

Akademia Górniczo-Hutnicza Im. Stanisława Staszica w Krakowie

Wydział Zarządzania

Informatyka i Ekonometria

Elementy Sztucznej Inteligencji

Opracowała:

Natalia Głodek

Spis treści

[1. Opis podjętego problemu 4](#_Toc105275547)

[Co przedstawiają dane? 4](#_Toc105275548)

[2. Analiza oraz przygotowanie danych do modelu 4](#_Toc105275549)

[Brakujące wartości 4](#_Toc105275550)

[Dobór zmiennych 5](#_Toc105275551)

[Skalowanie danych 5](#_Toc105275552)

[Podział na zbiór uczący i testowy 6](#_Toc105275553)

[Analiza danych 6](#_Toc105275554)

[3. Literatura 7](#_Toc105275555)

[*Machine learning with python cookbook: Practical solutions from preprocessing to deep learning* 7](#_Toc105275556)

[www.acme.byu.edu/wp-content/uploads/2020/01/Spark.pdf 7](#_Toc105275557)

[*Machine Learning: Data Representation -* https://faculty.cs.nku.edu/~waldenj/classes/2019/spring/csc640/lectures/ml-data-representation.pdf 7](#_Toc105275558)

[*Learning scikit-learn: Machine Learning in Python* 7](#_Toc105275559)

[https://www.kaggle.com/code/esraasayed98/titanic-machine-learning-from-disaster 7](#_Toc105275560)

[4. Stworzenie Sztucznej Sieci Neuronowej 8](#_Toc105275561)

[Model sieci 8](#_Toc105275562)

[Pierwsze przejście 10](#_Toc105275563)

[Uczenie sieci 10](#_Toc105275564)

[5. Analiza wybranych czynników na skuteczność działania sieci 11](#_Toc105275565)

[Ilość warstw 12](#_Toc105275566)

[Usunięcie jednej warstwy – 0 wartsw ukrytych 12](#_Toc105275567)

[Dodanie jednej warstwy – 2 warstwy ukryte 12](#_Toc105275568)

[Wnioski 12](#_Toc105275569)

[Ilość neuronów 13](#_Toc105275570)

[Liczba neuronów -2 13](#_Toc105275571)

[Ilość neuronów – 5 13](#_Toc105275572)

[Ilość neuronów – 6 14](#_Toc105275573)

[Liczba neuronów – 50 14](#_Toc105275574)

[Liczba neuronów – 500 14](#_Toc105275575)

[Wnioski: 15](#_Toc105275576)

[Funkcja aktywacji 15](#_Toc105275577)

[Funkcja sigmoid: 15](#_Toc105275578)

[Funkcja liniowa 16](#_Toc105275579)

[Funkcja tanh 17](#_Toc105275580)

[Wnioski 17](#_Toc105275581)

[Wielkość i sposób doboru próby uczącej i testowej 17](#_Toc105275582)

[Ucząca – 20%, testowa – 80% 17](#_Toc105275583)

[Ucząca – 50%, testowa – 50% 18](#_Toc105275584)

[Ucząca – 99%, testowa 1% 18](#_Toc105275585)

[Wnioski 18](#_Toc105275586)

[Wnioski 19](#_Toc105275587)

[6. Podsumowanie 19](#_Toc105275588)

# Opis podjętego problemu

Praca zajmuję się przewidywaniem skutku podróży pasażera Tytanika. Opierać się będzie na danych z pliku „titanic.csv”, jest to powszechny zbiór danych do uczenia maszynowego. Sieć stworzyłam w języku python bez użycia bibliotek. Będzie ona podawała dwa prawdopodobieństwa: na przynależność do klasy pierwszej (1 – przeżył) oraz przynależność do klasy drugiej (0 – nie przeżył). Swoje rozważania zaczęłam od przygotowania i analizy danych.

## Co przedstawiają dane?

Początkowo tabelka zawiera kolumny z informacjami o:

* imieniu pasażera,
* klasie pokładu (pclass: 1 – upper, 2 – middle, 3 – lower),
* płci,
* wieku,
* ilości dzieci/rodziców na pokładzie (parch),
* ilości rodzeństwa/małżonków na pokładzie (sibsp),
* opłacie pasażerskiej (fare),
* numerze kabiny
* porcie zaokrętowania (embarked: C = Cherbourg, Q = Queenstown, S = Southampton)
* tym czy pasażer przeżył czy nie (survived: 1 – przeżył, 0 – nie przeżył)

Tabela 1 pierwsze 5 wierszy z tabeli titanic.csv

Obraz zawierający stół

Opis wygenerowany automatycznie

# Analiza oraz przygotowanie danych do modelu

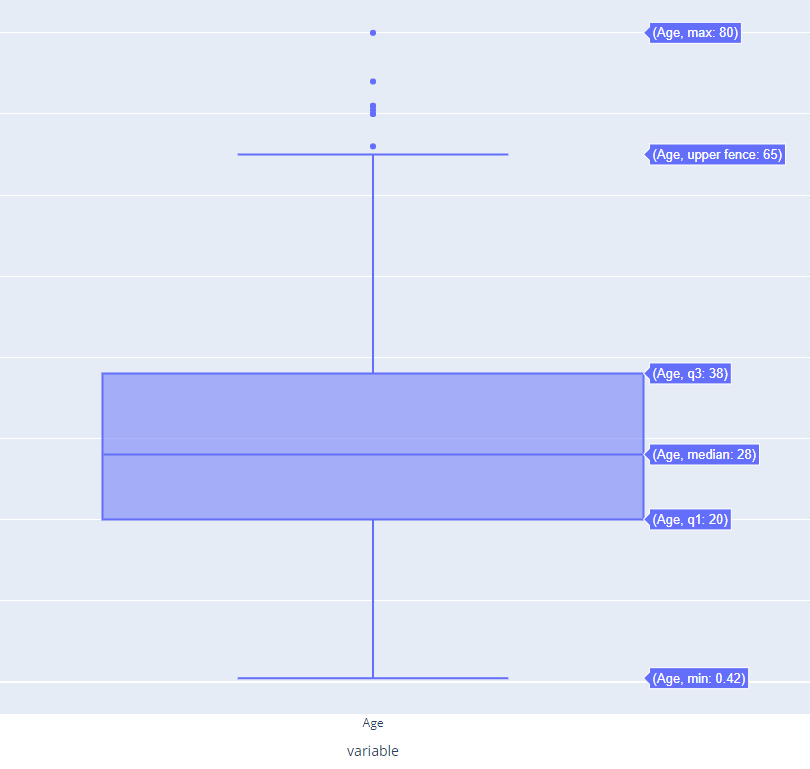
## Brakujące wartości

Obraz zawierający tekst, urządzenie, pomiar

Opis wygenerowany automatycznieNastępnie sprawdzam jak dużo braków znajduje się w naszym zbiorze:

Tabela z sumą braków

W przypadku kolumny embarked braki są na tyle niewielkie, że możemy usunąć te wiersze całkowicie. Jednak w przypadku zmiennej wiek usunięcie braków spowodowało stratę zbyt dużej liczby obserwacji, dlatego braki zastąpimy średnią wieku. Po sprawdzeniu czy nie ma wartości odstających, wyliczam średnią i wstawiam ją w miejsce Nan.



Boxplot – wartości odstające

## Dobór zmiennych

Postanowiłam usunąć kolumny takie jak id, imię, numer kabiny i biletu, gdyż są one zbyt indywidualne. Kolumny Sex i Embarked to dane kategoryczne dlatego zmieniam je na numeryczne (dummy varies). Po zmianach dane wyglądaj tak:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Dane po zmianie

## Skalowanie danych

Ponieważ wartości w kolumnie age i fare są dość duże przekształcam je na takie z zakresu od 0 do 1.. Od każdej wartości odejmuje wartość minimalną dzięki czemu będą zaczynały się od 0. Następnie dzielę przez różnicę między największą a najmniejszą wartością. Po obróbce dane prezentują się następująco:

Obraz zawierający tekst

Opis wygenerowany automatycznie

Dane po normalizacji

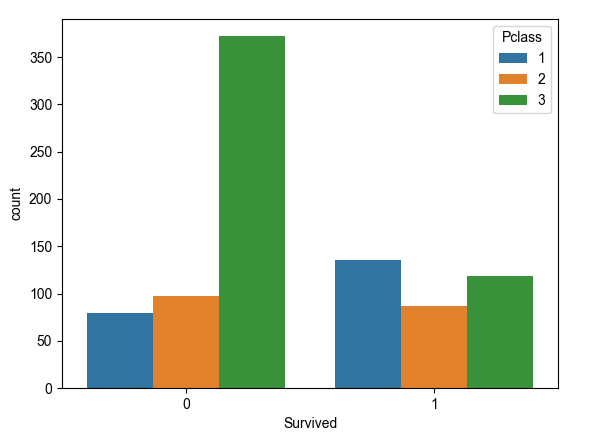
## Podział na zbiór uczący i testowy

Tworze tabele x\_data, czyli dane bez kolumny Survived oraz tabele y\_data, czyli wektor z kolumną survived. Modyfikuje wektor y\_data tak, aby wyglądał zgodnie z metodą hot coding, ułatwi mi to później wyliczanie straty. Teraz wektor y zawiera wektory [1,0] – oznacza, że dana obserwacja jest z pierwszej klasy (przeżył) lub [0,1] – obserwacja jest z drugiej klasy (nie przeżył). Następnie dzielę zbiór na uczący i testowy w proporcji 90% do 10%.

## Analiza danych

wykres 1 -

Wykres 2 - przeżycie według płci



wykres 3 - przeżycie według klasy

Jak widać powyżej większość ludzi nie przeżyła. Wśród nich było znacznie więcej mężczyzn oraz osób z 3 klasy, czyli najtańszej. Wśród osób, które przeżyły dominują kobiety z 1 klasy.

# Literatura

## *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*

W przytoczonej książce tej autor opisuje jak przygotowywać dane do modelu w pythonie. Opisał między innymi jak usunąc wartości brakujące, jak usunąć kolumnę, czy jak połączyć dwie tabele. Autor w wielu przypadkach użył podobnych funkcji lub metod do tych, które wykorzystuje w swojej pracy.

## [www.acme.byu.edu/wp-content/uploads/2020/01/Spark.pdf](http://www.acme.byu.edu/wp-content/uploads/2020/01/Spark.pdf)

Autor opisuje jak używać Apache Spark do uczenia maszynowego oraz obróbki danych.

Do predykcji używa Spark ML. Podobnie jak ja, zmienia on kolumnę pclass i sex na wartości numeryczne, lecz dokonuje to w bardziej profesjonalny sposób. Następnie dzieli próbę na: zbiór uczący i testowy w proporcji 75% i 25% oraz biorąc losowe wartości. Całość „uczenia” odbywa się za pomocą wbudowanych bibliotek (np. LogisticRegression). Jego dokładność (accuracy) na początku wynosi 80%, natomiast po ewaluacji 84%. Jest to wynik nieco lepszy od mojego, który wynosi około 78% dokładności.

Obraz zawierający tekst, stół

Opis wygenerowany automatycznie

Użyte przez autora biblioteki do SSN.

## *Machine Learning: Data Representation -* <https://faculty.cs.nku.edu/~waldenj/classes/2019/spring/csc640/lectures/ml-data-representation.pdf>

Autor opowiada o tym jak przygotować dane do modelu. Mówi między innymi o tym jak zeskalować dane. Podaje pięć różnych technik. Ja zdecydowałam się użyć MinMax Scaling. Można było również znormalizować dane lub zestandaryzować. Podał również nazwy gotowych bibliotek i funkcji (from sklearn.preprocessing import MinMaxScaler) do tego celu. Autor wspomina również o transformacji danych, aby model działał lepiej, między innymi użycie wzoru log(1 + x) ma sprawić, że model będzie szybciej znajdował wzory. W mojej pracy taka transformacja nie miała miejsca.

## *Learning scikit-learn: Machine Learning in Python*

W książce tej autor pokazuje jak użyć biblioteki *sklearn* aby zaprognozować dane ze zbioru titanic.csv. Używa on drzewa decyzyjnego i dokładność stworzonego modelu wynosi 84%. Ponownie nie jest to liczba, która znacząco różni się od uzyskane przeze mnie. Pokazuje on również jak wybrać najbardziej istotne zmienne, a używa do tego testu chi kwadrat.

### <https://www.kaggle.com/code/esraasayed98/titanic-machine-learning-from-disaster>

Autor prezentuje swój kod na stronie kaggle.com. Po przygotowaniu danych, tak jak w poprzednich opracowaniach używa pakietu *sklearn*. Tworzy klasyfikator i następnie go testuje. Dokładność w jego przypadku wynosi 79%, więc jest bardzo zbliżona do mojej.

# Stworzenie Sztucznej Sieci Neuronowej

## Model sieci

Zaprojektowałam moją sieć tak, aby bez trudu można było dodawać warstwy, zmieniać funkcje aktywacji czy liczbę neuronów.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Klasa warstwy odpowiada za utworzenie warstwy oraz przypisaniu jej wag i biasów, które są losowane z przedziału -0.1 do 0.1

Obraz zawierający tekst

Opis wygenerowany automatycznie

Klasa Funkcje\_aktywacji zawiera 4 różne funkcje aktywacji: ReLU, tanh, liniową oraz sigmoid.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Klasa Aktywacja\_Softmax sprawia, że output reprezentuje przewidywane prawdopodobieństwa wszystkich klas, sumując się do 1.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

Klasa Ocena odpowiada za wyliczenie straty oraz accuracy.

W funkcji calc\_loss wyliczam entropię krzyżową (class-entropy) , którą wykorzystuje jako funkcję straty. Oblicza ona różnicę między dwoma rozkładami prawdopodobieństwa.

W funkcji calc\_accuracy wyliczam 4 możliwe wyjścia:

* Dobrze zaprognozowane przeżycie – TN – true positive
* Dobrze zaprognozowane nie przeżycie TN – true negative
* Źle zaprognozowane przeżycie (w predykcji wyszło, że nie przeżył a tak naprawdę przeżył) – FN – false negative
* Źle zaprognozowane nie przeżycie (w predykcji wyszło, że przeżył a tak naprawdę nie przeżył) – FP – false positive

Dzięki nim obliczam accuracy – czyli miarę dokładności. Jednak miara ta nie zawsze jest idealna. Powodem jest to, że w przypadku, gdy mamy więcej obserwacji z jednej klasy to duża liczba „odgadniętych” wartości ma duży wpływ na wynik dokładności i eliminuje błędy. Dlatego dodatkowo obliczam sensitivity – współczynnik poprawnych pozytywnych oraz Specificity – współczynnik poprawnych negatywnych.

## Pierwsze przejście

Obraz zawierający tekst

Opis wygenerowany automatycznie

Tworzę pierwszą sieć, ma ona 3 warstwy. Input ma 10 neuronów, ponieważ każda kolumna odpowiada za jeden neuron, warstwa ukryta ma 3 neurony, natomiast output ma 2 neurony, ponieważ w naszym przypadku prognozujemy prawdopodobieństwo przynależenia do dwóch klas.

## Uczenie sieci

Obraz zawierający tekst

Opis wygenerowany automatycznie

Zapisuje wagi jako najlepsze po pierwszym przejściu oraz ustawiam minimalny błąd na 1000 a następnie z każdym kolejnym przejściem sprawdzam, czy ten błąd jest mniejszy, jeżeli tak to zapisuje nowe wagi i nową najmniejszą stratę

Następnie badam działanie sieci na zbiorze testowym.

# Analiza wybranych czynników na skuteczność działania sieci

Analizę przeprowadzę zmieniając jeden czyynik a resztę pozostawiając bez zmian. Zacznę od modelu bazowego:

* 3 warstwy (input, warstwa ukryta, output
* 3 neurony w warstwie ukrytej
* Funkcja ReLu
* 800 obserwacji w zbiorze uczącym, 89 w zbiorze testowym
* Obraz zawierający tekst

  Opis wygenerowany automatyczniena outpucie zastosowana jest funkcja softmax

Podczas uczenia strata maleje a dokładność modelu wzrasta do 76%. W przypadku zbioru testowego jest to 87%

Obraz zawierający stół

Opis wygenerowany automatyczniePoniższa grafika pokazuje loss, accuracy, sensitivity oraz specificity dla kolejnych 10 prób:

Średnio błąd wynosi: 0.52 , natomiast dokładność: 78%

## Ilość warstw

### Obraz zawierający stół Opis wygenerowany automatycznieUsunięcie jednej warstwy – 0 wartsw ukrytych

Wyniki nie różnią się znacznie od pozostałych a sieć jest dużo prostsza. W niektórych przypadkach osiągamy nawet bardzo wysoką accuracy, na poziomie 83%.

### Dodanie jednej warstwy – 2 warstwy ukryte

Obraz zawierający tekst

Opis wygenerowany automatycznie

### Dodanie jednej warstwy komplikuje model . Ma on również większą stratę niż w przypadku 1 warstwy.

### Wnioski

Najlepiej jest gdy model ma 1 warstwę ukrytą, czyli 3 warstwy łącznie. Jednak w przypadku zastosowania 2 warstw łącznie jest on wystarczający. Dwie warstwy ukryte pogarszają dokładność modelu

## Ilość neuronów

### Obraz zawierający stół Opis wygenerowany automatycznieLiczba neuronów -2

Jest to jedna z najlepszych opcji.

### Obraz zawierający stół Opis wygenerowany automatycznieIlość neuronów – 5

Wyniki są lepsze niż w przypadku 3 neuronów. Strata jest mniejsza a dokładność lepsza.

### Ilość neuronów – 6

Obraz zawierający stół

Opis wygenerowany automatycznie

Wyniki są lepsze niż w przypadku 3 neuronów i bardzo zbliżone do 5 neuronów.

### Obraz zawierający stół Opis wygenerowany automatycznieLiczba neuronów – 50

W tym przypadku wyniki są bardzo niedokładne, a strata większa od 1 sugeruje, że pojawia się błąd.

### Obraz zawierający stół Opis wygenerowany automatycznieLiczba neuronów – 500

### Wnioski:

Przy pierwszym doborze neuronów można wziąć połowę sumy neuronów w inpucie i outpucie. Częściowo się to sprawdza, ponieważ dla 6 otrzymujemy satysfakcjonujące wyniki.

Najlepiej jak warstwa ma 2 neurony, w przypadku 5,6 wyniki są nieco gorsze, ale wystarczające. Jednak przy 50 lub 500 wyniki są bardzo niekorzystne

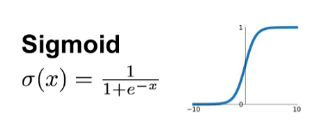
## Funkcja aktywacji

### Funkcja sigmoid:

Funkcja ta nazywana jest również funkcją logistyczną.

Im większe wejście (bardziej dodatnie), tym wartość wyjściowa będzie bliższa 1, natomiast im mniejsze wejście (bardziej ujemne), tym bliżej wyjście będzie wynosić 0.

Jest obliczana ze wzoru



Przypis 1

Obraz zawierający tekst

Opis wygenerowany automatycznieJest ona polecana dla klasyfikacji logistycznej, wiec można oczekiwać, zę wynik będą lepsze niż w przypadku funkcji ReLu.

Zgodnie z przewidywaniem wyniki są lepsze niż w przypadku funkcji ReLu. Poniżej tabelka dla kolejnych 10 iteracji.

Obraz zawierający stół

Opis wygenerowany automatycznie

### Funkcja liniowa

Jest ona nazywana funkcją tożsamościową lub po prostu brakiem funkcji aktywacji.

Obraz zawierający stół

Opis wygenerowany automatycznieObraz zawierający tekst

Opis wygenerowany automatycznieWylicza się ją ze wzoru: y=x

Strata jest średnio większa o 2% niż przy zastosowaniu funkcji ReLU i o 5% przy funkcji sigmoidalej, jednak dokładność utrzymuje się na podobnym poziomie.

### Obraz zawierający tekst Opis wygenerowany automatycznieFunkcja tanh

Wylicza się ją ze wzoru:

1

### 

### Wnioski

Najlepszą funkcją aktywacji jest funkcja sigmoid, tanh i ReLU również się sprawdzają. Najgorzej wypada funkcja liniowa.

Różnice nie są jednak bardzo duże. Może to być spowodowane użyciem funkcji Softmax, która sprowadza wyniki do prawdopodobieństwa. Jest to funkcja aktywacji na warstwie końcowej i jest ona niezbędna, aby wyniki były interpretowalne.

## Wielkość i sposób doboru próby uczącej i testowej

### Obraz zawierający tekst Opis wygenerowany automatycznieUcząca – 20%, testowa – 80%

Obraz zawierający stół

Opis wygenerowany automatycznieBłędy i dokładność dla próby uczącej

Obraz zawierający stół

Opis wygenerowany automatycznieBłędy i dokładność dla próby testowej

Na tym przypadku można zauważyć dwie rzeczy. Po pierwsze błędy są mniejsze (a dokładność większa) dla próby testowej co może wynikać z tego iż jest w nich dużo więcej danych.

Jednak jak widać w tabeli pierwszej, pojawia się kilka 1 i 0 w kolumnach 3 i 4. Ponieważ w danych jest więcej przypadków śmierci, nasz klasyfikator „stracił wszelką nadzieję i stał się pesymistą”. Dobrze klasyfikuje śmierci, jednak przeżycia klasyfikuje błędnie.

### Ucząca – 50%, testowa – 50%

Dla takiego doboru prób wyniki są bardzo zbliżone do bazowego modelu, czyli ucząca – 90% oraz testowa – 10%

### Ucząca – 99%, testowa 1%

Dla takiego doboru prób wyniki są trochę gorsze od bazowego modelu, czyli ucząca – 90% oraz testowa – 10%

### Wnioski

Najlepiej jest wybrać większą próbę uczącą (powyżej 50%). W przypadku próby uczącej mniejszej niż 40% wszystkich obserwacji model jest mniej dokładny, gdyż ma małą ilość danych do nauki.

## Trzeba również zwrócić uwagę, czy w próbie uczącej nie ma zbyt wielu obserwacji z jednej klasy. Ma to wpływ na generalizowanie i pogorszenie dokładności modelu.

## Wnioski

Dla wybranych przeze mnie danych najlepszy powinien być model o parametrach:

* 3 warstwy (input, warstwa ukryta, output
* 2 neurony w warstwie ukrytej
* Funkcja sigmoid
* 800 obserwacji w zbiorze uczącym, 89 w zbiorze testowym
* Obraz zawierający stół

  Opis wygenerowany automatyczniena outpucie zastosowana jest funkcja softmax

Zgodnie z przewidywaniami taki model jest najleszy. Strata wynosi 0,49 a accuracy aż 79%

# Podsumowanie

Zdaję sobie sprawę, że istnieje kilka możliwości na ulepszenie i rozszerzenie tego modelu (m. in. zastosowanie propagacji wstecznej). Byłoby to możliwe przy pracy w większym zespole.