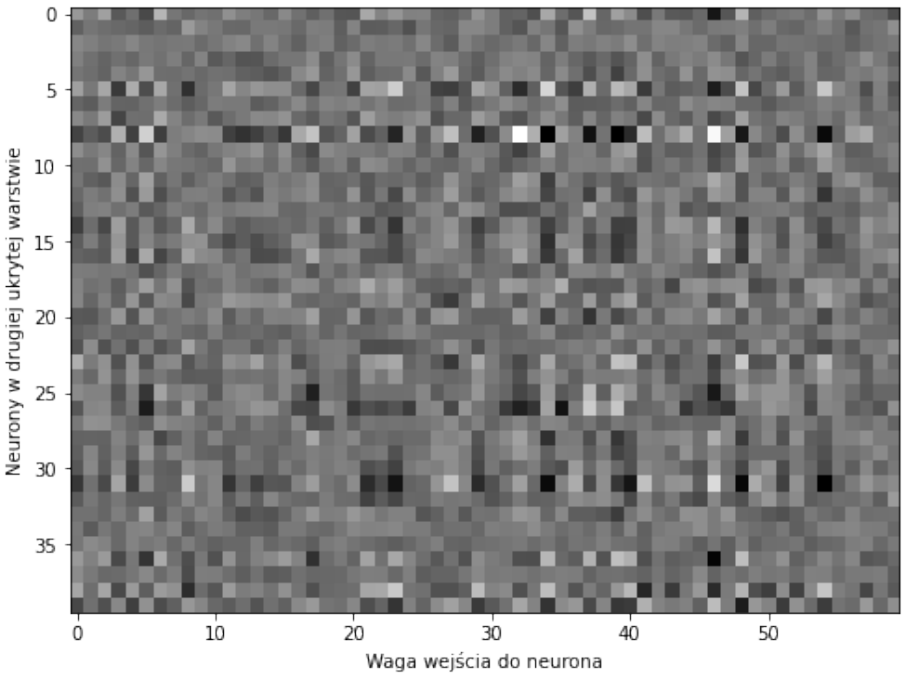


Figure 10: warstwa2_neuron

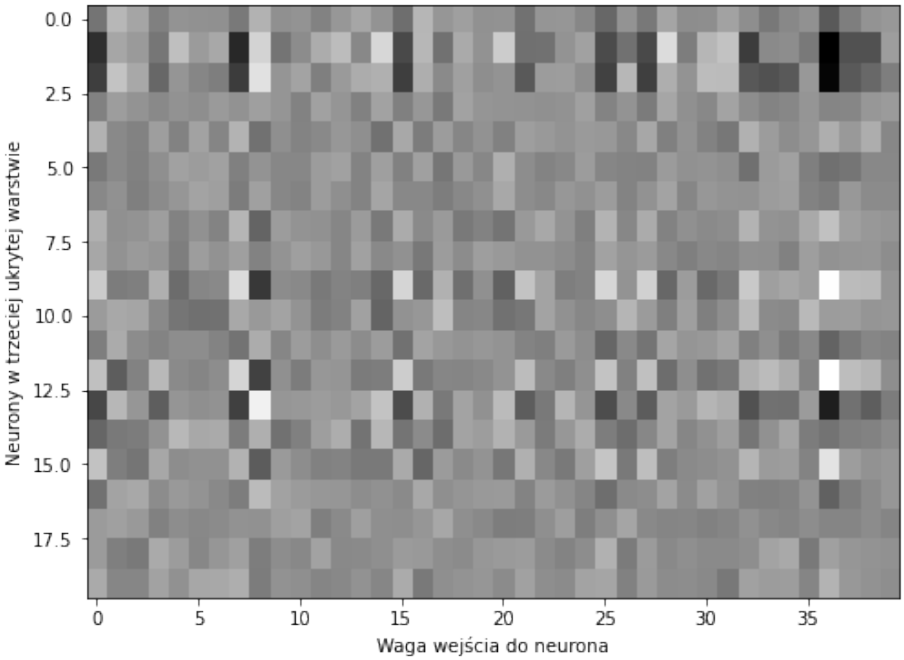


Figure 11: warstwa3_neuron

Klasyfikacja z wykorzystaniem jeden przeciwko reszcie

W kolejnym klasyfikatorze wykorzystaliśmy regresję logistyczną w ramach strategii "jeden przeciwko reszcie". W przypadku tej strategii klasyfikator ocenia czy dany obiekt pasuje do klasy lub nie. Przy pomocy tej klasyfikacji uzyskaliśy rezultat 95% skuteczności w klasyfikacji co jest dość bliskie wynikom z waldiacji krzyżowej. [0.94736842 0.94736842 0.94736842 0.92105263 0.94594595]

Macierz konfuzji: [[2 9] [0 178]]

Macierz konfuzji w reprezentacji graficznej:

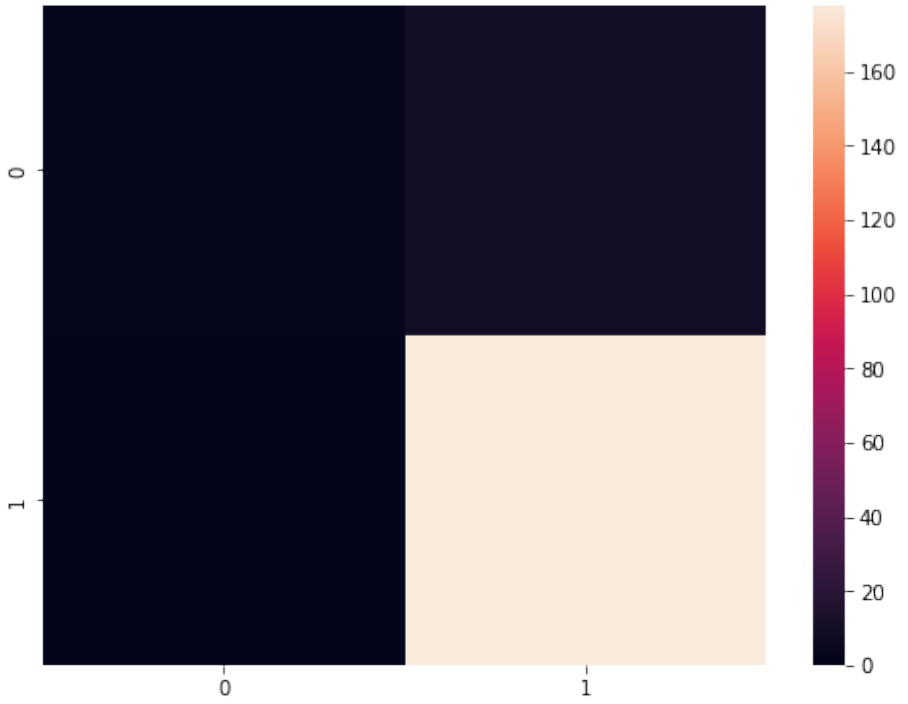


Figure 12: macierz_konfuzji_OVR

Klasyfikacja z wykorzystaniem liniowej klasyfikacji wektorów nośnych

W ostatnim opracowanym przez nas klasyfikatorze wykorzystaliśmy wektory nośnie jako sposób klasyfikacji danych. Przy ich zastosowaniu otrzymaliśmy celność modelu na poziomie 97% która była zawyżona względem średniej z waldiacji krzyżowej która wynosiła 93,64%

[0.94736842 0.94736842 0.94736842 0.92105263 0.91891892]

Macierz konfuzji: [[9 2] [4 174]]

Macierz konfuzji w reprezentacji graficznej:

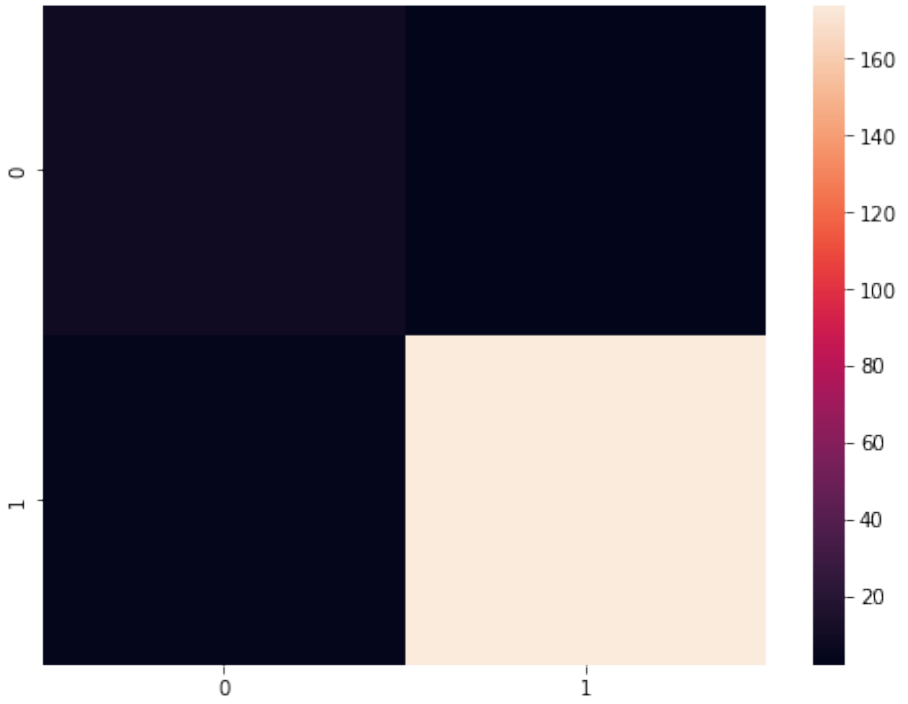


Figure 13: macierz_konfuzji_SVC

Podsumowanie

Każdy z badanych przez nas klasyfikatorów osiągnął skuteczność na poziomie przynajmniej 90% co jest bardzo dobrym wynikiem i pozwala nam na wykorzystanie każdego z modeli. Jeżeli mielibyśmy wskazać najlepszy klasyfikator do tego zbioru danych to będzie to sieć neuronowa MLPClassifier, ale jej wadą jest relatywnie długi czas wyliczania najoptymajniejszych parametrów ze względu na ilość kombinacji które chcieliśmy zbadać. Przy większym zbiorze danych lub takim który jest bardziej złożony np. obrazy czas przetwarzania stałby się dużą wadą. Pozostałe rozwiązania są równie szybkie a ich obliczenie zajmuje znacznie mniej czasu.

Dodatkowym aspektem który nie został przez nas zbadany jest rozmiar zapisanego modelu który miałby znaczenie przy mobilnych zastosowaniach modelu.

Źródła

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep learning.
2. C. C. Aggarwal, Neural Networks and Deep Learning, Springer 2018

3. Failure analysis of parameter-inducedsimulation crashes in climate modelsD. D. Lucas1, R. Klein1,2, J. Tannahill1, D. Ivanova1, S. Brandon1, D. Domyancic1 and Y. Zhang1
4. https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/
5. http://if.pw.edu.pl/~tomgrad/sn/04-mlp/04-mlp.html#/

Repozytorium https://github.com/WWindak/Climate_Model_Simulation_Crashes