

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

*Opracowanie systemu rekomendacji dla klientów sklepu internetowego*

Autor: Dominik Dawid Babiarczyk

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: *Doktor Inżynier Waldemar Bauer*

Kraków, 2024r

Spis treści

[2 Wstęp: 3](#_Toc185203966)

[3 Analiza danych: 4](#_Toc185203967)

[3.1 Metadane 4](#_Toc185203968)

[3.2 Reprezentacja 6](#_Toc185203969)

[3.3 Wyniki 7](#_Toc185203970)

[3.4 Baza danych 10](#_Toc185203971)

[4 Przygotowanie danych: 13](#_Toc185203972)

[4.1 Zmienne numeryczne 13](#_Toc185203973)

[4.2 Zmienne kategoryczne 13](#_Toc185203974)

[4.3 Dane cykliczne: 15](#_Toc185203975)

[4.4 Dane Tekstowe 17](#_Toc185203976)

[4.5 Algorytmy stemmingu 18](#_Toc185203977)

[4.6 Zbiór treningowy i walidacyjny 18](#_Toc185203978)

[5 Struktura sieci neuronowej 20](#_Toc185203979)

[5.1 Rekurencyjna sieć 20](#_Toc185203980)

[5.2 Łączenie zależności 21](#_Toc185203981)

[5.3 Warstwa TimeDistributed 22](#_Toc185203982)

[5.4 Funkcja straty: 23](#_Toc185203983)

[5.5 Metryki dokładności: 24](#_Toc185203984)

[5.6 Funkcje aktywacji: 25](#_Toc185203985)

[5.6.1 Softmax 27](#_Toc185203986)

[5.7 Optymalizator: 27](#_Toc185203987)

[6 Analiza i poprawa wyników sieci: 30](#_Toc185203988)

[6.1 Overfiting 30](#_Toc185203989)

[6.2 Regularyzacja L1 oraz L2 31](#_Toc185203990)

[6.3 DropOut: 31](#_Toc185203991)

[6.4 Regularyzacja typu max-norm 32](#_Toc185203992)

[6.5 Dogenerowanie danych: 32](#_Toc185203993)

[6.6 Interpretacja wyników 33](#_Toc185203994)

[7 Podsumowanie i wnioski 37](#_Toc185203995)

[8 Wykaz Literatury 38](#_Toc185203996)

# Wstęp:

Ogromną częścią dzisiejszego runku komercyjnego są sklepy internetowe. Duża część konsumentów przestała korzystać z tradycyjnych rozwiązań jak wyjście do sklepu na rzecz paru kliknięć w Internecie. Wystarczy tylko włączyć komputer, wejść na stronę internetową, wpisać interesujący nas przedmiot i zacząć przeglądać. Z pewnością jest to duże ułatwienie dla konsumentów ponieważ nie musimy tracić czasu na szukanie, najpierw odpowiedniego podmiotu dostarczającego produkt, a następnie samego produktu, który spełniałby nasze oczekiwania. Wszystko to dzieje się za sprawą odpowiednich filtrów, systemów wyszukujących i analizujących zakupy innych użytkowników, którzy również wpisali taką frazę jak my. W bardziej zaawansowanych sklepach stosowane są mechanizmy analizy zakupów, aby dostosowywać takie parametry jak: ilość produkowanych zasobów, cechy najbardziej pożądanych produktów z danej dziedziny oraz dostosowywanie zasobów na podstawie danych zebranych od użytkownika, aby dostarczyć jak najszybciej najbardziej pożądany produkt. Nie można ukryć, że wszystkie te zabiegi mają na celu zwiększenie zakupów oraz pomnożenie majątku danej organizacji udostępniającej produkt. Czy takie preferowanie produktów na każdym kroku jest moralne? Moim zdaniem jest to na pewno użyteczny zabieg w określonych sytuacjach, a dostosowywanie tego dla użytkowników, którzy faktycznie potrzebują produktu, oraz dla mniej zainteresowanych, jest zupełnie inną kwestią, która nie ma związku z takim systemem, ale decyzjami jak i kiedy z niej korzystać. W tej pracy postaram się stworzyć i dokładnie opisać aplikacje będzie korzystała z ogólnie dostępnej bazy danych zakupowych dostępnej na stronie Kaggle. Następnie użyje technologii do analizowania i predykcji cech produktów, aby lepiej zrozumieć przedstawione dane a następnie zastosuje je do Modelu sztucznej inteligencji, w celu uzyskania predykcji najbardziej pożądanego produktu, dla danego użytkownika. W tym celu mam zamiar użyć technologii takich jak: Python, React.js, TensorFlow. Cała aplikacja będzie miała interfejs do przeglądania wyników i oceny ich jakości.

Projekt polega na realizacji analizy i systemu rekomendacji zakupów, tak aby zminimalizować czas szukania oraz dostosować produkty pod względem ich cech, tak aby były jak najbardziej dostosowane do potrzeb i oczekiwań konsumenta

# Analiza danych:

Do stworzenia systemu rekomendacji niezbędne jest pozyskanie odpowiednich danych, na których będziemy operować. Większość dostępnych zbiorów jest surowa – dostarcza pewnych informacji, jednak znaczna ich część, zwłaszcza te dane, które są najbardziej potrzebne, stanowi jedynie fragment dostępnego zasobu. Niezbędna jest więc analiza danych pod kątem tego, co możemy z nich uzyskać po procesie wyodrębniania i przekształcania.

Przydatne może okazać się obliczenie średnich, median oraz odchyleń standardowych, aby lepiej zrozumieć rozkład i charakterystykę danych. Kluczowe znaczenie ma również analiza korelacji, która pozwala zbadać związek między poszczególnymi zmiennymi oraz określić, jak jedne wpływają na drugie. Na tej podstawie można wskazać, które dane są bardziej przydatne dla naszego modelu, a które mniej. Szczególnie istotne jest zbadanie korelacji między cechami określonego zachowania a wynikiem predykcji [1]. Po odpowiednim przygotowaniu danych można przystąpić do analizy predykcyjnej.

Analiza predykcyjna to gałąź zaawansowanej analityki, której celem jest prognozowanie przyszłych zdarzeń, zachowańi wyników. Wykorzystuje ona techniki statystyczne, takie jak algorytmy uczenia maszynowego oraz zaawansowane modelowanie predykcyjne, do analizy bieżących i historycznych danych, a także oceny prawdopodobieństwa wystąpienia różnych zdarzeń [2].

Pierwszym krokiem w stworzeniu systemu rekomendacji dla sklepu internetowego jest znalezienie odpowiedniej bazy danych. W niniejszej pracy wykorzystamy ogólnodostępne dane z platformy Kaggle [3]. Kluczowe jest uzyskanie dostępu do cech, które mają istotny wpływ na wynik naszych przewidywań. Im więcej cech znacząco odpowiadających za zakup danego przedmiotu uda się zidentyfikować, tym dokładniejsze będą prognozy.

## Metadane

Platforma Kaggle udostępnia cztery pliki: category\_tree.csv, events.csv, item\_properties\_part1.csv oraz item\_properties\_part2.csv. Pliki dotyczące właściwości przedmiotów zostały podzielone na dwie części ze względu na ich duży rozmiar. Dane zebrano z rzeczywistej witryny e-commerce i są one w formie surowej. Ze względu na poufność są one haszowane, co jednak nie wpływa znacząco na ich przydatność w uczeniu maszynowym. Poniżej przedstawiono szczegóły każdego pliku:

* **„events.csv”** zawiera kolumny takie jak:
  + „timestamp” – czas zdarzenia,
  + „visitorid” – ID odwiedzającego,
  + „event „– typ wydarzenia (np. przeglądanie, dodanie do koszyka, zakup),
  + „itemid” – ID przedmiotu,
  + „transaction” – ID transakcji (pozwala identyfikować przedmioty zakupione podczas jednej transakcji).
* **„item\_properties”** zawiera informacje o przedmiotach, takie jak:
  + „timestamp” – moment zmiany cechy przedmiotu,
  + „itemid” – identyfikator przedmiotu,
  + „property” – właściwości przedmiotu (może to być liczba opisująca cechę, albo słowo „categoryid” prowadzące do tabeli „category\_tree” lub „available” określające dostępność przedmiotu).
* **„category\_tree”** opisuje przynależność przedmiotów do kategorii. Jeśli w tabeli „item\_properties” pojawia się „categoryid”, wartość w kolumnie „value’ odnosi się do kolumny „categoryid” w tej tabeli. Wartość w kolumnie „parentid” (jeśli występuje) odnosi się do kategorii nadrzędnej.

W celu wstępnej analizy danych zaimportowano je do środowiska PyCharm. Oto wyniki podstawowej analizy:

* Liczba kategorii: **1 669**
* Liczba kategorii pokrewnych: **1 644**
* Liczba odwiedzających: **1 407 545**
* Liczba przedmiotów: **235 057**
* Liczba przeprowadzonych zakupów: **22 457**
* Liczba właściwości przedmiotów: **1 102**

## Reprezentacja

Warto przeanalizować poszczególne tabele, identyfikując najczęściej występujące cechy oraz ich średnie wartości. W tym celu można pracować w Pythonie, korzystając z biblioteki Pandas, lub utworzyć lokalną bazę danych.

Zalety pracy z bazami danych:

1. Zgodność ze standardami – określenie typów kolumn pomaga unikać błędów, takich jak dodawanie danych o niewłaściwym formacie.
2. Szybkość działania – silniki baz danych są zoptymalizowane pod kątem wydajnego przetwarzania zapytań SQL.
3. Łatwość użycia języka SQL – zapytania SQL umożliwiają intuicyjne przeszukiwanie danych bez konieczności pisania skomplikowanych algorytmów [4].

W projekcie wykorzystam bazę danych **MySQL**, której główne zadania obejmą:

* szybki dostęp do danych,
* przetwarzanie danych,
* prezentację wyników analiz.

Pracę z bazą danych rozpocząłem w programie MySQL Workbench – darmowym narzędziu firmy Oracle, oferującym przyjazny interfejs do zarządzania bazą danych. Po stworzeniu bazy danych należy przygotować odpowiednie tabele, załadować dane z plików CSV oraz wstawić wszystkie wiersze do tabeli.

Po utworzeniu tabel z danymi konieczne jest utworzenie odpowiednich indeksów dla kolumn, które będą wykorzystywane do filtrowania, łączenia z innymi kolumnami, grupowania czy sortowania. W indeksach system bazy danych zapisuje wewnętrzne przyporządkowania bloków danych do odpowiednich rekordów, dzięki czemu operacja wyszukiwania sprowadza się do przeszukania indeksu, a nie całej tabeli. Jest to szczególnie przydatne przy pracy z dużymi zbiorami danych, które są niezbędne do budowy bardziej złożonych modeli sieci neuronowych [5].

## Wyniki

Po stworzeniu i zoptymalizowaniu bazy danych, można przejść do analizy. Za pomocą tak przygotowanej bazy danych możemy sprawdzić najczęściej kupowane przedmioty.

|  |  |  |
| --- | --- | --- |
| Lp. | Id przedmiotu | Ilość sprzedaży |
| 1 | 461686 | 133 |
| 2 | 119736 | 97 |
| 3 | 213834 | 92 |
| 4 | 7943 | 46 |
| 5 | 312728 | 46 |
| 6 | 445351 | 45 |
| 7 | 48030 | 41 |
| 8 | 420960 | 38 |
| 9 | 248455 | 38 |
| 10 | 17478 | 37 |

***Tabela 1.*** *Ilość sprzedaży w zależności od przedmiotu.*

Najczęściej przeglądane przedmioty:

|  |  |  |
| --- | --- | --- |
| Lp. | Id przedmiotu | Ilość przeglądnięć |
| 1 | 187946 | 3410 |
| 2 | 461686 | 2539 |
| 3 | 5411 | 2325 |
| 4 | 370653 | 1854 |
| 5 | 219512 | 1740 |
| 6 | 298009 | 1642 |
| 7 | 96924 | 1633 |
| 8 | 309778 | 1565 |
| 9 | 257040 | 1531 |
| 10 | 384302 | 1528 |

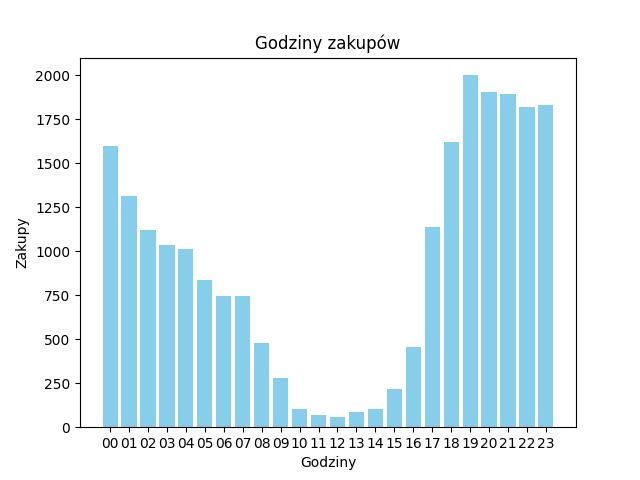
***Tabela 2.*** *Ilość przedmiotu w zależności od przedmiotu*

Najczęściej występujące właściwości:

|  |  |  |
| --- | --- | --- |
| Lp. | Właściwość | Ilość wystąpienia |
| 1 | 888 | 3000398 |
| 2 | NULL | 2291853 |
| 3 | 790 | 1790516 |
| 4 | 6 | 631471 |
| 5 | 283 | 597419 |
| 6 | 776 | 574220 |
| 7 | 678 | 481966 |
| 8 | 364 | 476486 |
| 9 | 202 | 448938 |
| 10 | 839 | 417239 |

***Tabela 3.*** *Najczęściej występujące właściwości*

Najpopularniejsze godziny zakupów:



***Wykres 1.*** *Zakupy w zależności od pory dnia*

Właściwości występujące razem:

|  |  |  |  |
| --- | --- | --- | --- |
| Lp. | Przedmiot 1 | Przedmiot 2 | Ilość wspólnych właściwości |
| 1 | 213834 | 445351 | 39 |
| 2 | 171878 | 461686 | 9 |
| 3 | 387697 | 428891 | 9 |
| 4 | 91547 | 98633 | 9 |
| 5 | 32581 | 461686 | 8 |

***Tabela 4.*** *Różnorodność właściwości*

Uzyskane wyniki wskazują, że 11 290 przedmiotów zostało zakupionych razem. Oznacza to, że współwystępowanie przedmiotów w transakcjach ma duże znaczenie i może dostarczyć istotnych informacji przy budowie systemu rekomendacji.

Średnia liczba przeglądnięć

Otrzymujemy tabelę, w której dla każdego użytkownika można odczytać liczbę przeglądnięć oraz liczbę zakupów.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Id Odwiedzającego | Przeglądnięcia przedmiotów | Wykonanych zakupów | Przeglądnięcia  /zakupy |
| 1 | 1150086 | 3809 | 532 | 7.1598 |
| 2 | 152963 | 1596 | 319 | 5.0031 |
| 3 | 530559 | 2204 | 259 | 8.5097 |
| 4 | 684514 | 1179 | 184 | 6.4076 |
| 5 | 861299 | 1080 | 183 | 5.9016 |

***Tabela 5.*** *Ilość przeglądnięć oraz zakupów dla każdego odwiedzającego*

Z Informacji o każdym użytkowniku możemy otrzymać średnią:

|  |  |  |
| --- | --- | --- |
| Średnie przeglądnięcia | Średnio zakupów | Średnie przeglądnięcia /średnio zakupów |
| 9,263 | 1,815 | 3.59932105 |

**Tabela 6.** Zależność zakupów od przeglądnięć

Po poznaniu struktury danych można przejść do wyboru odpowiednich cech, które zostaną wykorzystane w modelu predykcji. Niezbędne jest uwzględnienie wszystkich cech mających realny wpływ na decyzję zakupową konsumenta.

Najwięcej informacji na temat tego, czego konsument potrzebuje w określonym czasie, dostarczają jego poprzednie wyszukiwania. Przeglądane przedmioty wskazują na zainteresowanie użytkownika, jednak istnieje duże prawdopodobieństwo, że nie znalazł jeszcze produktu idealnie odpowiadającego jego potrzebom. Analizując historię przeglądnięć, możemy określić, jakich produktów szuka konsument i zasugerować mu te, które są najczęściej kupowane przez innych użytkowników.

Potrzebujemy zatem informacji o produktach przeglądanych przez użytkownika na krótko przed zakupem. W tym celu określimy przedmioty przeglądane około 3 godziny przed transakcją i utworzymy tabelę, w której każdy wiersz będzie reprezentował konkretny zakup. Kolumny będą zawierały informacje o produktach przeglądanych przed zakupem oraz ich cechy.

## Baza danych

Podczas tworzenia tabel w MySQL (lub innych bazach danych) należy przestrzegać zasad normalizacji, które pozwalają utrzymać porządek i spójność w strukturze bazy danych. Ich celem jest eliminacja anomalii danych spowodowanych redundancją. Zasady te, znane jako „Postacie normalne”, przedstawiają się następująco:

1. Pierwsza postać normalna (1NF) – Wszystkie kolumny muszą zawierać atomowe, nierozłączne dane. W komórkach bazy danych umieszczamy tylko jedną niepodzielną informację. Tabela powinna zawierać klucz podstawowy.
2. Druga postać normalna (2NF) – Wymaga spełnienia pierwszej postaci normalnej. Kolumny, które nie są częścią klucza podstawowego, muszą być zależne wyłącznie od tego klucza.
3. Trzecia postać normalna (3NF) – Wymaga spełnienia drugiej postaci normalnej. Należy usunąć przechodnie zależności między kolumnami. Oznacza to, że kolumny nie mogą zależeć częściowo od innych kolumn, które same zależą od klucza podstawowego [6].

Pierwsza tabela będzie zawierać informacje o zakupionych przedmiotach oraz użytkowniku, który dokonał zakupu. Tabela ta jest już w zasadzie gotowa – to tabela „events\_on\_item”, wystarczy ją odpowiednio przefiltrować. Z tej tabeli wybieramy:

* Unikalny identyfikator, który zostaje przepisany z tabeli „events\_on\_item”,
* Identyfikator użytkownika,
* Identyfikator zakupionego przedmiotu.

Czas zakupu ani identyfikator transakcji nie są potrzebne, ponieważ nie stanowią cech wpływających na zakupiony produkt ani wartości, które chcemy przewidzieć.

Kolejnym krokiem jest stworzenie tabeli zawierającej informacje o produktach przeglądanych przed zakupem. Do tej tabeli wstawiamy te cechy które będziemy chcieli użyć w modelu. Tabela ta będzie zawierać:

* Klucz główny odnoszący się do przeglądnięcia przedmiotu,
* Klucz obcy – odwołanie do tabeli „events\_on\_item”,
* Czas przeglądnięcia
* ID przeglądanego produktu przed zakupem
* Informację o kolejności przeglądanych przedmiotów w danej sesji.
* Typ wydarzenia na tym przedmiocie
* Dostępność przedmiotu w czasie zakupu
* Kategoria
* Pokrewna Kategoria

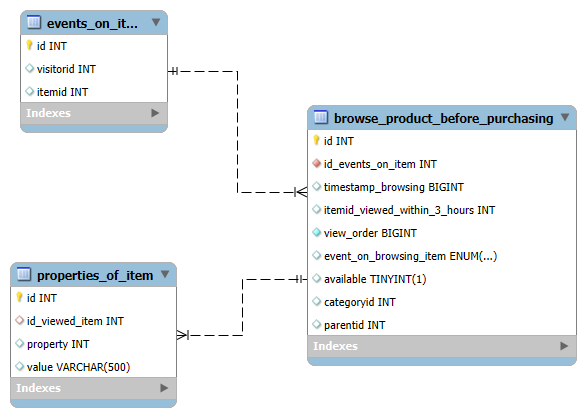
Ostatnia tabela zawiera informacje o właściwościach. A więc tabela będzie zawierała kolumny:

* Klucz główny
* Klucz obcy
* Nazwa właściwości
* Wartość właściwości

Dzięki takiemu zestawieniu mamy dane o zakupie danego przedmiotu oraz połączone informacje o właściwościach i kolejności przeglądanych produktów przed zakupem. Dane te są przedstawione w przejrzysty i znormalizowany sposób.  
 Uzupełnienie tabeli „ostatnio przeglądanych przedmiotów” wymagało wyodrębniania „kategorii” i „dostępności’ z tabeli „item properties”, ponieważ znajdowały się w kolumnie „value”, jeśli ich etykieta znajdowała się w kolumnie „properties", tak aby spełniało drugą zasadę normalną - każda cecha musi być zależna wyłącznie od klucza głównego

W przypadku kolumny „additional information”, jeżeli etykieta ma wartość „categoryid”, oznacza to, że kolumna „value” zawiera identyfikator kategorii z tabeli „category tree”. Tabela ta pozwala nam odczytać zarówno kategorię główną, jak i kategorię pokrewną, które mogą okazać się przydatne w modelu predykcji. Tworzymy zapytanie przypisujące przedmiotom: odpowiednią nazwę kategorii oraz kategorię pokrewną z tabeli „category tree”. Wszystkie dane zostały tak przefiltrowane aby zawierały informacje o wykonywanych czynnościach 3 godziny przed zakupem.

Po utworzeniu struktury bazy w MySQL Workbench możemy wygenerować schemat bazy. Dla tak utworzonej struktury, diagram prezentuje się następująco:



***Rysunek 1.*** *Schemat bazy danych. Żółty klucz – klucz główny tabeli, czerwony rąb – klucz obcy, niebieski rąb – kolumna tabeli, wychodząca strzałka – wskazuje do jakiej tabeli odnosi się klucz obcy.*

# Przygotowanie danych:

## Zmienne numeryczne

Jeśli mamy już wybrane dane i wiemy, które cechy chcemy przekazać do modelu oraz które mają realny wpływ na wartość przewidywaną, pozostaje kwestia wyboru sposobu ich przekazania.

W przypadku zmiennych liczbowych przekształcenie polega na przeskalowaniu danych w taki sposób, aby ich średnia wynosiła 0, a odchylenie standardowe 1. Proces ten nosi nazwę normalizacji. Gdybyśmy nie zastosowali normalizacji, różnice w wartościach numerycznych mogłyby być znaczne, co miałoby duży wpływ na proces uczenia. Na przykład, wartości różniące się w przedziale 1–10, byłyby znacznie mniej istotne w porównaniu do wartości w przedziale 1000–10 000. Taka sytuacja prowadziłaby do wolniejszej zbieżności i znacząco wpływałaby na wagi modelu. Normalizacja zmniejsza ryzyko powstania gradientów o zbyt małym lub zbyt dużym wpływie na proces uczenia i aktualizację parametrów modelu. Dzięki temu wszystkie cechy mają porównywalny wpływ na wynik.

Normalizację, w której modyfikujemy wartości tak, aby ich średnia wynosiła 0, a odchylenie standardowe 1, nazywamy standaryzacją i używamy w zmiennych numerycznych. Uzyskuje się ją za pomocą prostej operacji:

***Wzór 1.*** *Postać znormalizowana,* - oryginalna wartość cechy,- średnia wartość cechy, – odchylenie standardowe cechy,

## Zmienne kategoryczne

W przypadku zmiennych kategorycznych, czyli takich, w których liczba możliwych wartości jest z góry ustalona, konieczne jest odpowiednie zakodowanie danych, aby model mógł je poprawnie przetwarzać. Jedną z najczęściej stosowanych metod jest kodowanie "jeden do wielu", nazywane również "gorącą jedynką". Polega ono na reprezentowaniu każdej kategorii przez oddzielny bit, który wskazuje, czy dana obserwacja należy do tej kategorii, czy też nie. Metoda ta sprawdza się dobrze w przypadku zmiennych kategorycznych o liczbie kategorii nieprzekraczającej 20.

Jeśli liczba kategorii jest większa, warto przeanalizować częstość ich występowania. W przypadku kategorii rzadko pojawiających się sensowne może być ich usunięcie, podobnie jak postępuje się z wartościami odstającymi. Kategoria o bardzo małej częstości występowania ma niewielki wpływ na proces uczenia modelu, a nawet może pogorszyć wyniki.

Za pomocą prostego zapytania SQL można ustalić, że liczba unikalnych wartości zmiennej „visitorid” wynosi 11 719. Jest to bardzo duża liczba, co sprawia, że zastosowanie kodowania "jeden do wielu" byłoby nieefektywne ze względu na ogromną ilość pamięci potrzebnej do przechowywania każdej kategorii.

Pierwszym krokiem jest analiza powtarzalności wartości. Z danych wynika, że spośród 22 457 rekordów wielu odwiedzających dokonało zakupu tylko raz. Takie dane pojedynczo występujące nie dostarczają wystarczającej ilości informacji, aby sieć neuronowa mogła nauczyć się ich cech. Dlatego zbiory danych, które występują tylko raz, zostały przypisane do wartości 0. Następnie pozostałe wartości zostały przenumerowane, aby ich zakres wynosił od 0 do 2 576. Taka operacja pozwoliła znacząco oczyścić dane i zmniejszyć liczbę kategorii.

Niemniej jednak liczba kategorii pozostaje zbyt duża, aby zastosować kodowanie "gorącą jedynką". W takim przypadku lepszym rozwiązaniem jest kodowanie za pomocą osadzeń (ang. *embeddings*). Metoda ta polega na mapowaniu każdej kategorii na wektor o mniejszym wymiarze. Jest to podejście często stosowane w modelach przetwarzających tekst lub słowa. W TensorFlow można to zrealizować za pomocą dedykowanej warstwy (Embeding), która uczy się optymalnych wartości dla poszczególnych kategorii.

Dobór wymiaru wektora zależy od liczby kategorii, jednak nie ma jasnej regułu sprawdzającej się w każdym przypadku. Przyjęto stosowanie pierwiastka 4 stopnia do określenia wstępnego wymiaru. W omawianym przypadku przyjęto wektor 10-wymiarowy, co pozwoliło na efektywne przetwarzanie danych przy zachowaniu ich najważniejszych cech.

Natomiast w przypadku zmienne określającej wydarzenie (jedno z 3 wydarzeń) doskonale sprawdzi się metoda „gorącej jedynki”.

Kolejną analizowaną cechą jest „categoryid”. Liczba kategorii dla tej cechy wynosi 844. Wśród nich zidentyfikowano 41 wartości, które pojawiają się tylko raz lub dwa razy. Te wartości uznano za odstające i usunięto z danych. Następnie wszystkie pozostałe wartości w kolumnie zostały posortowane od najmniejszej do największej, a każdej z nich przypisano unikalny numer porządkowy. Podobnie jak w przypadku innych cech kategorycznych, zastosowano osadzenia (ang. embeddings), aby efektywnie przekazać informacje o kategoriach do modelu

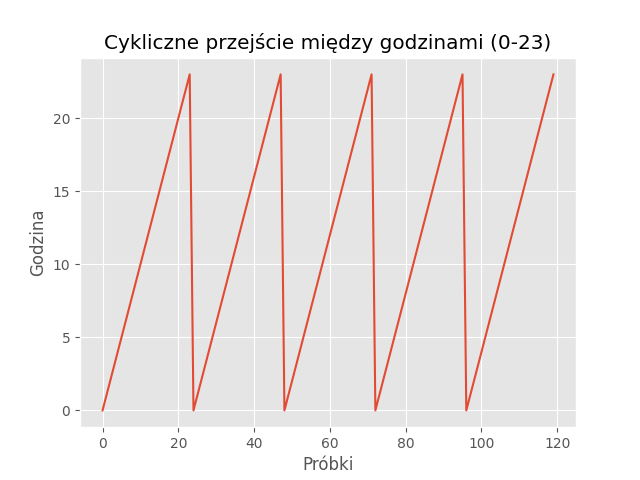
Analogiczny proces zastosowano do danych dotyczących kategorii pokrewnych. Liczba tych kategorii wynosi 252, z czego 10 występuje w zbiorze mniej niż dwa razy. Dane te również zostały uznane za odstające i usunięte. Pozostałym wartościom przypisano numery od 0 w górę, przy czym wartość 0 oznacza brak cechy.

## Dane cykliczne:

Kolejną cechą, którą należy uwzględnić w modelu, jest czas dokonania przeglądania. Może on mieć istotne znaczenie, ponieważ w zależności od momentu interakcji można przewidywać różne zachowania użytkowników. Na przykład, przeglądanie produktów w okresie świątecznym może wskazywać na większe prawdopodobieństwo zakupu przedmiotów związanych z tym czasem. Z kolei w okresie letnim użytkownicy częściej wybierają produkty sezonowe, takie jak odzież letnia.

Dane dotyczące czasu w zbiorze są zapisane w formacie „Unix timestamp” (przykładowa wartość: 1433221896102). Format ten zazwyczaj przedstawia liczbę sekund, które upłynęły od 1 stycznia 1970 r. W tym przypadku wartości są wyrażone w milisekundach, co wymaga ich przeskalowania przez podzielenie przez 1 000. Wysyłanie daty w formacie „Unix timestamp” bezpośrednio do sieci neuronowej nie jest dobrym rozwiązaniem, ponieważ nie pozwala na wyodrębnienie kluczowych informacji, takich jak sezonowość czy cykliczność danych. Aby poprawnie przekazać te dane do modelu, konieczne jest ich rozdzielenie na składniki zmieniające się co pewien okres czasu.

W MySQL można w prosty sposób skonwertować dane na składowe, takie jak: rok, miesiąc, dzień, dzień tygodnia, godzina, minuta oraz sekunda. Dzień tygodnia również warto uwzględnić, ponieważ np. w weekendy użytkownicy częściej przeglądają produkty związane z wypoczynkiem czy rozrywką. Pojawia się jednak problem związany z cyklicznością danych — szczególnie w przypadku wartości granicznych. Sieć neuronowa nie dostrzeże podobieństwa między godziną 23:59 a 00:00, traktując je jako zupełnie różne i skrajne wartości. Dobrze to przedstawia wykres poniżej:



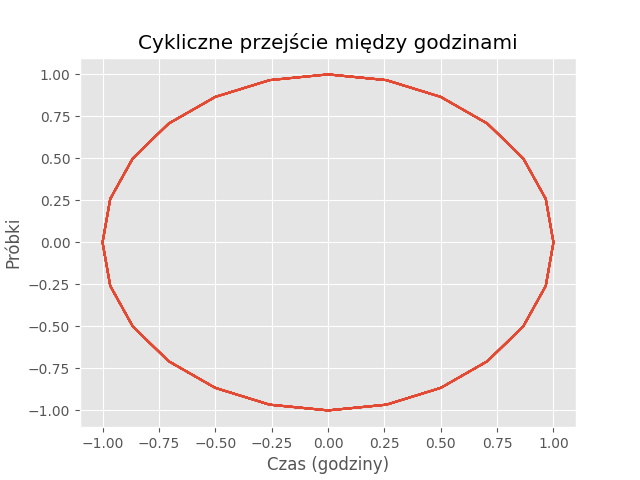
***Wykres 2.*** *Cykliczne przejścia niezmodyfikowanych danych.*

Wartości gwałtownie się zmieniają, co może spowodować pogorszenie wyników sieci. Rozwiązaniem tego problemu, jest zastosowanie metody kodowanie cech cyklicznych. Przeprowadzenie transformacji sinusoidalnej i kosinusoidalnej cechy przez wzory [7]:

***Wzór 2.*** *Transformacja sinusoidalna, x – wartość wejściowa.*

***Wzór 3.*** *Transformacja cosinusowa, x – wartość wejściowa.*

Dodatkowo, otrzymujemy również godziny przeskalowane do wartości od -1 do 1, co jest bardzo pożądane w danych trafiających do modelu. Nowe wartości przedstawione na wykresie zależności od prezentuje się w następujący sposób:



***Wykres 3.*** *Cykliczne przejścia zmodyfikowanych danych.*

## Dane Tekstowe

Każdemu z przeglądanych produktów odpowiada cała seria właściwości opisujących przedmiot. Każda iteracja zapisana w tabeli „ostatnio przeglądanych produktów” ma osobny ciąg właściwości, nawet jeśli jest to ten sam przedmiot, ponieważ właściwości przedmiotów zmieniały się w czasie.

Słowa w tych danych zostały już wstępnie haszowane. Wszystkie wartości liczbowe zostały oznaczone znakiem „n” na początku i mają dokładność do trzech miejsc po przecinku, np. „5” reprezentuje „n5.000”, a „-3.67584” reprezentuje „n-3.675”. Wszystkie słowa w wartościach tekstowych zostały znormalizowane (procedura stemmingu) i haszowane. Na przykład tekst „Hello world 2017!” został przekształcony na „24214 44214 n2017.000”. Konieczne jest więc oddzielenie wszystkich składowych ciągu, a następnie oddzielenie słów od liczb. Liczby w osobnych kolumnach zostały zamienione na typ „float”. Słowa zostały również umieszczone w osobnych kolumnach zgodnie z kolejnością występowania. Słowa trafiły do warstwy osadzeń, natomiast liczby po znormalizowaniu do wejścia modelu.

## Algorytmy stemmingu

Proces stemmingu polega na redukcji odmiennych (często pochodnych) form słów do ich rdzenia. Oznacza to, że słowa o podobnym rdzeniu, lecz różniące się prefiksami, sufiksami lub przekształceniami, zostaną zastąpione jedynie rdzeniem. Upraszcza to tekst, jednocześnie zachowując główne znaczenie słów. Przetwarzanie bez tej operacji wymagałoby znacznie większych zasobów obliczeniowych i bardziej zaawansowanej sieci. Na nasze potrzeby takie uproszczenie jest wystarczające.

Następnie słowa są haszowane na liczby, ponieważ sieć neuronowa operuje wyłącznie na wartościach zmiennoprzecinkowych. Tak przetworzone dane są wysyłane do warstwy osadzeń.

Jednym z popularnych algorytmów stemmingu jest wykorzystanie tablicy LUT (ang. Lookup Table). Polega to na stworzeniu prostego słownika, który każdemu kluczowi (słowu) przypisuje odpowiedni rdzeń. Takie podejście jest wydajne, ponieważ zamienia skomplikowane obliczenia na szybkie wyszukiwanie w słowniku. Dodatkowym atutem jest łatwa obsługa wyjątków – wystarczy dodać nowy klucz do słownika.

Do wad tego podejścia należy konieczność wymienienia wszystkich (często bardzo podobnych) przypadków oraz brak obsługi nowych, nieznanych słów. W językach o dużej fleksji tabela może być ogromna, co sugeruje zastosowanie bardziej dynamicznych algorytmów, np. takich, które bazują na określonych algorytmach konwersji słów na rdzenie.

## Zbiór treningowy i walidacyjny

Ostatnim krokiem przed wprowadzeniem danych do modelu jest ich podział. Dane, których będziemy potrzebować do procesu tworzenia modelu, to: dane treningowe, dane walidacyjne oraz dane testowe. Dane treningowe, jak sama nazwa wskazuje, będą używane do trenowania naszego modelu. To na nich, przez propagację wsteczną, będą obliczane nowe wagi i przesunięcia modelu. Dane walidacyjne będą używane do testowania modelu podczas uczenia. Pozwolą nam one dobrać odpowiednie parametry oraz modyfikować model w zależności od otrzymanych wyników. Będziemy mogli np. sprawdzić, czy nie występuje przeuczenie, ponieważ będziemy mieli informacje o tym, jak wygląda dokładność danych treningowych i testowych. Jeśli różnica będzie znacząca, da to nam informację, czy nie należy zmniejszyć liczby neuronów w sieci głębokiej lub zastosować regularyzację. W sieci neuronowej takie dostrajanie modelu jest bardzo istotne, ponieważ nie ma jasnych reguł określających, w jakich warunkach dana sieć sprawdzi się najlepiej. Ostatnie dane to dane testowe, które nie wchodzą w skład ani danych walidacyjnych, ani treningowych. Używane są one do ostatecznego przetestowania modelu. Dają nam one finalną informację o jakości i skuteczności naszego modelu.

# Struktura sieci neuronowej

## Rekurencyjna sieć

Rekurencja to sytuacja, w której funkcja wywołuje samą siebie. W prostej funkcji rekurencyjnej, która wykonuje podstawową operację, np. podnosi liczbę do potęgi, wprowadzamy pierwszy argument, który przechodzi przez funkcję i trafia z powrotem do tej samej funkcji, wykonując tę samą operację.

W takim procesie warto zwrócić uwagę na wpływ pierwszej liczby na wynik. W drugim obiegu pętli ten wpływ będzie bardzo duży, ponieważ liczba została już podniesiona do potęgi i wynik został przekazany dalej. Oznacza to, że początkowa wartość znacząco wpływa na kolejne obliczenia. Jednak w sytuacji, gdy rozpatrujemy wynik po większej liczbie iteracji, ten wpływ maleje, ponieważ pierwotna liczba została już wielokrotnie przetworzona, a wynik stanowi jedynie niewielką część wartości początkowej.

Podobną zależność obserwujemy w przypadku rozumienia tekstu czy sekwencji przez sieci neuronowe. W wielu sytuacjach poszczególne słowa w tekście mają duży wpływ na wystąpienie następnych i są z nimi powiązane semantycznie. Na przykład w wyrażeniu „zielona choinka”, słowo „zielona” ma znaczący wpływ, ponieważ określa, jaka jest choinka. Takie podejście do przetwarzania danych w sieciach neuronowych nazywa się RNN (ang. Recurrent Neural Network) [8]. Prosty schemat działania takiej sieci można zobaczyć na obrazie:Obraz zawierający diagram, zrzut ekranu

Opis wygenerowany automatycznie

***Rysunek 2.*** *Rozwinięcie w czasie rekurencyjnej sieci neuronowej, h – wewnętrzny stan neuronu, x – wejście neuronu, o – wyjście neuronu, u – wagi powiązane z wejściem, w – wagi powiązane z wyjściem, v – połączenie rekurencyjne*

Widzimy tutaj rekurencyjne wywołania operacji jednego neuronu. Wartość X to wejściowy parametr, natomiast Q to wynik uzyskany w danej iteracji. Kolejne wartości V są przekazywane do głębszych wywołań. Jednak wadą takiego podejścia jest znikomy wpływ słów znacząco oddalonych od siebie w strukturze zdania. Na przykład w zdaniu: „Nigdy wcześniej czegoś takiego nie widziałem”, słowa „Nigdy” oraz „nie widziałem” tworzą pełne znaczenie, ale są oddzielone trzema słowami.

Rozwiązaniem problemu w takich przypadkach jest zastosowanie warstwy LSTM (Long Short-Term Memory). Taka sieć usprawnia działanie RNN poprzez uwzględnienie "stanu komórki". Taka pamięć działa w dwóch kierunkach, dzięki czemu może analizować tekst na dwa sposoby: od przodu i od tyłu.

W przypadku LSTM model wyglądałby podobnie jak w przypadku RNN, jednak komunikacja byłaby dwukierunkowa z uwzględnieniem "stanu komórki". Takie rozwiązanie idealnie sprawdza się w przetwarzaniu języka naturalnego oraz danych sekwencyjnych. Dlatego za naszymi wartościami czasowymi, umieścimy właśnie warstwę LSTM. Natomiast za warstwą LSTM umieścimy dodatkową warstwę głęboką, która będzie przetwarzała wartości wyjściowe z warstwy rekurencyjnej z pamięcią długoterminową [9].

## Łączenie zależności

Pierwszą rzeczą, o której powinniśmy pamiętać podczas projektowania sieci neuronowej, jest łączenie najbardziej podstawowych zależności na najgłębszym poziomie. Oznacza to, że najbardziej powiązane ze sobą cechy powinny być łączone jako pierwsze. Nasz model będzie projektowany z użyciem biblioteki TensorFlow, a więc do tego procesu użyjemy funkcji Concatenate: z pakietu „tensorflow.keras”. Funkcja ta służy do łączenia dwóch tensorów w jeden.

Tensor to uogólnienie wektorów i macierzy do potencjalnie wyższych wymiarów. Wewnętrznie TensorFlow reprezentuje tensory jako n-wymiarowe tablice bazowych typów danych. Każdy element w tensorze ma ten sam typ danych, a typ danych jest zawsze znany.

Podczas analizy tekstowego opisu właściwości produktu przeglądanego w sklepie oddzieliłem słowa od liczb, aby liczby mogły być poddane procesowi normalizacji i przedstawione w sposób bardziej zrozumiały dla modelu. Natomiast są to dane wejściowe, które powinny zostać połączone w pierwszej kolejności. Takie połączenie wskazuje, że te dane są od siebie zależne i mają dużo wspólnego. Jednak aby tego dokonać, konieczne jest najpierw spłaszczenie danych do tego samego rozmiaru. Próba połączenia wejść o różnych rozmiarach spowoduje błąd.

Po każdym wejściu danych kategorycznych, w którym decydujemy się nie używać kodowania "gorącą jedynką", dodajemy warstwę osadzeń (embedding), która będzie przekształcała oryginalny numer wejściowy na wektor osadzeń reprezentujący wartość kategoryczną w mniejszym wektorze.

Następnie łączymy wszystkie warstwy według zależności, tzn.:

* Wszystkie wartości związane z czasem (miesiąc, dzień, godzina itd.) są od siebie zależne, więc łączymy je ze sobą.
* Następnie łączymy je z kolejną cechą, np. informacją, co wydarzyło się o tej godzinie (dostępność, rodzaj wydarzenia itd.).

Po każdym wprowadzeniu danych dołożyłem jeszcze podstawową warstwę neuronów. Tak połączone wartości dają nam jedno wejście, które trafia do warstwy głębokiej. Liczba neuronów w warstwie wejściowej jest dobierana eksperymentalnie — nie ma jasno określonych zasad. Przyjmuje się jednak, że w pierwszej warstwie głębokiej powinno być około dwa razy więcej neuronów niż liczba wszystkich elementów cech wejściowych.

Kiedy zdecydujemy się na zbyt małą liczbę neuronów w warstwie głębokiej, nasz model może być niedouczony, co skutkuje gorszą dokładnością. Natomiast w przypadku zbyt dużej liczby neuronów w warstwach głębokich istnieje ryzyko zbyt dużego dopasowania modelu (overfitting).

Overfitting to zjawisko w statystyce, które występuje, gdy model statystyczny ma zbyt wiele parametrów w stosunku do rozmiaru danych, na podstawie których został skonstruowany. W uczeniu maszynowym oznacza to, że model doskonale dopasowuje się do danych treningowych, ale daje gorsze wyniki na nowych danych, których wcześniej nie widział [10].

## Warstwa TimeDistributed

W bibliotece TensorFlow warstwa TimeDistributed jest używana do przetwarzania sekwencji danych, wykonując tę samą operację na różnych zestawach danych w każdej iteracji. Jest to przydatne, gdy chcemy zastosować tę samą operację do każdej jednostkowej części sekwencji, na przykład dla każdej klatki filmu lub każdego tokena w tekście. Warstwa ta zapewnia, że operacja jest stosowana niezależnie do każdego elementu w sekwencji.

## Funkcja straty:

Jak oceniać zwracaną wartość przez nasz model? Do tego właśnie używamy funkcję straty, która dla każdej odpowiedzi zwraca wartość, określającą, jak daleko jesteśmy od poprawnego przewidywania. W zależności od tego, co nasz model przewiduje, używa się następujących funkcji:

* Średni błąd kwadratowy (MSE). Jest to suma kwadratów różnic między wartościami w wektorze predykcji a wektorem prawdy, przeznaczona dla predykcji wartości ciągłych. Idealnie sprawdza się w przypadku problemu regresji. Wzór tej funkcji jest następujący:

***Wzór 4.*** *Średniokwadratowa funkcja błędu, N – ilość wartości w wektorze wyjściowym,*

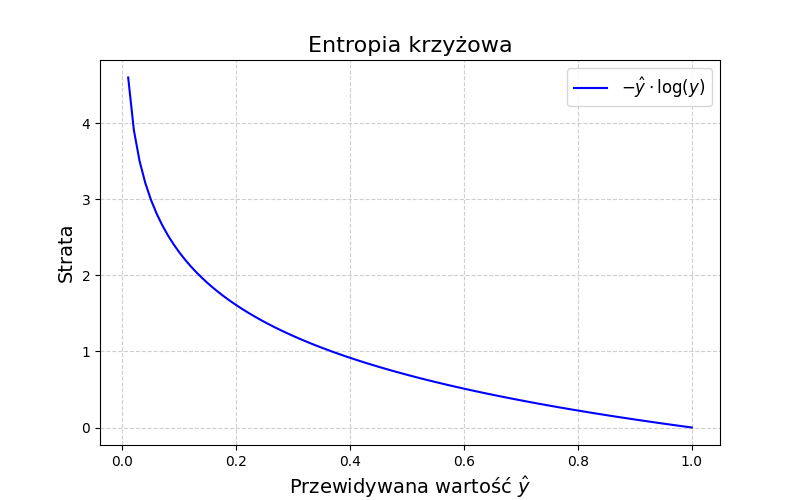
*- wartość otrzymana, - wartość prawidłowa*

* Funkcja straty entropii krzyżowej. Jest przeznaczona do danych kategorycznych. Wyznacza ona różnice między prawdopodobieństwem wystąpienia danej kategorii a etykietą. Do wykorzystania tej funkcji konieczne jest przedstawienie etykiet za pomocą kodowania „one-hot”, czyli wektora z jedną jedynką określającą, jaką klasę reprezentuje etykieta. Tą funkcję wykorzystałem w moim projekcie.

***Wzór 5.*** *Funkcja entropii krzyżowej, N – ilość wartości w wektorze wyjściowym,*

*- wartość otrzymana, - wartość prawidłowa*

:



***Wykres 4****. Entropia krzyżowa dla wartości od 0 do 1*

* Średni bezwzględny błąd procentowy. Również jest przeznaczony do problemów regresji, natomiast uwzględnia on skalę, w jakiej błąd jest popełniony. Jeśli mamy np. wartość dokładną 500, a nasz model przewidzi 505, jest to mniej istotna różnica niż w przypadku wartości np. 5 i 10. Jest to rzadko używana funkcja straty, a jej wzór prezentuje się następująco:

***Wzór 6.*** *Średni bezwzględny błąd procentowy, N – ilość wartości w wektorze wyjściowym, - wartość otrzymana, - wartość prawidłowa* [11]

## Metryki dokładności:

Kolejnym ważnym parametrem, jaki należy wybrać do modelu, jest sposób mierzenia poprawności otrzymywanych wyników. Do tego używamy metryki.

Przestrzeń metryczna to zbiór z zadaną na nim metryką, tj. funkcją, która określa odległość między każdą parą elementów tego zbioru[12].

W naszym przykładzie wybieramy metrykę dokładności z pakietu TensorFlow. Dokładność mierzy, ile próbek pasuje do przewidywań, a następnie dzieli je przez wszystkie próby wykonywane przez nasz model. Innymi popularnymi metrykami są:

* Precyzja – metryka tworzy dwie zmienne: „true positives” (liczba poprawnie zgadniętych pozytywnych wyników) oraz „false positives” (liczba błędnie przewidzianych pozytywnych wyników). Następnie dzieli „true positives” przez sumę tych dwóch zmiennych. Daje to informację, jaki procent pozytywnych wyników został wykryty.
* Recall – dzieli zmienną „true positives” przez sumę „true positives” oraz „false negatives”, dając informację o procencie wszystkich rzeczywistych pozytywnych wyników, które zostały poprawnie wykryte.
* F1 Score – połączenie precyzji oraz recall.

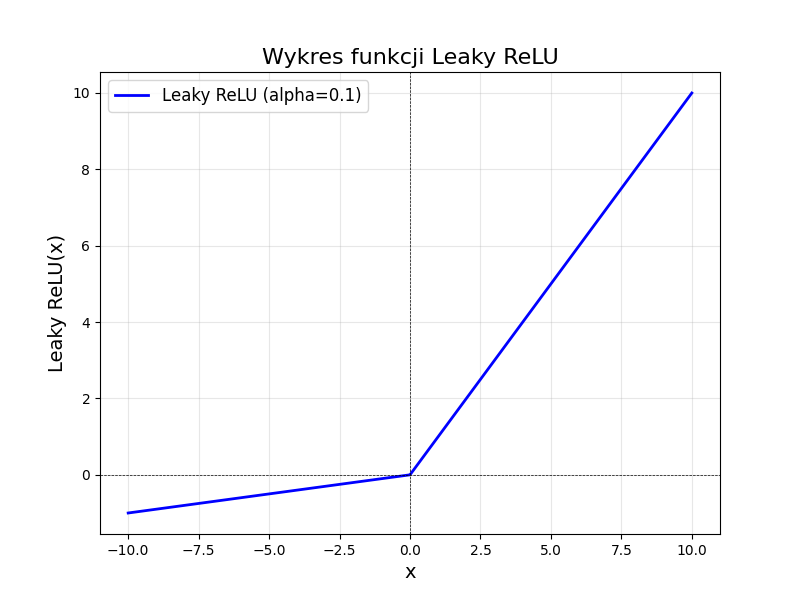
***Wzór 6.*** *Średni bezwzględny błąd procentowy.*

## Funkcje aktywacji:

Kolejnym elementem, na który warto zwrócić uwagę, jest funkcja aktywacji. Najczęściej używaną funkcją aktywacji jest funkcja ReLU, w której dla argumentów od -∞ do 0 wartość zwracana jest równa 0, natomiast dla argumentów większych od 0 jej wartość rośnie liniowo. Jednym z problemów sieci neuronowej jest problem zanikających/eksplodujących gradientów. Ponieważ propagacja wsteczna polega na obliczeniu straty na wyjściu modelu i wyznaczeniu wartości gradientu, który następnie jest przekazywany do wcześniejszych warstw przez propagację wsteczną, pojawia się sytuacja, w której gradient jest redukowany przez przechodzenie do kolejnych warstw, co sprawia, że dla początkowych warstw gradient jest prawie niezauważalny. W niektórych sytuacjach możemy napotkać na przeciwny problem, niektóre połączenia są bardziej znaczące, co sprawia, że ich gradient rośnie do znacznych wartości.

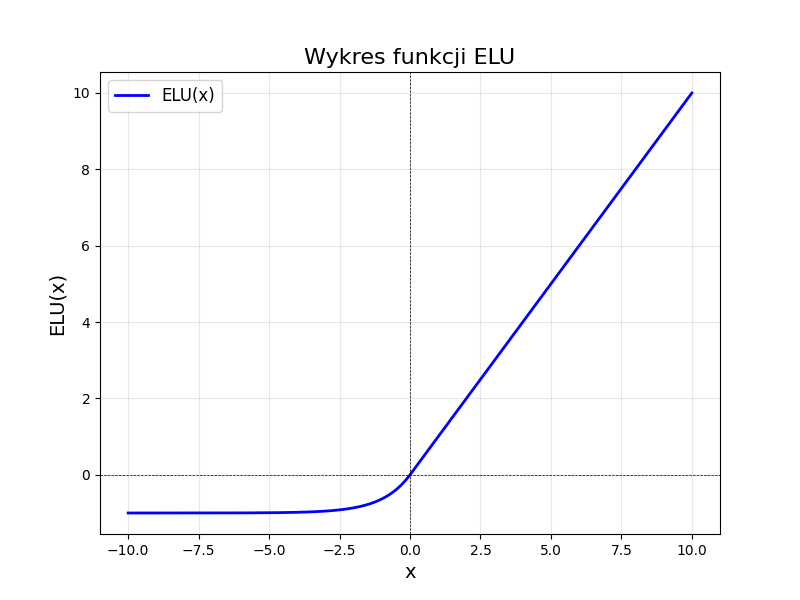
W 2010 roku Xavier Glorot oraz Yoshua Bengio w swojej pracy zajmującej się zjawiskiem zanikających/eksplodujących gradientów doszli do wniosku, że jest to częściowo spowodowane między innymi funkcją aktywacji. Funkcja sigmoidalna, która była często używana do tamtej pory, nasycała się w wartościach argumentów znacząco odstających od 0. Jednym ze sposobów poprawy wyników było zastosowanie funkcji aktywacji ReLU, która nie nasycała się dla wartości powyżej 0.

Z kolei jednak, funkcja ReLU cierpi na problem „śmierci neuronów”. Funkcja dla wartości poniżej 0 jest stała i równa 0, co powoduje, że niektóre neurony stale osiągają wartość 0, co prowadzi do całkowitego zaniku przekazywanego przez nie sygnału. Pewnym rozwiązaniem jest zastosowanie „przeciekającej” funkcji aktywacji ReLU, która wygląda następująco:



***Wykres 5.*** *Przeciekająca funkcja ReLU.*

Takie rozwiązanie sprawia, że neurony mogą zostać wyłączone, jednak przy odpowiednich zmieniających się danych podczas uczenia istnieje szansa na „przebudzenie” takich neuronów. Parametr nachylenia funkcji dla ujemnych argumentów (alpha) możemy dobrać eksperymentalnie lub jako wartość uczącą się. Funkcja, która jeszcze lepiej sprawdza się w takich przypadkach jest jednostka wykładniczo-liniowa (Exponential Linear Unit – ELU). Funkcja ELU cechuje się jeszcze lepszymi wynikami, ponieważ jest gładka w każdym punkcie, co przyspiesza metodę gradientu prostego, oraz przyjmuje wartości mniejsze od 0, co zapobiega zanikającym gradientom[13].



**Wykres 6.** Funkcja aktywacji ELU.

### Softmax

Jest to funkcja aktywacji używana zazwyczaj w predykcji kategorii do ostatniej warstwy, gdyż zwraca ona wartość prawdopodobieństwa należącą do danej klasy, sumując się do 1. Sprawdza się idealnie do problemów klasyfikacji, a warstwa wykorzystująca tę funkcję zostaje umieszczona jako wyjściowa z modelu. Podczas uczenia od tego wektora odejmowany jest wektor etykiet z oznaczeniem przynależności do danej kategorii, a na tej podstawie obliczany jest gradient.

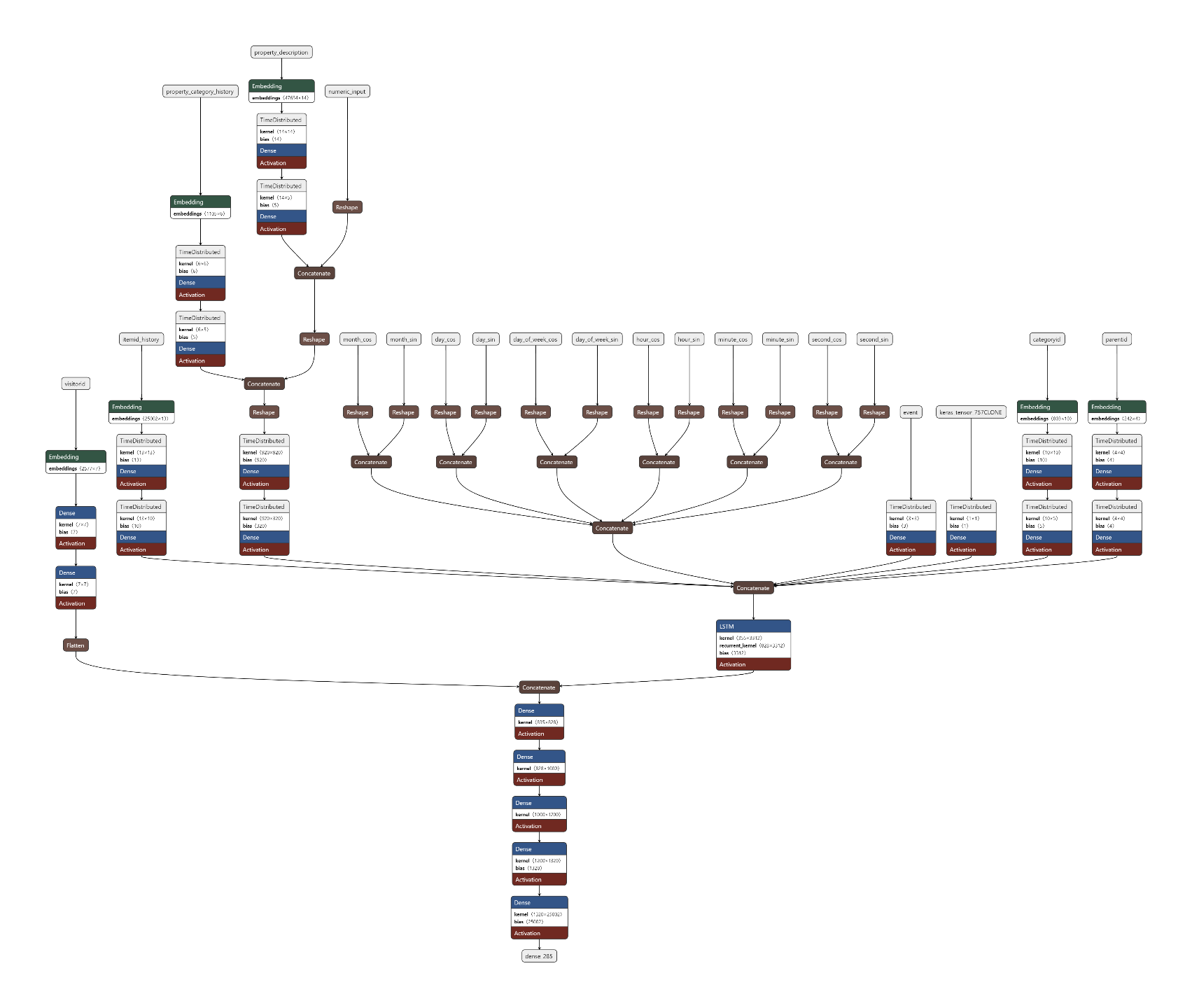
## Optymalizator:

Ostatnim wyborem, jaki trzeba podjąć przy tworzeniu modelu, jest optymalizator, który pełni rolę nadzorowania procesu uczenia oraz aktualizacji wag poszczególnych warstw na podstawie wcześniej obliczonych gradientów. Niektóre optymalizatory w trakcie uczenia dostosowują współczynnik uczenia, aby uzyskać lepsze wyniki.

Do tego projektu użyłem uniwersalnego optymalizatora ADAM, który cechuje się szybką zbieżnością i stabilnością. Jest on często używany oraz łączy zalety algorytmów SGD oraz RMSProp. Jego domyślnym hiper parametrem jest współczynnik uczenia równy 0.001.

Inne stosowane optymalizatory to:

* RMSProp (Root Mean Square Propagation) – dynamicznie dostosowuje współczynnik uczenia
* SGD (Stochastic Gradient Descent) – stały lub dynamiczny współczynnik uczenia. • Adagrad – efektywny dla rzadkich danych.
* Adadelta – ogranicza sumę kwadratu gradientu, co poprawia elastyczność modelu.



***Rysunek 3.*** *Schemat sieci neuronowej*

# Analiza i poprawa wyników sieci:

Mimo że rozumiemy reguły uczenia, proces trenowania oraz ideę, przez bardzo duże skomplikowanie całej struktury i połączeń między neuronami, nie jesteśmy w stanie w pełni zrozumieć działania AI ani zdefiniować jasnych reguł odnoszących się do podobnych problemów rozwiązywanych przez sieć. Bardzo ważną rolę odgrywa więc proces testowania i poprawy niektórych jego parametrów.

Każdy zestaw danych jest określoną ilość razy trenowany przez sieć, a nazwa takiego jednego przejścia przez sieć nosi nazwę epoki. W otrzymanym podsumowaniu naszego trenowania możemy w łatwy sposób przeanalizować, ile wynosił nasz błąd (wartość określona przez dobraną przez nas funkcję straty) oraz dokładność (metryka określająca jakość modelu przez ocenę jego predykcji) po każdej epoce. W ten sposób otrzymujemy wykresy dokładności oraz straty po każdej epoce zarówno dla danych treningowych, jak i walidacyjnych.

## Overfiting

Po wytrenowaniu naszego modelu możemy często zaobserwować, że po pewnym okresie uczenia wyniki dokładności lub funkcji straty dla danych walidacyjnych zaczynają się pogarszać, mimo że dokładność zbioru treningowego cały czas rośnie. Takie zjawisko nosi nazwę overfittingu, czyli nadmiernego dopasowania. Model po pewnym czasie przestaje odwzorowywać regułę, jaką przewidują etykiety, ale zaczyna dostosowywać się do danych treningowych, zamiast szukać ogólnych reguł. Jest to bardzo często spotykane zjawisko podczas uczenia maszynowego. Istnieje wiele metod radzenia sobie z przetrenowaniem, większość parametrów należy dobierać empirycznie. Wyróżniamy następujące sposoby eliminowania przeuczenia:

* Zmniejszenie współczynnika uczenia – spowalnia uczenie, sprawia, że wagi i biasy neuronów wolniej dostosowują się do funkcji straty.
* Zastosowanie algorytmu zatrzymania trenowania, gdy wyniki przestają się poprawiać – specjalna metoda przerywająca uczenie w momencie braku poprawy dokładności.
* Regularyzacja - dzieli się na kilka metod, które wpływają na sposób poprawy parametrów modelu.

## Regularyzacja L1 oraz L2

Opiera się na modyfikacji wartości funkcji kosztu dla danej warstwy. Regularyzacja L1 (Lasso regression) dodaje do funkcji kosztu wartość bezwzględną wag poszczególnych neuronów w warstwie. Taki zabieg zapobiega lawinowemu wzrostowi gradientu, ponieważ poprzez „karanie” naszej warstwy za zbyt duże wagi, zapobiegamy ich nadmiernemu wzrostowi. Dodatkowo, regularyzacja L1 jest korzystniejsza, kiedy przypuszczamy, że pewne cechy odgrywają większą rolę w predykcji niż inne. W L1 preferowane są te wagi, które mają większy wpływ na predykcję, poprzez nawet wyzerowanie innych wag. Do kontroli nad taką operacją służy parametr λ. Zazwyczaj jego wartość oscyluje w okolicach 0.01.

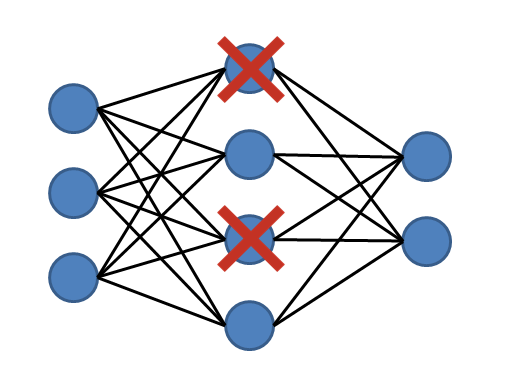
***Wzór 7.*** *Funkcja straty z regularyzacją L1, - wartość wejściowa, – wartość prawidłowa, – współczynnik regularyzacji, - waga*

W przypadku regularyzacji L2 (Regresja grzbietowa, Ridge regression) wagi są bardziej hamowane w większych wartościach, ale nie są całkowicie eliminowane. Prowadzi to do bardziej stabilnego i równomiernego rozłożenia wag w warstwie. Regularyzacja L2 również skutecznie ogranicza lawinowy wzrost wag co zmniejsza ryzyko przeuczenia.

***Wzór 8.*** *Funkcja straty z regularyzacją L2, - wartość wejściowa, – wartość prawidłowa, – współczynnik regularyzacji, - waga* [14]

## DropOut:

Jest to technika, która eliminuje wpływ pewnej liczby losowych neuronów w procesie uczenia. Oznacza to, że pewna część losowych neuronów jest całkowicie ignorowana w procesie uczenia, przez co model uczy się odmiennych sytuacji i nie przyzwyczaja się do konkretnych wag tych neuronów. Można to porównać do firmy, która raz na jakiś czas nie korzysta ze swoich pracowników, przez co inni pracownicy nie mogą polegać na nieobecnych, co stymuluje ich do nauki i radzenia sobie w odmiennych sytuacjach. Liczbę wyłączonych neuronów określamy przez parametr) [10].



***Rysunek 4.*** *Dropout..*

## Regularyzacja typu max-norm

Ta metoda opiera się ograniczeniu normy L2 wektora wag całej warstwy do wartości maksymalnej r.

***Zależność 1.*** *W-waga, r - parametr*

Jeśli jednak ta zależność nie jest spełniona to wartości jest przeskalowywana:

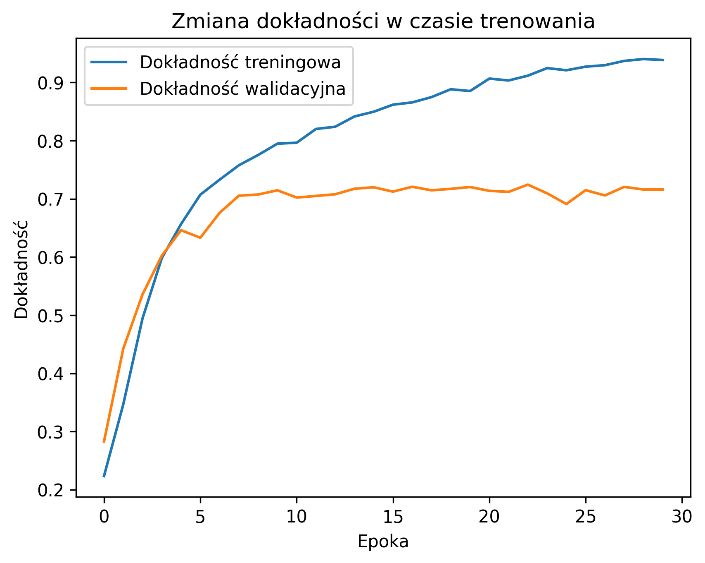
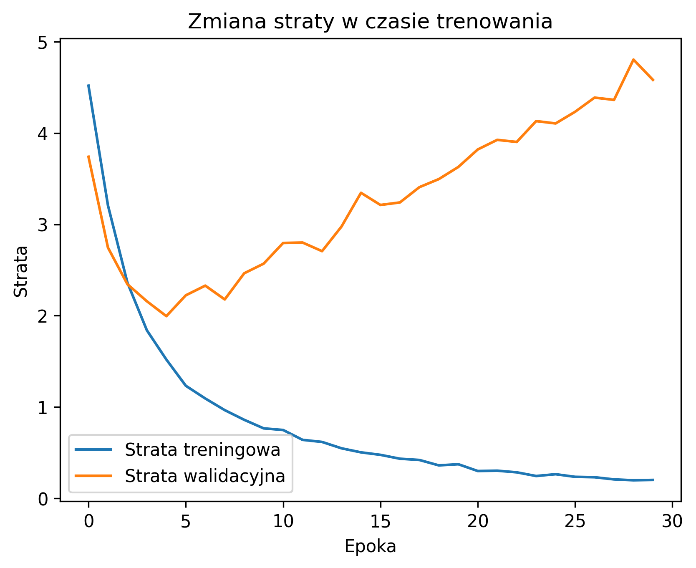
Takie ograniczenie zapobiega dużym wagom co jest m.in. przyczyną przetrenowania [15].

## Dogenerowanie danych:

Często spotykaną metodą jest również generowanie danych, które pod pewnymi aspektami różnią się od oryginałów. Na przykład obrazy można przyciemnić lub obrócić o pewien kąt, przez co model nie będzie przyzwyczajał się do konkretnego ustawienia zdjęcia. Taki zabieg często stosuje się, używając bibliotek przed samym trenowaniem, aby nie musieć generować dodatkowych danych.

## Interpretacja wyników

Bardzo ważnym, a może nawet najważniejszym etapem podczas tworzenia modelu sztucznej sieci neuronowej, jest analiza wyników modelu i sprawdzenie, jak dobrze model radzi sobie z dobranymi hiper parametrami. Analizowane będą dwa wykresy: pierwszy przedstawia wyniki funkcji straty dla każdej epoki, a drugi przedstawia zmiany dokładności modelu. Współczynnik uczenia wynosi 0,001, natomiast nie zostały zastosowane żadne środki zapobiegających przeuczeniu. Liczba epok, przez które model był trenowany, wyniosła 30.

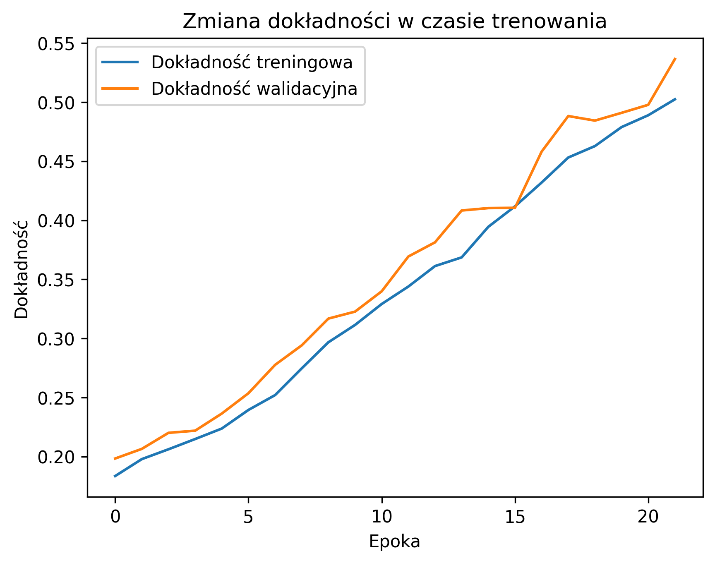
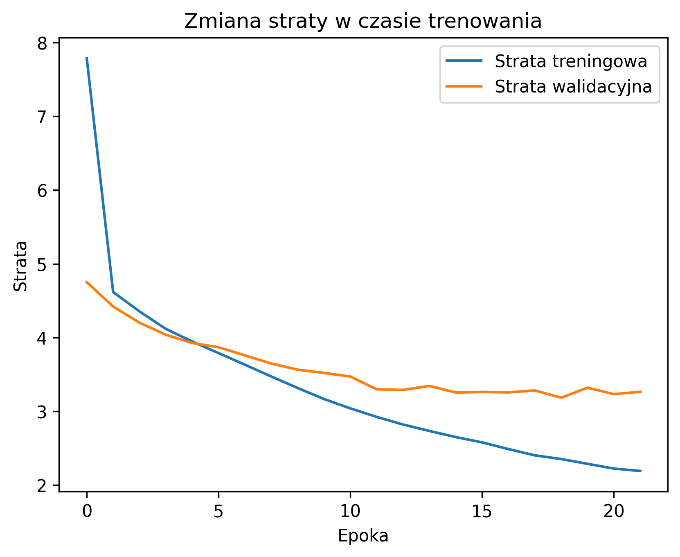


***Wykres 7,8.*** *Wykresy straty i dokładności.*

Jak widać, dla zbioru walidacyjnego największy postęp w uzyskaniu dobrego modelu pojawia się w pierwszych 5 epokach uczenia. Później, w przypadku straty, następuje pogorszenie wyników, a w przypadku dokładności wyniki utrzymują się na tym samym poziomie. Natomiast dla wyników zbioru treningowego występuje poprawa przez cały okres uczenia. Dzieje się tak, ponieważ model coraz bardziej dopasowuje się do danych na których jest uczony.

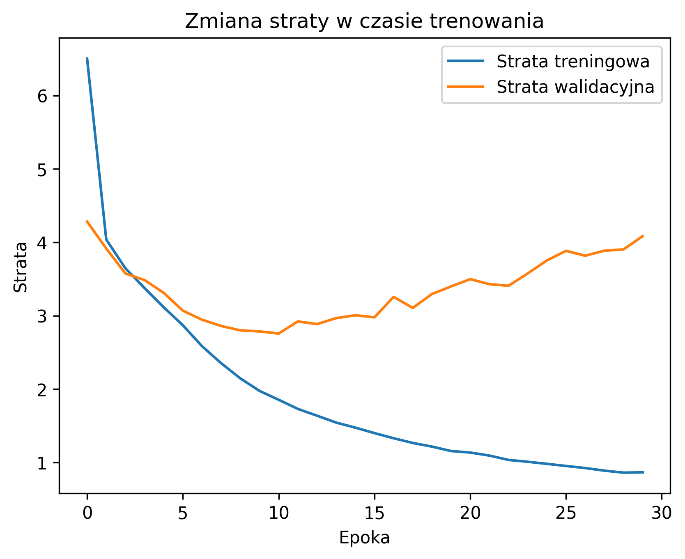
Jednak dla rzetelnej oceny modelu należy zwracać uwagę na wyniki dla zbioru walidacyjnego, którego model nie widział. To właśnie na takich danych model będzie pracował podczas okresu jego użytkowania. W przypadku zbioru walidacyjnego dokładność osiąga około 70%, co biorąc pod uwagę 721 możliwych kategorii, jest całkiem dobrym wynikiem. Natomiast znaczny wzrost straty po 5. epoce wskazuje na przeuczenie i zbytnie dopasowanie modelu do danych treningowych.

Dlatego też, modyfikując model o regularyzację L2 dla każdej warstwy ze współczynnikiem regularyzacji na poziomie 0,2 oraz zastosowaniem „dropout” na poziomie 0,3 dla 3 warstw przed wyjściem. Otrzymałem następujące wyniki:



***Wykres 9,10.*** *Wykresy straty i dokładności.*

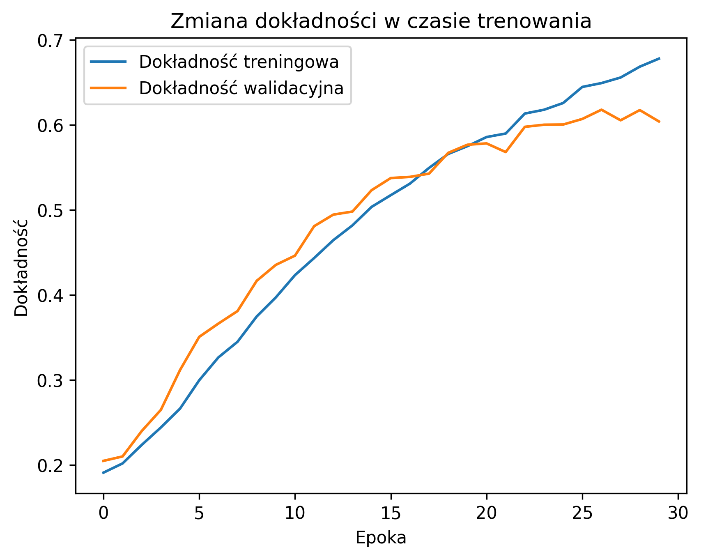
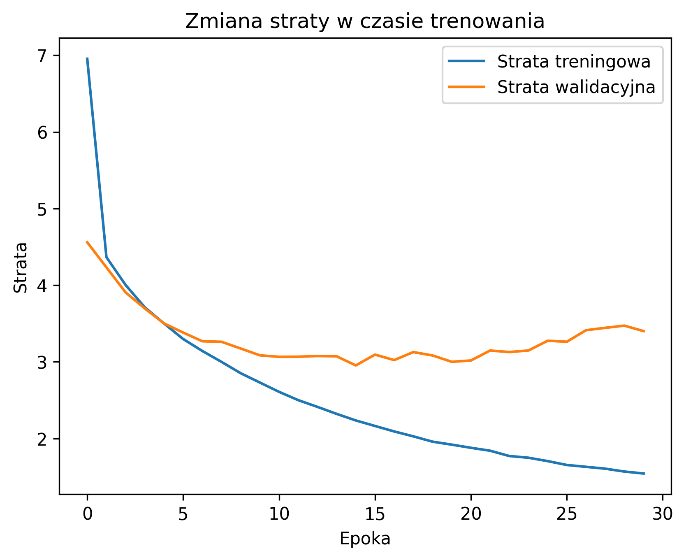
Jak widać na wykresie, udało się znacząco wy płaszczyć krzywą i usunąć nadmierne dopasowanie, jednak kosztem dokładności. Zbytnie „uogólnienie” wprowadzone do modelu spowodowało, że zaczęły pojawiać się trudności w procesie uczenia, przez co model nie osiągnął tak dobrych wyników jak wcześniej. Należy jednak zwrócić uwagę na liczbę epok, tym przypadku było to 21. Wyraźnie widać, że dla takich parametrów model nie jest optymalny i należy zmniejszając hiper parametry regularyzacji, aby uzyskać lepsze wyniki.

Obraz zawierający tekst, diagram, linia, Wykres

Opis wygenerowany automatycznie

***Wykres 10, 11.*** *Wykresy straty i dokładności.*

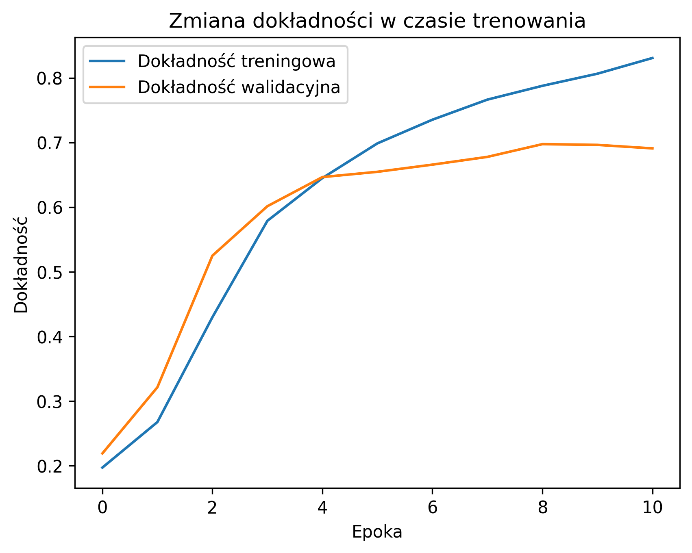
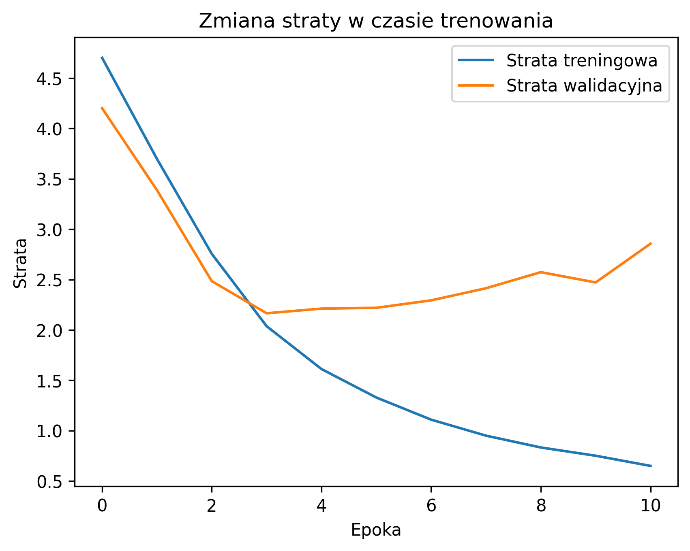
W tym przypadku widać zauważalną poprawę wyników względem poprzedniego niedouczenia. Wartość parametru „dropout” została zmniejszona do 0,2, a regularyzacja L2 osiągnęła wartość 0,15. Dokładność na poziomie 65% również nie jest złym wynikiem. Kolejny test przedstawia jeszcze większe zmniejszenie parametrów mechanizmu regularyzacji.



***Wykres 12, 13.*** *Wykresy straty i dokładności.*

W tym przypadku jednak otrzymaliśmy zbyt słaby wynik dokładności na poziomie około 60% w porównaniu do poprzednich prób. Znaczące polepszenie wyników uzyskano natomiast przy zastosowaniu funkcji aktywacji ELU, która została dokładniej opisana w poprzednim rozdziale. Zmniejszenie wymiarowości warstwy aktywacji również przyniosło małą poprawę, podobnie jak zastosowanie małego współczynnika regularyzacji L2 dla dwóch głębokich warstw.

Do uzyskania optymalnego modelu wykorzystano również metodę wcześniejszego zatrzymania trenowania, która monitoruje wybrany przez nas parametr, w tym przypadku dokładność na zbiorze walidacyjnym. Po uzyskaniu dwóch kolejnych gorszych wyników, trenowanie zostało przerwane, a wagi zostały przywrócone do stanu z najlepszymi wynikami. Wyniki wyglądają następująco:



***Wykres 13, 14.*** *Wykresy straty i dokładności.*

Jak widać, dokładność osiąga około 70%, a funkcja straty ok. 2,5. Funkcja straty może rosnąć mimo wzrostu dokładności, ponieważ wynika to z większego zróżnicowania wyników, mimo że najwyższa wartość prawdopodobieństwa dla oczekiwanej kategorii jest poprawna.

# Podsumowanie i wnioski

Zastosowanie sztucznych sieci neuronowych w procesie rekomendacji porusza wiele zagadnień związanych z przetwarzaniem danych. Większość pracy opiera się na odpowiednim przetworzeniu danych do postaci liczbowej oraz ich przekształceniu w formę najbardziej efektywną dla działania sieci. W projekcie przetwarzałem dane w lokalnej bazie danych SQL i poznałem zalety wynikające z tego podejścia, dzięki temu przekształcanie danych było sprawne.

W dużej mierze proces tworzenia modelu polegał na używaniu już bardzo dobrze przygotowanych komponentów biblioteki TensorFlow. Byłem pozytywnie zaskoczony sprawnością i szybkością działania algorytmów trenowania modelu. Mimo niezbyt mocnego sprzętu i miliona parametrów do nauczenia czas procesu uczenia umożliwiał trenowanie kilku modeli w ciągu jednego dnia. Za pomocą danych oraz dostępnych zasobów sprzętowych można stworzyć znakomite algorytmy radzące sobie z najtrudniejszymi zadaniami. Dodatkowo interfejs do modelowania jest bardzo rozbudowany, co daje mnóstwo możliwości testowania i poprawiania wyników. Było to intrygujące i wprowadzało zaciekawienie w oczekiwaniu na rezultaty.

Technologie AI różnią się od innych ścisłych nauk informatycznych tym, że nie mamy pełnej informacji i wytycznych nad tworzonym obiektem i nie rozumiemy dokładnych procesów odpowiedzialnych za wyniki danego modelu. Używając tej technologii, opieramy się na podejściu „od ogółu do szczegółu”, z czasem poprawiając działanie modelu w szczegółowych aspektach. Projektujemy modele, bazując na ogólnych wytycznych oraz wynikach i obserwacjach z podobnych prób. Wykorzystanie dotychczasowych technologii znacząco pomaga jednak w osiąganiu lepszej skuteczności.

Sztuczna inteligencja bez wątpienia jest technologią przyszłości. Pytanie, jakie się nasuwa, to gdzie może nas doprowadzić i jaka będzie rola człowieka w tej nie aż tak odległej przyszłości.

# Wykaz Literatury

|  |  |
| --- | --- |
| [1] | Marcin Szeliga, „Data Science i uczenie maszynowe” Warszawa 2017, Wydawnictwo naukowe PWN SA, Wydanie 1. (pp. 44-50) |
| [2] | „Co to jest analiza predykcyjna? | Definicja, znaczenie, przykłady”. <https://www.sap.com/poland/products/technology-platform/cloud-analytics/what-is-predictive-analytics.html>, Dostęp w dniu 10.10.2024r. |
| [3] | R. Zykov, „Retailrocket recommender system dataset”. <https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset>. Dostęp w dniu 10.10.2024r. |
| [4] | D. Melniczuk, „SQLite i Python - czy warto? - Jak badać dane?,”. <https://jakbadacdane.pl/sqlite-i-python/>, dostęp w dniu 10.10.2024r. |
| [5] | home.pl, ”Indeksy”. <https://pomoc.home.pl/baza-wiedzy/indeksy/>, dostęp w dniu 12.10.2024r. |
| [6] | Marcin Szeliga, „Praktyczne uczenie maszynowe”, Warszawa 2019, Wydawnictwo naukowe, PWN SA, Wydanie 1. |
| [7] | Andrich van Wyk, „Kodowanie funkcji cyklicznych na potrzeby głębokiego uczenia”. <https://mlengineer.substack.com/p/encoding-cyclical-features-for-deep-learning>, dostęp w dniu 25.10.2024r. |
| [8] | Laurence Moroney „Sztuczna inteligencja i uczenie maszynowe dla programistów”, Helion S.A, 2021 (pp. 139-142) |
| [9] | Laurence Moroney „Sztuczna inteligencja i uczenie maszynowe dla programistów”, Helion S.A, 2021, (pp. 144-156) |
| [10] | Kayle Gallatin, Chris Albon, „Uczenie maszynowe w Pythonie”. Helion S.A. 2024, wydanie 2 (pp. 354 – 357) |
| [11] | [Artem Oppermann](https://builtin.com/authors/artem-oppermann), „ Built In Loss Functions in Neural Networks & Deep Learning” <https://builtin.com/machine-learning/loss-functions>, dostęp w dniu 12.11.2024r. |
| [12] | „[*Przestrzeń metryczna*](https://encyklopedia.pwn.pl/haslo/;3940160)*”*, [*Encyklopedia PWN*](https://pl.wikipedia.org/wiki/Encyklopedia_PWN_(internetowa)), [Wydawnictwo Naukowe PWN](https://pl.wikipedia.org/wiki/Wydawnictwo_Naukowe_PWN), dostęp w dniu 19.11.2024r. |
| [13] | Aurelien Geron, „Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow” Helion 2017, (pp. 278-281) |
| [14] | Aurelien Geron, „Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow” Helion 2017, (pp. 303-304) |
| [15] | Aurelien Geron, „Uczenie maszynowe z użyciem Scikit-Learn i TensorFlow” Helion 2017, (pp. 306-308) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |